SIAM J. COMPUT.
Vol. 37, No. 1, pp. 1–19

ON THE COMPUTATIONAL COMPLEXITY OF THE FORCING CHROMATIC NUMBER*

FRANK HARARY[†], WOLFGANG SLANY[‡], AND OLEG VERBITSKY[§]

Abstract. We consider vertex colorings of graphs in which adjacent vertices have distinct colors. A graph is s-chromatic if it is colorable in s colors and any coloring of it uses at least s colors. The forcing chromatic number $F_\chi(G)$ of an s-chromatic graph G is the smallest number of vertices which must be colored so that, with the restriction that s colors are used, every remaining vertex has its color determined uniquely. We estimate the computational complexity of $F_\chi(G)$ relating it to the complexity class US introduced by Blass and Gurevich [*Inform. Control*, 55 (1982), pp. 80–88]. We prove that recognizing whether $F_\chi(G) \le 2$ is US-hard with respect to polynomial-time many-one reductions. Moreover, this problem is coNP-hard even under the promises that $F_\chi(G) \le 3$ and G is 3-chromatic. On the other hand, recognizing whether $F_\chi(G) \le k$, for each constant k, is reducible to a problem in US via a disjunctive truth-table reduction. Similar results are obtained also for forcing variants of the clique and the domination numbers of a graph.

Key words. computational complexity, complexity classes, combinatorial forcing, chromatic number of a graph, unique satisfiability

AMS subject classifications. 03D15, 05C15, 05C69, 68Q15, 68Q17, 68R10

DOI. 10.1137/050641594

1. Introduction.

The vertex set of a graph G will be denoted by $V(G)$. An s-*coloring* of G is a map from $V(G)$ to $\{1, 2, \dots, s\}$. A coloring c is *proper* if $c(u) \ne c(v)$ for any adjacent vertices u and v. A graph G is s-*colorable* if it has a proper s-coloring. The minimum s for which G is s-colorable is called the *chromatic number* of G and denoted by $\chi(G)$. If $\chi(G) = s$, then G is called s-*chromatic.*

A *partial coloring* of G is any map from a subset of $V(G)$ to the set of positive integers. Suppose that G is s-chromatic. Let c be a proper s-coloring and p be a partial coloring of G. We say that p *forces* c if c is a unique extension of p to a proper s-coloring. The domain of p will be called a *defining set for* c. We call $D \subseteq V(G)$ a *forcing set in* G if this set is defining for some proper s-coloring of G. The minimum cardinality of a forcing set is called the *forcing chromatic number* of G and denoted by $F_\chi(G)$. This graph invariant was introduced by Harary in [22]. Here we study its computational complexity.

To establish the hardness of computing $F_\chi(G)$, we focus on the respective slice decision problems which are defined for each nonnegative integer k as follows:

FORCE$_\chi(k)$
Given: a graph G.
Decide if: $F_\chi(G) \le k$.

The cases of $k = 0$ and $k = 1$ are tractable. It is clear that $F_\chi(G) = 0$ iff $\chi(G) = 1$,

*Received by the editors September 30, 2005; accepted for publication (in revised form) August 10, 2006; published electronically March 30, 2007. A preliminary version of this paper appeared as [26].

http://www.siam.org/journals/sicomp/37-1/64159.html

[†]The author is deceased. Former address: Computer Science Department, New Mexico State University, Las Cruces, NM 88003.

[‡]Institut für Softwaretechnologie, Technische Universität Graz, A-8010 Graz, Austria (wsi@ist.tugraz.at).

[§]Institut für Informatik, Humboldt Universität Berlin, D-10099 Berlin, Germany (verbitsk@informatik.hu-berlin.de). This author's work was supported by an Alexander von Humboldt fellowship.

that is, G is empty. Furthermore, $F_\chi(G) = 1$ iff $\chi(G) = 2$ and G is connected, that is, G is a connected bipartite graph. Thus, we will restrict attention to the case where $k \geq 2$. Since there is a simple reduction of $\textsc{Force}_\chi(k)$ to $\textsc{Force}_\chi(k+1)$ (see Lemma 3.2), it would suffice to show that even $\textsc{Force}_\chi(2)$ is computationally hard. This is indeed the case.

Let 3COL denote the set of 3-colorable graphs and U3COL the set of those graphs in 3COL having a unique, up to renaming colors, proper 3-coloring. First of all, note that a hardness result for $\textsc{Force}_\chi(2)$ is easily derivable from two simple observations:

$$\begin{array}{ll} \text{if } F_\chi(G) \leq 2, \text{ then } G \in \text{3COL}; \\ \text{if } G \in \text{U3COL}, \text{ then } F_\chi(G) \leq 2. \end{array} \tag{1}$$

The set 3COL was shown to be NP-complete at the early stage of the NP-completeness theory in [45, 18] by reduction from SAT, the set of satisfiable Boolean formulas. We will benefit from the use of a well-known stronger fact: There is a polynomial-time many-one reduction p from SAT to 3COL which is *parsimonious*; that is, any Boolean formula Φ has exactly as many satisfying assignments to variables as the graph $p(\Phi)$ has proper 3-colorings. (Colorings obtainable from one another by renaming colors are not distinguished.) In particular, if Φ has a unique satisfying assignment, then $p(\Phi) \in$ U3COL and hence $F_\chi(p(\Phi)) \leq 2$, while if Φ is unsatisfiable, then $p(\Phi) \notin$ 3COL and hence $F_\chi(p(\Phi)) > 2$.

Valiant and Vazirani [47] designed a polynomial-time computable randomized transformation r of the set of Boolean formulas such that, if Φ is a satisfiable formula, then with a nonnegligible probability the formula $r(\Phi)$ has a unique satisfying assignment, while if Φ is unsatisfiable, then $r(\Phi)$ is unsatisfiable with probability 1. Combining r with the parsimonious reduction p of SAT to 3COL, we arrive at the conclusion that $\textsc{Force}_\chi(2)$ is NP-hard with respect to *randomized* polynomial-time many-one reductions. As a consequence, the forcing chromatic number is not computable in polynomial time unless any problem in NP is solvable by a polynomial-time Monte Carlo algorithm with one-sided error.

We aim at determining the computational complexity of $F_\chi(G)$ more precisely. Our first result establishes the hardness of $\textsc{Force}_\chi(2)$ with respect to *deterministic* polynomial-time many-one reductions. Unless we explicitly state otherwise, all reductions considered in this paper are deterministic many-one reductions. The complexity class US, introduced by Blass and Gurevich [4], consists of languages L for which there is a polynomial-time nondeterministic Turing machine N such that a word w belongs to L iff N on input w has exactly one accepting computation path. Denote the set of Boolean formulas with exactly one satisfying assignment by USAT. This set is complete for US. As is easily seen, U3COL belongs to US and, by the parsimonious reduction from SAT to 3COL, U3COL is another US-complete set. By the Valiant–Vazirani reduction, the US-hardness under polynomial-time many-one reductions implies the NP-hardness under randomized reductions, and hence the former hardness concept should be considered stronger. It is known that US includes coNP [4], and this inclusion is proper unless the polynomial-time hierarchy collapses [42]. Thus, US-hardness implies coNP-hardness. We prove that the problem $\textsc{Force}_\chi(2)$ is US-hard.

Note that this result is equivalent to the reducibility of U3COL to $\textsc{Force}_\chi(2)$. Such a reduction would follow from the naive hypothesis, which may be suggested by (1), that a 3-chromatic G is in U3COL iff $F_\chi(G) = 2$. It should be stressed that the latter is far from being true in view of Lemma 2.7.3 below.

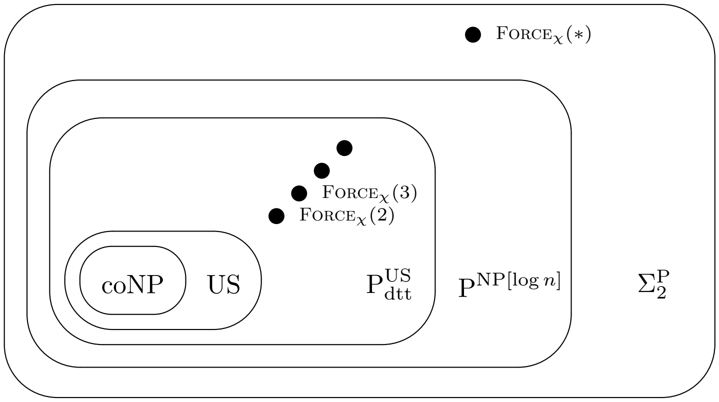

FORCE$_\chi(*)$
FORCE$_\chi(3)$
FORCE$_\chi(2)$
coNP
US
$\mathrm{P}^{\mathrm{US}}_{\mathrm{dtt}}$
$\mathrm{P}^{\mathrm{NP}[\log n]}$
Σ^{P}_2

FIG. 1. *Location of the slice decision problems for $F_\chi(G)$ in the hierarchy of complexity classes.*

On the other hand, we are able to estimate the complexity of each FORCE$_\chi(k)$ from above by putting this family of problems into a complexity class which is a natural extension of US. We show that, for each $k \geq 2$, the problem FORCE$_\chi(k)$ is reducible to a set in US via a polynomial-time disjunctive truth-table reduction (*dtt-reduction* for brevity; see section 2 for definitions). This improves on the straightforward inclusion of FORCE$_\chi(k)$ in Σ^{P}_2.

Denote the class of decision problems reducible to US under dtt-reductions by $\mathrm{P}^{\mathrm{US}}_{\mathrm{dtt}}$. As shown by Chang, Kadin, and Rohatgi [5], $\mathrm{P}^{\mathrm{US}}_{\mathrm{dtt}}$ is strictly larger than US unless the polynomial time hierarchy collapses to its third level. The position of the problems under consideration in the hierarchy of complexity classes is shown in Figure 1, where $\mathrm{P}^{\mathrm{NP}[\log n]}$ denotes the class of decision problems solvable by polynomial-time Turing machines with logarithmic number of queries to an NP oracle. The latter class coincides with the class of problems polynomial-time truth-table reducible to NP; see [29]. Another relation of $\mathrm{P}^{\mathrm{US}}_{\mathrm{dtt}}$ to known complexity classes is $\mathrm{P}^{\mathrm{US}}_{\mathrm{dtt}} \subseteq \mathrm{C}_=\mathrm{P} \subseteq \mathrm{PP}$, where a language L is in PP (resp., $\mathrm{C}_=\mathrm{P}$) if it is recognizable by a nondeterministic Turing machine M with the following acceptance criterion: An input word w is in L iff at least (resp., precisely) half of the computing paths of M on w are accepting. This inclusion follows from the facts that $\mathrm{US} \subseteq \mathrm{C}_=\mathrm{P}$ and that $\mathrm{C}_=\mathrm{P}$ is closed under dtt-reductions (see [30, Theorem 9.9]).

In a recent paper [28], Hatami and Maserrat obtain a result that, in a sense, is complementary to our work, and for this reason is also shown in Figure 1. Let FORCE$_\chi(*) = \{\, (G,k) : F_\chi(G) \leq k\}$. The authors of [28] prove that the recognition of membership in FORCE$_\chi(*)$ is a Σ^{P}_2-complete problem. Note that [28] and our paper use different techniques and that, moreover, the two approaches apparently cannot be used in place of one another.

Our next result gives more detailed information about the hardness of FORCE$_\chi(2)$. Note that, if $\chi(G) = 2$, then $F_\chi(G)$ is equal to the number of connected components of G. It turns out that the knowledge that $\chi(G) = 3$ does not help in computing $F_\chi(G)$. Moreover, it is hard to recognize whether or not $F_\chi(G) = 2$, even if it is known that $F_\chi(G) \leq 3$. Stating these strengthenings, we relax our hardness concept from US-hardness to coNP-hardness. (As already mentioned, the former implies the latter, but the converse is not true unless the polynomial time hierarchy collapses.) Thus, we prove that the problem FORCE$_\chi(2)$ is coNP-hard even under the promises that

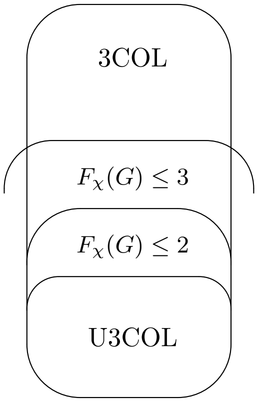


FIG. 2. *The class of graphs* $\{G : F_\chi(G) \le 2\}$ *and surroundings.*

$G \in$ 3COL and $F_\chi(G) \le 3$ (see Figure 2). Note that the Valiant–Vazirani reduction implies no kind of hardness result for the promise version of $\text{FORCE}_\chi(2)$.

In fact, many other graph characteristics also have natural forcing variants. Recall that a *clique* in a graph is a set of pairwise adjacent vertices. The maximum number of vertices of G in a clique is denoted by $\omega(G)$ and called the *clique number* of G. A clique is *optimal* if it consists of $\omega(G)$ vertices. A set of vertices is called *forcing* if it is included in a unique optimal clique. We denote the minimum cardinality of a forcing set by $F_\omega(G)$ and call it the *forcing clique number* of G.

Furthermore, we say that a vertex of a graph G *dominates* itself and any adjacent vertex. A set $D \subseteq V(G)$ is called *dominating* if every vertex of G is dominated by a vertex in D. The *domination number* of G, denoted by $\gamma(G)$, is the minimum cardinality of a dominating set of G. Similarly to the above, a *forcing set* of vertices is one included in a unique optimal dominating set. The minimum cardinality of a forcing set is denoted by $F_\gamma(G)$ and called the *forcing domination number* of G. This graph invariant is introduced and studied by Chartrand et al. [6].

For the forcing clique and domination numbers we consider the respective slice decision problems $\text{FORCE}_\omega(k)$ and $\text{FORCE}_\gamma(k)$ and show the same relation between them and the class US that we have for the forcing chromatic number. Actually, the dtt-reducibility to US is proved for all the three numbers by a uniform argument. However, the US-hardness with respect to many-one reductions for ω and γ is proved differently than for χ. The case of ω and γ seems combinatorially simpler because of the following equivalence: A graph G has a unique optimal clique iff $F_\omega(G) = 0$ and similarly with γ. The study of *unique optimum (UO) problems* was initiated by Papadimitriou [39]. Due to the US-hardness of the UO CLIQUE and UO DOMINATING SET problems, we are able to show the US-hardness of $\text{FORCE}_\omega(k)$ and $\text{FORCE}_\gamma(k)$ using only well-known standard reductions, whereas for $\text{FORCE}_\chi(k)$ we use somewhat more elaborate reductions involving graph products.

Overview of previous related work.

Forcing chromatic number of particular graphs. Let K_n (resp., C_n, P_n) denote the complete graph (resp., the cycle, the path) on n vertices. As a simple exercise, we have $F_\chi(C_{2m+1}) = m + 1$. Mahmoodian, Naserasr, and Zaker [37] compute the forcing chromatic number of several Cartesian products: $F_\chi(C_{2m+1} \times K_2) = m +$

1, $F_\chi(C_m \times K_n) = m(n-3)$ and $F_\chi(P_m \times K_n) = m(n-3)+2$ if $n \geq 6$, and $F_\chi(K_m \times K_n) = m(n-m)$ if $n \geq m^2$. Mahdian et al. [36] determine a few missing values: $F_\chi(C_m \times K_3) = \lfloor m/2 \rfloor + 1$ and, if m is even, $F_\chi(C_m \times K_5) = 2m$. On the other hand, the asymptotics of $F_\chi(K_n \times K_n)$ are unknown (a problem having arisen in research on Latin squares; see below). The best lower and upper bounds $\lfloor 4(n-2)/3 \rfloor \leq F_\chi(K_n \times K_n) \leq n^2/4$ are obtained, respectively, in [31] for $n \geq 8$ and [14] for all n. Our results show that no general approach for efficient computing of the forcing chromatic number is possible unless NP = P (and even US = P).

Latin squares and the complexity of recognizing a forcing coloring. A *Latin square of order* n is an $n \times n$ matrix with entries in $\{1, 2, \ldots, n\}$ such that every row and column contains all the n numbers. In a *partial Latin square* some entries may be empty, and every number occurs in any row or column at most once. A partial Latin square is called a *critical set* if it can be completed to a Latin square in a unique way. Colbourn, Colbourn, and Stinson [11] proved that recognition of whether a given partial Latin square L is a critical set is coNP-hard. Moreover, the problem is coNP-complete even if one extension of L to a Latin square is known. The result of [11] is strengthened in [16].

As is observed in [37], there is a natural one-to-one correspondence between Latin squares of order n and proper n-colorings of the Cartesian square $K_n \times K_n$, which matches critical sets and forcing colorings.[1] It follows that it is coNP-hard to recognize whether a given partial coloring p of a graph is forcing. Moreover, even if this problem is restricted to graphs $K_n \times K_n$ and one extension of p to a proper n-coloring is given, the problem is coNP-complete (see also Proposition 2.8 below).

Complexity of the forcing matching number. Given a graph G with a perfect matching, Harary, Klein, and Živković [25] define its forcing matching number as the minimum size of a forcing set of edges, where the latter is a set contained in a unique perfect matching. Denote this number by $F_\nu(G)$. Let $\textsc{Force}_\nu(k)$ be the problem of recognition, given G, if $F_\nu(G) \leq k$, and let $\textsc{Force}_\nu(*)$ be the variant of the same problem with k given as a part of an input. From the polynomial-time solvability of the perfect matching problem, it easily follows that each $\textsc{Force}_\nu(k)$ is polynomial-time solvable and that $\textsc{Force}_\nu(*)$ is in NP. Afshani, Hatami, and Mahmoodian [3], using an earlier result by Adams, Mahdian, and Mahmoodian [1], prove that the latter problem is NP-complete.

Variety of combinatorial forcing numbers. Critical sets have been studied since the seventies (Nelder [38]). The forcing chromatic, domination, and matching numbers attracted attention of researchers in the nineties. In fact, a number of other problems in diverse areas of combinatorics have a similar forcing nature. Defining sets in block designs (Gray [19]) have a rather rich bibliography. Other graph invariants whose forcing versions have appeared in the literature are the orientation number (Chartrand et al. [7]); the geodetic number, the hull number, the dimension of a graph (Chartrand and Zhang in, respectively, [8, 9, 10]); and this list is still inexhaustive. As a general concept, combinatorial forcing was presented by Harary in a series of talks, e.g., [22].

Unique colorability. The concept of a uniquely colorable graph was introduced by Harary, Hedetniemi, and Robinson [23], [24]. Complexity-theoretic concepts related to this combinatorial phenomenon were introduced by Valiant and Vazirani [47] and Blass and Gurevich [4]. However, complexity theorists prefer dealing with USAT, the uniqueness version of the archetypical SATisfiability problem. The results established for USAT, as US- and coNP-hardness under many-one reductions and NP-hardness

[1] Such a correspondence is also well known for edge colorings of the complete bipartite graph $K_{n,n}$.

under randomized reductions, carry through for U3COL by means of the parsimonious many-one reduction of SAT to 3COL. A direct, purely combinatorial way to show the coNP-hardness of U3COL is given by a result of Greenwell and Lovász ([20]; see also [32, Theorem 8.5]). They prove, in particular, that if a connected graph G is not 3-colorable, then the categorical product $G \cdot K_3$ is a uniquely colorable graph. On the other hand, if G is 3-colorable, then $G \cdot K_3$ can be colored in two different ways. The latter follows from a simple observation: Any proper 3-coloring of one of the factors efficiently induces a proper 3-coloring of the product $G_1 \cdot G_2$. By the NP-completeness of 3COL, we arrive at the conclusion that the problem of recognizing, given a graph H and its proper 3-coloring, whether or not H is uniquely 3-colorable is coNP-complete. Later this complexity-theoretic fact was observed by Dailey [15], who uses an identical combinatorial argument.

Organization of the paper. In section 2 we define the categorical and Cartesian graph products, which will play an important role in our proofs, and prove some preliminary lemmas about forcing sets in product graphs. We also state a few basic bounds for $F_\chi(G)$ and determine the complexity of recognizing whether a given set of vertices in a graph is forcing (the latter result is not used later in the paper but is worth noticing). The hardness of $\textsc{Force}_\chi(k)$ is established in section 3. A closer look at $\textsc{Force}_\chi(2)$ is taken in section 4. In section 5, using a combinatorial result of Hajiabolhassan et al. [21], we analyze the complexity of a related graph invariant, namely, the largest cardinality of an inclusion-minimal forcing set. Before taking into consideration the forcing clique and domination numbers, we suggest a general setting for forcing-combinatorial numbers in section 6. It is built upon the standard formal concept of an NP optimization problem. We benefit from this formal framework in some proofs. Section 7 is devoted to the forcing clique and domination numbers. The dtt-reducibility of $\textsc{Force}_\pi(k)$ to US for $\pi \in \{\chi, \omega, \gamma\}$ is shown in section 8. Section 9 contains a concluding discussion and some open questions.

2. Background.

2.1. Basics of complexity theory. We suppose that the discrete structures under consideration are encoded by binary words. For example, graphs are naturally representable by their adjacency matrices. The set of all binary words is denoted by $\{0,1\}^*$. A decision problem is identified with a language, i.e., a subset of $\{0,1\}^*$, consisting of all yes-instances. A *many-one reduction* of a problem X to a problem Y is a map $r : \{0,1\}^* \to \{0,1\}^*$ such that $x \in X$ iff $r(x) \in Y$. If $r(x)$ is computable in time bounded by a polynomial in the length of x, the reduction is called *polynomial-time.* We write $X \leq_{\mathrm{m}}^{\mathrm{P}} Y$ to say that there is a polynomial-time many-one reduction from X to Y.

Let C be a class of decision problems (or languages). A problem Z is called *C-hard* if any X in C is $\leq_{\mathrm{m}}^{\mathrm{P}}$-reducible to Z. A problem Z is called *C-complete* if Z is C-hard and belongs to C. USAT and U3COL are examples of US-complete problems.

A *disjunctive truth-table reduction* (or *dtt-reduction*) of a language X to a language Y is a transformation which takes any word x to a set of words $y_1, \ldots, y_m$ so that $x \in X$ iff $y_i \in Y$ for at least one $i \leq m$. We write $X \leq_{\mathrm{dtt}}^{\mathrm{P}} Y$ to say that there is such a polynomial-time reduction from X to Y.

If C is a class of languages and $\leq$ is a reducibility, then $C \leq X$ means that $Y \leq X$ for all Y in C (i.e., X is C-hard under $\leq$), and $X \leq C$ means that $X \leq Y$ for some Y in C.

The formal framework of promise problems is developed in [44]. Let Y and Q

be languages. Whenever referring to a *decision problem Y under the promise Q*, we mean that membership in Y is to be decided only for inputs in Q. A reduction r of an ordinary decision problem X to a problem Y under the promise Q is a usual many-one reduction from X to Y with the additional requirement that $r(x) \in Q$ for all x. This definition allows us to extend the notion of C-hardness to promise problems.

A polynomial-time computable function $h : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ is called an *AND_2 function for a language Z* if for any pair x, y we have both x and y in Z iff $h(x,y)$ is in Z. Such an h is an *OR_2 function for Z* if we have at least one of x and y in Z iff $h(x,y)$ is in Z.

2.2. Graph products. Let $E(G)$ denote the set of edges of a graph G. Given two graphs G_1 and G_2, we define a product graph on the vertex set $V(G_1) \times V(G_2)$ in two ways. Vertices (u_1, u_2) and (v_1, v_2) are adjacent in the *Cartesian product* $G_1 \times G_2$ if either $u_1 = v_1$ and $\{u_2, v_2\} \in E(G_2)$ or $u_2 = v_2$ and $\{u_1, v_1\} \in E(G_1)$. They are adjacent in the *categorical product* $G_1 \cdot G_2$ if both $\{u_1, v_1\} \in E(G_1)$ and $\{u_2, v_2\} \in E(G_2)$.

A set $V(G_1) \times \{v\}$ for $v \in V(G_2)$ will be called the *G_1-layer of v*, and a set $\{u\} \times V(G_2)$ for $u \in V(G_1)$ will be called the *G_2-layer of u*.

LEMMA 2.1 (Sabidussi [43]; see also [32, Theorem 8.1]). $\chi(G \times H) = \max\{\chi(G), \chi(H)\}$.

If c is a proper coloring of G, it is easy to see that $c^*(x,y) = c(x)$ is a proper coloring of the categorical product $G \cdot H$. We will say that *c induces c^**. Similarly, any proper coloring of H induces a proper coloring of $G \cdot H$. This implies the following well-known fact.

LEMMA 2.2. $\chi(G \cdot H) \leq \min\{\chi(G), \chi(H)\}$.

The next proposition shows that the Cartesian and the categorical products are, respectively, AND$_2$ and OR$_2$ functions for 3COL (see [34] for an exposition of AND and OR functions).

PROPOSITION 2.3.

1. *$G \times H \in$ 3COL iff both G and H are in 3COL.*
2. *$G \cdot H \in$ 3COL iff at least one of G and H is in 3COL.*

Proof. Item 1 is straightforward by Lemma 2.1. However, item 2 does not follow from Lemma 2.2 alone, because the equality $\chi(G \cdot H) = \min\{\chi(G), \chi(H)\}$ is still an unproven conjecture made by Hedetniemi. Luckily, the conjecture is known to be true for 4-chromatic graphs (El-Zahar and Sauer [17]; see also [32, section 8.2]), and this suffices for our claim. □

2.3. Preliminary lemmas.

LEMMA 2.4 (Greenwell and Lovász [20]). *Let G be a connected graph with $\chi(G) > n$. Then $G \cdot K_n$ is uniquely n-colorable.*

The proof can be found also in [32, Theorem 8.5]. We will use not only Lemma 2.4 itself but also a component of its proof stated as the next lemma. We call a partial coloring *injective* if no two vertices have the same color.

LEMMA 2.5. *Let G be a connected graph and p be an injective coloring of a K_n-layer of $G \cdot K_n$. Then p forces the K_n-induced coloring of $G \cdot K_n$.*

Proof. Suppose that p is an injective coloring of the K_n-layer of $u \in V(G)$ and that a proper n-coloring c of $G \cdot K_n$ extends p. Let $v \in V(G)$ be adjacent to u. It follows easily that $c(v,i) = c(u,i)$ for any $i \in V(K_n)$. Since G is connected, c is forced by p to be monochromatic on each G-layer. □

LEMMA 2.6. *Any proper 3-coloring of $K_3 \cdot K_3$ is induced by one of the two factors K_3.*

Proof. Let $V(K_3) = \{1,2,3\}$ and denote $L_i = \{(i,1),(i,2),(i,3)\}$. Suppose that

c is a 3-coloring of $K_3 \cdot K_3$. Consider c on one of the layers, say L_2. If $|c(L_2)| = 3$, then c is induced by the second factor by Lemma 2.5. Assume that $|c(L_2)| = 2$, for example, $c(2,1) = 1$, $c(2,2) = 2$, and $c(2,3) = 2$. Then $c(1,2) = c(3,3) = 3$ is forced, contradictory to the fact that these vertices are adjacent. There remains the possibility that $|c(L_2)| = 1$, for example, $c(2,1) = c(2,2) = c(2,3) = 2$. Color 2 can occur neither in L_1 nor in L_3. Assume that one of these layers has both colors 1 and 3, say, $c(1,1) = 1$ and $c(1,2) = 3$. This forces $c(3,3) = 2$, a contradiction. Thus, $|c(L_2)| = 1$ is possible only if L_1, L_2, and L_3 are all monochromatic and have pairwise distinct colors. This is the coloring induced by the first factor. □

2.4. A few basic bounds. We call two s-colorings *equivalent* if they are obtainable from one another by permutation of colors. Proper s-colorings of a graph G are equivalent if they determine the same partition of $V(G)$ into s independent sets. Let $N_\chi(G)$ denote the number of such partitions for $s = \chi(G)$. Thus, $N_\chi(G)$ is equal to the number of inequivalent proper s-colorings of s-chromatic G, while the total number of such colorings is equal to $\chi(G)!\,N_\chi(G)$. A graph G is *uniquely colorable* if $N_\chi(G) = 1$. In particular, $G \in$ U3COL iff $\chi(G) = 3$ and $N_\chi(G) = 1$.

Lemma 2.7.

1. $\chi(G) - 1 \le F_\chi(G) \le \log_2 N_\chi(G) + \log_2(\chi(G)!)$.
2. *If $N_\chi(G) = 1$, then $F_\chi(G) = \chi(G) - 1$.*
3. *For any k, there is a 3-chromatic graph G_k on $4k+2$ vertices with $F_\chi(G_k) = 2$ and $N_\chi(G_k) = 2^{k-1} + 1$.*

The lower bound in item 1 is sharp by item 2. The upper bound is sharp because, for example, $F_\chi(mK_2) = m$ while $N_\chi(mK_2) = 2^{m-1}$, where mK_2 denotes the graph consisting of m isolated edges. Item 3 shows that the converse of item 2 is false. It is also worth noting that both the lower and the upper bounds in item 1 are computationally hard, even if one is content with finding an approximate value. The NP-hardness of approximation of the chromatic number is established in [35]. An NP-hardness result for approximately computing $\log_2 N_\chi(G)$, even for 3-chromatic graphs, is obtained in [48].

Proof. Item 2 and the lower bound in item 1 are obvious. To prove the upper bound in item 1, we have to show that an s-chromatic graph G has a forcing set of at most $l = \lfloor \log_2(s!N_\chi(G)) \rfloor$ vertices $v_1, v_2, \ldots$. Let C_1 be the set of all $s!N_\chi(G)$ proper s-colorings of G. We choose vertices $v_1, v_2, \ldots$ one by one as follows. Let $i \ge 1$. Assume that the preceding $i-1$ vertices have been chosen and that a set of colorings C_i has been defined and has at least two elements. We set v_i to be an arbitrary vertex such that not all colorings in C_i coincide at v_i. Furthermore, we assign v_i a color $p(v_i)$ occurring in the multiset $\{\, c(v_i) : c \in C_i \}$ least frequently. Finally, we define $C_{i+1} = \{\, c \in C_i : c(v_i) = p(v_i) \}$. Note that $|C_{i+1}| \le |C_i|/2$. We eventually have $|C_{i+1}| = 1$ for some $i \le l = \lfloor \log_2 |C_1| \rfloor$. For this i, denote the single coloring in C_{i+1} by c. By construction, $v_1, \ldots, v_i$ is defining for c.

To prove item 3, consider $H = K_3 \times K_2$. This graph has two inequivalent colorings c_1 and c_2, shown in Figure 3. Let $u, v, w \in V(H)$ be as in Figure 3. Note that a partial coloring $p_1(u) \ne p_1(v)$ forces c_1 or its equivalent and that $p_2(u) = p_2(v) \ne p_2(w)$ forces c_2.

Let G_k consist of k copies of H with all u and all v identified; that is, G_k has $4k + 2$ vertices. Since the set $\{u, v\}$ stays forcing in G_k, we have $F_\chi(G_k) = 2$. If u and v are assigned the same color, we are free to assign each copy of w any of the two remaining colors. It follows that $N_\chi(G_k) = 2^{k-1} + 1$. □

2.5. Complexity of recognizing a forcing set.

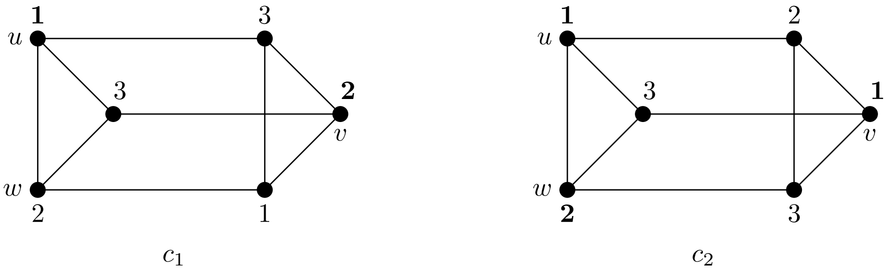


FIG. 3. *Proper 3-colorings of $K_3 \times K_2$.*

PROPOSITION 2.8. *The problem of recognizing, given a graph G and two vertices $u, v \in V(G)$, whether or not $\{u, v\}$ is a forcing set is US-complete.*

Proof. Suppose that $\chi(G) \geq 3$, for else the question is efficiently decidable. Set $p(u) = 1$ and $p(v) = 2$. Obviously, $\{u, v\}$ is a forcing set iff p forces a proper 3-coloring of G. Verification of the latter condition is clearly in US.

We now describe a reduction R of the US-complete problem U3COL to the problem under consideration. If an input graph G is empty, let $R(G)$ be an arbitrary graph with two vertices which are not a forcing set. Otherwise, let u and v be the lexicographically first pair of adjacent vertices in G. We set $R(G) = (G, u, v)$. It is not hard to see that $G \in$ U3COL iff $R(G)$ consists of a graph and a 2-vertex forcing set in it. □

3. Complexity of $F_\chi(G)$: A lower bound.

THEOREM 3.1. *For each $k \geq 2$, the problem* FORCE$_\chi(k)$ *is US-hard. Moreover, this holds true even if we consider only connected graphs.*

We first observe that the family of problems FORCE$_\chi(k)$ is linearly ordered with respect to $\leq_{\mathrm{m}}^{\mathrm{P}}$-reducibility. A simple reduction showing this does not preserve connectedness of graphs. However, if we restrict ourselves to connected graphs, we are able to show that FORCE$_\chi(2)$ remains the minimum element in this order. We then prove that FORCE$_\chi(2)$ is US-hard (even for connected graphs).

LEMMA 3.2. FORCE$_\chi(k) \leq_m^P$ FORCE$_\chi(k+1)$.

Proof. Given a nonempty graph G, we add one isolated vertex to it. Denoting the result by $G + K_1$, it is enough to notice that $F_\chi(G + K_1) = F_\chi(G) + 1$. □

LEMMA 3.3. *Let $k \geq 2$. Then* FORCE$_\chi(2)$ *reduces to* FORCE$_\chi(k)$ *even if we consider the problems only for connected graphs.*

Proof. Let G be a graph on n vertices and $m \leq n$. Writing $H = G \oplus mK_2$, we mean that

- $V(H) = \{v_1, \ldots, v_n\} \cup \bigcup_{i=1}^m \{a_i, b_i\}$,
- H induces on $\{v_1, \ldots, v_n\}$ a graph isomorphic to G,
- $\{v_i, a_i\}$ and $\{a_i, b_i\}$ for all $i \leq m$ are edges of H, and
- H has no other edges.

Suppose that $\chi(G) \geq 3$ and $H = G \oplus mK_2$. Let us check that $F_\chi(H) \leq 2 + m$ if $F_\chi(G) = 2$ and $F_\chi(H) \geq 3 + m$ if $F_\chi(G) \geq 3$. This will give us the following reduction of FORCE$_\chi(2)$ to FORCE$_\chi(k)$: Given G, construct an $H = G \oplus (k-2)K_2$.

Let $F_\chi(G) = 2$. Note that G must be 3-chromatic. To show that $F_\chi(H) \leq 2 + m$, we construct a forcing set of $2+m$ vertices. We will identify G and its copy spanned by $\{v_1, \ldots, v_n\}$ in H. Let v_j and v_l be two vertices such that a partial coloring $p(v_j) = 1$ and $p(v_l) = 2$ forces a proper 3-coloring c of G. Color each b_i differently from $c(v_i)$. Then $\{v_j, v_l, b_1, \ldots, b_m\}$ becomes a forcing set in H.

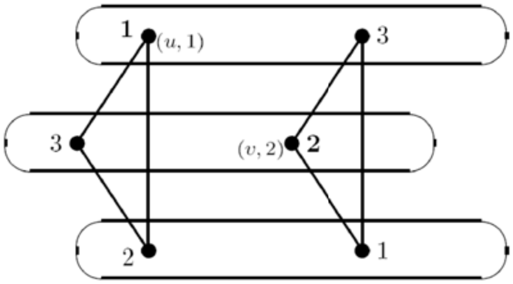


FIG. 4. *Proof of Lemma* 3.5 *(Case* 1*).*

Let $F_\chi(G) \geq 3$. Clearly, any forcing set contains all b_i's. We hence have to show that no set $S = \{b_1, \ldots, b_m, x, y\} \subset V(H)$ is forcing in H. We will assume that $\chi(G) = 3$, as otherwise the claim is easy. Suppose that $x \in \{v_j, a_j\}$ and $y \in \{v_l, a_l\}$ for some j, l and that S is augmented with a coloring p extendable to a proper 3-coloring c of H. Let c' denote the restriction of c to $\{v_1, \ldots, v_n\}$. Since c' is not a unique coloring of G compatible with p (otherwise $\{v_j, v_l\}$ would be a forcing set for G), there is another such coloring c''. Obviously, c'' and p have a common extension to a proper coloring of H. Thus, S is not a forcing set. □

LEMMA 3.4. FORCE$_\chi(2)$ *is US-hard even if restricted to connected graphs.*

To prove the lemma, we describe a reduction from U3COL. Note that U3COL remains US-complete when restricted to connected graphs and that our reduction will preserve connectedness. Since the class of 2-colorable graphs is tractable and can be excluded from consideration, the desired reduction is given by the following lemma.

LEMMA 3.5. *Suppose that* $\chi(G) \geq 3$. *Then* $G \in$ U3COL *iff* $F_\chi(G \times K_3) = 2$.

Proof. Case 1: $G \in$ U3COL. We have to show that $F_\chi(G \times K_3) = 2$.

Fix arbitrary $u, v \in V(G)$ whose colors in the proper 3-coloring of G are different, for example, u and v can be any adjacent vertices of G. Let $V(K_3) = \{1, 2, 3\}$. Assign $p(u, 1) = 1$ and $p(v, 2) = 2$ and check that p forces a proper 3-coloring of $G \times K_3$. Assume that c is a proper 3-coloring of $G \times K_3$ consistent with p. Since c on each G-layer coincides with the 3-coloring of G up to permutation of colors, we easily infer that $c(v, 1) = c(u, 2) = 3$ (see Figure 4). This implies $c(u, 3) = 2$ and $c(v, 3) = 1$. Thus, in each G-layer we have two vertices with distinct colors, which determines colors of all the other vertices. As easily seen, the coloring obtained is really proper.

Case 2: $G \in$ 3COL $\setminus$ U3COL. We have to check that $F_\chi(G \times K_3) \geq 3$.

Given a partial coloring p of two vertices $a, b \in V(G \times K_3)$, we have to show that it is not forcing. The cases when $p(a) = p(b)$ or when a and b are in the same G- or K_3-layer are easy. Without loss of generality we therefore suppose that $p(a) = 1$, $p(b) = 2$, $a = (u, 1)$, and $b = (v, 2)$, where u and v are distinct vertices of G. Define two partial colorings of G by $c_1(u) = c_1(v) = 1$ and by $c_2(u) = 1$, $c_2(v) = 3$.

Subcase 2.1: *Both* c_1 *and* c_2 *extend to proper* 3*-colorings of* G. Denote the extensions by e_1 and e_2, respectively. Denote the three G-layers of $G \times K_3$ by G_1, G_2, G_3 and consider e_1, e_2 on G_1. Note that e_1 and e_2 each agree with p. We will show that e_1 and p have a common extension to a proper coloring of $G \times K_3$ and that the same is true for e_2 and p. Given a 3-coloring e of G, let e^+ (resp., e^-) denote the coloring obtained from e by renaming colors accordingly to the cyclic permutation (123) (resp., (132)). If e is proper, so are e^+ and e^-. Moreover, if we assign e, e^+, and e^- to G_1, G_2, and G_3 (in any of the six possible ways), we obtain a proper coloring of $G \times K_3$. Assign now e_1, e_1^+, e_1^- and e_2, e_2^-, e_2^+ to G_1, G_2, and G_3, for each triple in the order as written. We obtain two proper colorings of $G \times K_3$, both compatible with p. Thus, p is not forcing.

Subcase 2.2: *Only c_1 extends to a proper 3-coloring of G.* Since G is not uniquely colorable, there must be at least two extensions, e_1 and e_2, of c_1 to proper 3-colorings of G_1. As in the preceding case, e_1 and e_2 each agree with p and together with p extend to two distinct colorings of $G \times K_3$ (say, e_1, e_1^+, e_1^- and e_2, e_2^+, e_2^- on G_1, G_2, G_3, respectively).

Subcase 2.3: *Only c_2 extends to a proper coloring of G.* This case is completely similar to Subcase 2.2 (we have only to take e_1, e_1^-, e_1^+ and e_2, e_2^-, e_2^+).

Case 3: *$G \notin$ 3COL.* We have $\chi(G \times K_3) \geq 4$ by Lemma 2.1 and $F_\chi(G \times K_3) \geq 3$ by Lemma 2.7.1. ▯

Theorem 3.1 immediately follows from Lemmas 3.4 and 3.3.

4. Hardness of Force$_\chi$(2): A closer look.

Theorem 4.1. *The problem* Force$_\chi(2)$ *is coNP-hard even under the promises that $F_\chi(G) \leq 3$ and $\chi(G) \leq 3$ and even if an input graph G is given together with its proper 3-coloring.*

Let us for a while omit the promise that $F_\chi(G) \leq 3$. Then the theorem is provable by combining the Greenwell–Lovász reduction of coNP to US (Lemma 2.4) and our reduction of US to Force$_\chi(2)$ (Lemma 3.5). Doing so, we easily deduce the following:

- If $\chi(G) > 3$ and G is connected, then $G \cdot K_3$ is uniquely 3-colorable and hence $F_\chi((G \cdot K_3) \times K_3) = 2$.
- If $\chi(G) = 3$, then $G \cdot K_3$ is 3-chromatic because it contains an odd cycle (this is an easy particular case of the aforementioned Hedetniemi conjecture). Moreover, $G \cdot K_3$ has two induced 3-colorings, and hence $F_\chi((G \cdot K_3) \times K_3) \geq 3$.

To obtain Theorem 4.1 (without the promise $F_\chi(G) \leq 3$), it now suffices to make the following observation.

Lemma 4.2. $\chi((G \cdot K_3) \times K_3) = 3$ *for any graph G. Moreover, a proper 3-coloring is efficiently obtainable from the explicit product representation of $(G \cdot K_3) \times K_3$.*

Proof. By Lemma 2.2, we have $\chi(G \cdot K_3) \leq 3$ and hence, by Lemma 2.1, $\chi((G \cdot K_3) \times K_3) = 3$. Let $V(K_3) = \{1, 2, 3\}$ and denote $V_{i,j} = \{\, (v, i, j) : v \in V(G)\}$. It is not hard to see that $V_{1,1} \cup V_{2,2} \cup V_{3,3}$, $V_{1,2} \cup V_{2,3} \cup V_{3,1}$, and $V_{1,3} \cup V_{2,1} \cup V_{3,2}$ is a partition of $V((G \cdot K_3) \times K_3)$ into independent sets. ▯

Remark 4.3. The known facts about graph factorizations [32, Chapters 4 and 5] imply a nontrivial strengthening of Lemma 4.2 under certain, rather general conditions. Namely, if G is a connected nonbipartite graph and no two vertices of G have the same neighborhood, then we do not need to assume that $H = (G \cdot K_3) \times K_3$ is explicitly represented as a product graph because the product structure is efficiently reconstructible from the isomorphism type of H. We thank Wilfried Imrich for this observation.

To obtain the full version of Theorem 4.1, we only slightly modify the reduction: Before transforming G into $(G \cdot K_3) \times K_3$, we add to G a triangle with one vertex in $V(G)$ and two new vertices. Provided $\chi(G) \geq 3$, this does not change $\chi(G)$, and hence the modified transformation is an equally good reduction. The strengthening (the promise $F_\chi(G) \leq 3$) is given by the following lemma.

Lemma 4.4. *If a graph G is connected and contains a triangle, then $F_\chi((G \cdot K_3) \times K_3) \leq 3$.*

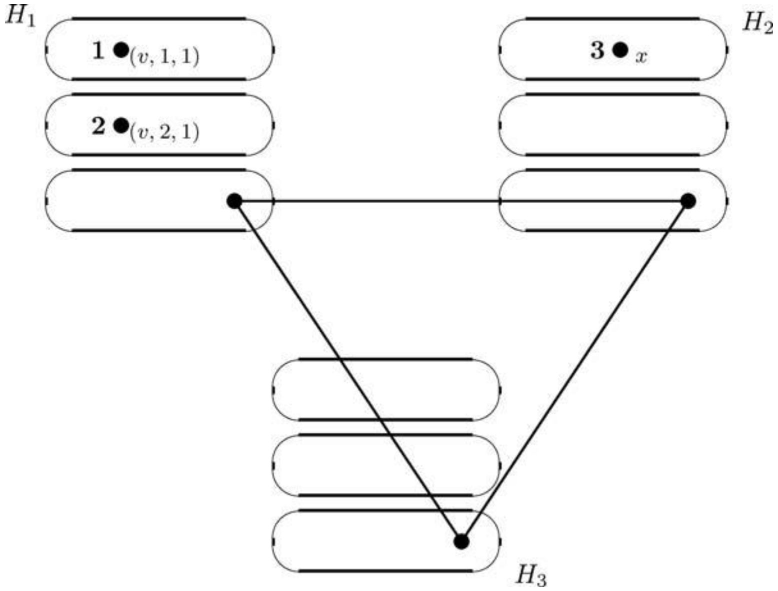


FIG. 5. $(G \cdot K_3) \times K_3$ *(proof of Lemma* 4.4*).*

Proof. Let v be a vertex of a triangle T in G. Consider the product $H = G \cdot K_3$ and a partial coloring $p(v,1) = 1$, $p(v,2) = 2$. We claim that p forces the K_3-induced coloring of H. Obviously, the latter is an extension of p. To show that no other extension is possible, assume that c is a proper 3-coloring of H compatible with p and consider the restriction of c to $T \cdot K_3$. By Lemma 2.6, c on $T \cdot K_3$ coincides with the coloring induced by the second factor. In particular, $c(v,3) = 3$. Our claim now follows from the connectedness of G by Lemma 2.5.

Now, let H_1, H_2, H_3 denote the three H-layers in $H \times K_3$ and, for each $i = 1,2,3$, let G_{i1}, G_{i2}, G_{i3} denote the three G-layers in H_i. Let $p(v,1,1) = 1$, $p(v,2,1) = 2$ be the forcing partial coloring for H_1 as described above (see Figure 5). Thus, p forces coloring the whole G_{1j} in color j for each $j = 1,2,3$. Suppose that c is a proper 3-coloring of $H \times K_3$ that agrees with p. From the product structure of $H \times K_3$ we see that, for each j, color j cannot occur in G_{2j}. Let T^2 denote the copy of $T \cdot K_3$ in H_2. By Lemma 2.6, c on T^2 is induced by the second factor, and hence $c(v,1,2)$, $c(v,2,2)$, and $c(v,3,2)$ are pairwise distinct. By Lemma 2.5, each of G_{21}, G_{22}, and G_{23} is monochromatic, and these layers receive pairwise distinct colors. We already know that $c(G_{2j}) \neq j$. Thus, we can choose an arbitrary $x \in G_{21}$ and define p for the third point by $p(x) = 3$. This forces $c(G_{21}) = 3$, $c(G_{22}) = 1$, and $c(G_{23}) = 2$. Since every vertex in G_{3j} is in a triangle with its clones in G_{1j} and G_{2j}, c is uniquely determined on H_3. □

The proof of Theorem 4.1 is complete.

5. Maximum size of a minimal forcing set. Another related invariant of a graph G is the largest cardinality of an inclusion-minimal forcing set in G. We will denote this number by $F^*_\chi(G)$. A complexity analysis of $F^*_\chi(G)$ is easier, owing to the characterization of uniquely colorable graphs obtained in [21].

LEMMA 5.1 (Hajiabolhassan et al. [21]). *A connected graph G is uniquely* 3-*colorable iff $F^*_\chi(G) = 2$.*

THEOREM 5.2. *The problem of deciding, given a graph G and its proper 3-coloring, whether or not $F^*_\chi(G) \le 2$ is coNP-complete.*

Proof. The problem is in coNP because a no-instance of it has a certificate consisting of a proper 3-coloring c of G and a 3-vertex set $A \subset V(G)$ such that no 2-element subset B of A is defining for c. The latter fact, for each B, is certified by another proper 3-coloring c_B that agrees with c on B but differs from c somewhere outside B.

The completeness is proved by reduction from the decision problem whether or not $\chi(G) > 3$. The latter problem is coNP-complete even if restricted to connected graphs with $\chi(G) \ge 3$. Let G be such a graph. Our reduction just transforms G into the categorical product $G \cdot K_3$. If $\chi(G) > 3$, then $G \cdot K_3$ is uniquely 3-colorable by Lemma 2.4 and, by Lemma 5.1, we have $F^*_\chi(G \cdot K_3) = 2$. If $\chi(G) = 3$, then $G \cdot K_3$ has at least two inequivalent proper 3-colorings, namely, those induced by the two factors. By Lemma 5.1, we have $F^*_\chi(G \cdot K_3) \ge 3$. □

6. General setting. In fact, many other graph characteristics also have natural forcing variants. Taking those into consideration, it will be convenient to use the formal concept of an NP optimization problem (see, e.g., [12]).

Let $\{0,1\}^*$ denote the set of binary strings. The length of a string $w \in \{0,1\}^*$ is denoted by $|w|$. We will use notation $[n] = \{1, 2, \ldots, n\}$.

An *NP optimization problem* $\pi = (\mathrm{opt}_\pi, I_\pi, \mathit{sol}_\pi, \mathrm{v}_\pi)$ (where subscript π may be omitted) consists of the following components:

- $\mathrm{opt} \in \{\max, \min\}$ is a *type* of the problem.
- $I \subseteq \{0,1\}^*$ is the polynomial-time decidable set of *instances* of π.
- Given $x \in I$, we have $\mathit{sol}(x) \subset \{0,1\}^*$, the set of *feasible solutions* of π on instance x. We suppose that all $y \in \mathit{sol}(x)$ have the same length and that length depends only on $|x|$ and is bounded by $|x|^{O(1)}$. Given x and y, it is decidable in polynomial time whether $y \in \mathit{sol}(x)$.
- $\mathrm{v} : \{0,1\}^* \times \{0,1\}^* \to \mathbf{N}$ is a polynomial-time computable *objective function* taking on positive integer values. If $y \in \mathit{sol}(x)$, then $\mathrm{v}(x, y)$ is called the *value* of y.

The problem is, given an instance x, to compute the optimum value

$$\pi(x) = \mathrm{opt}_{y \in \mathit{sol}(x)} \mathrm{v}(x, y).$$

Such a problem is called *polynomially bounded* if $\mathrm{v}(x, y) = |x|^{O(1)}$ for all $x \in I$ and $y \in \mathit{sol}(x)$.

Any $y \in \mathit{sol}(x)$ whose value is optimal is called an *optimum solution* of π on instance x. Let $\mathit{optsol}(x)$ denote the set of all such y. Given an NP optimization problem π, we define

$$\mathrm{UO}_\pi = \{\, x : |\,\mathit{optsol}(x)| = 1 \,\}.$$

Example 6.1. The problem of computing the chromatic number of a graph is expressible as a quadruple $\chi = (\min, I, \mathit{sol}, \mathrm{v})$ as follows. A graph G with vertex set $V(G) = \{v_1, \ldots, v_n\}$ is represented by its adjacency matrix written down row after row as a binary string x of length n^2. A feasible solution, that is a proper coloring $c : V(G) \to [n]$, is represented by a binary string $y = c(v_1) \ldots c(v_n)$ of length n^2, where a color i is encoded by string $0^{i-1}10^{n-i}$. The value $\mathrm{v}(x, y)$ is equal to the actual number of colors occurring in y.

Example 6.2. For the problem of computing the clique number it is natural to fix the following representation. An instance graph G is encoded as above. A feasible solution, which is a subset of $V(G)$, is encoded by its characteristic binary string of length n. The problem of computing the domination number is represented in the same way.

Given a nonempty set $U \subseteq \{0,1\}^l$, we define *force*(U) to be the minimum cardinality of a set $S \subseteq [l]$ such that there is exactly one string in U with 1 at every position from S. With each NP optimization problem π we associate its *forcing number* F_π, an integer-valued function of instances of π defined by

$$F_\pi(x) = \mathit{force}(\mathit{optsol}(x)).$$

Let $\textsc{Force}_\pi(k) = \{\, x : F_\pi(x) \le k\}$. It is easy to check that, if χ, ω, and γ are represented as in Examples 6.1 and 6.2, then F_χ, F_ω, and F_γ are precisely those graph invariants introduced in section 1.

Note that *force*$(U) = 0$ iff U is a singleton. It follows that for $\pi \in \{\omega, \gamma\}$ we have

$$x \in \mathrm{UO}_\pi \quad \text{iff} \quad F_\pi(x) = 0. \tag{2}$$

This will be the starting point of our analysis of decision problems $\textsc{Force}_\omega(k)$ and $\textsc{Force}_\gamma(k)$ in the next section.

7. Hardness of $\textsc{Force}_\omega(k)$ and $\textsc{Force}_\gamma(k)$. The results stated here are based on known reducibilities between several optimization problems related to our work. Since this material is dispersed through the literature with proofs sometimes skipped, for the reader's convenience we outline some details. We first introduce some reducibility concepts for NP optimization problems.

Let π and ϖ be NP optimization problems of the same type. Let $f : \{0,1\}^* \to \{0,1\}^*$ and $g : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ be polynomial-time computable functions such that for every $x \in I_\pi$ we have $f(x) \in I_\varpi$ and for every $y \in \mathit{sol}_\varpi(f(x))$ we have $g(x,y) \in \mathit{sol}_\pi(x)$. Such a pair (f,g) is said to be an *S-reduction* from π to ϖ if for every $x \in I_\pi$ we have

$$\mathrm{opt}_\pi(x) = \mathrm{opt}_\varpi(f(x))$$

and, in addition, for every $y \in \mathit{sol}_\varpi(f(x))$ we have

$$\mathrm{v}_\pi(x, g(x,y)) = \mathrm{v}_\varpi(f(x), y).$$

We call an S-reduction (f,g) a *parsimonious reduction* from π to ϖ if, for any $x \in I_\pi$, $g(x,\cdot)$ is a one-to-one map from $\mathit{sol}_\varpi(f(x))$ onto $\mathit{sol}_\pi(x)$. If only a weaker condition is met, namely, that $g(x,\cdot)$ is a one-to-one correspondence between the optimum solutions of ϖ on instance $f(x)$ and the optimum solutions of π on instance x, then (f,g) will be called a *weakly parsimonious reduction* from π to ϖ.

Given a Boolean formula Φ in conjunctive normal form (CNF), let $\sigma(\Phi)$ denote the maximum number of clauses of Φ that can be satisfied by any one truth assignment. By σ_3 we denote the restriction of σ to 3CNF formulas (those having at most three literals per clause). We regard σ and σ_3 as NP optimization problems. Both problems belong to the class MAX NP introduced by Papadimitriou and Yannakakis [41]. Crescenzi, Fiorini, and Silvestri [13], who introduced the notion of an S-reduction, proved that every problem in MAX NP is S-reducible to ω, the maximum clique problem. We need a somewhat stronger fact about σ_3.

Lemma 7.1. *There is a parsimonious reduction from σ_3 to ω.*

Proof. Let ϕ be a disjunctive clause, and X be the set of variables occurring in ϕ. Let $D(\phi)$ denote the set of all conjunctions that contain every variable from X or its negation and imply ϕ. Note that ϕ is logically equivalent to the disjunction of all ψ in $D(\phi)$.

Given a 3CNF formula Φ, we construct a graph G as follows. Let $V(G)$ be the union of $D(\phi)$ over all clauses ϕ of Φ. We join ψ_1 and ψ_2 in $V(G)$ by an edge if these conjunctions are consistent, i.e., no variable occurring in ψ_1 occurs in ψ_2 with negation and vice versa. ∎

Lemma 7.2 (Thierauf [46, section 3.2.3]). *The decision problem* UO_ω *is US-hard.*

Proof. Denote the restrictions of SAT and USAT to 3CNF formulas by 3SAT and U3SAT, respectively. Since there is a parsimonious $\le_{\mathrm{m}}^{\mathrm{P}}$-reduction from SAT to 3SAT (see, e.g., [40]), U3SAT is US-complete. We now show that U3SAT $\le_{\mathrm{m}}^{\mathrm{P}} \mathrm{UO}_\omega$.

Given a 3CNF formula Φ, let G be the graph constructed from Φ by the reduction of Lemma 7.1. Let m denote the number of clauses in Φ and $H = G + 2K_{m-1}$, the disjoint union of G and two copies of K_{m-1}.

If $\Phi \in$ U3SAT, then $\omega(H) = \omega(G) = m$ and $H \in \mathrm{UO}_\omega$ because $G \in \mathrm{UO}_\omega$.

If $\Phi \in$ SAT $\setminus$ U3SAT, then $\omega(H) = \omega(G) = m$ and $H \notin \mathrm{UO}_\omega$ because $G \notin \mathrm{UO}_\omega$.

If $\Phi \notin$ SAT, then $\omega(G) \le m-1$, $\omega(H) = m-1$, and $H \notin \mathrm{UO}_\omega$ having at least two optimal cliques.

Thus, $\Phi \in$ U3SAT iff $H \in \mathrm{UO}_\omega$. ∎

Lemma 7.3. $\mathrm{UO}_\omega \le_m^P \mathrm{UO}_\gamma$.

Proof. Recall that a *vertex cover* of a graph G is a set $S \subseteq V(G)$ such that every edge of G is incident to a vertex in S. The *vertex cover number* of G is defined to be the minimum cardinality of a vertex cover of G and denoted by $\tau(G)$. It is easy to see and well known that $S \subseteq V(G)$ is a clique in G iff $V(G) \setminus S$ is a vertex cover in the graph complementary to G. It follows that

$$\mathrm{UO}_\omega \le_{\mathrm{m}}^{\mathrm{P}} \mathrm{UO}_\tau. \tag{3}$$

We now show that

$$\mathrm{UO}_\tau \le_{\mathrm{m}}^{\mathrm{P}} \mathrm{UO}_\gamma \tag{4}$$

by regarding τ and γ as NP minimization problems and designing a weakly parsimonious reduction from τ to γ. Recall that, given a set X and a system of its subsets $\mathcal{Y} = \{Y_1, \ldots, Y_n\}$, a subsystem $\{Y_{i_1}, \ldots, Y_{i_k}\}$ is called a *set cover* if $X = \bigcup_{j=1}^{k} Y_{i_j}$. We compose two known reductions between minimization problems, Reduction A from the minimum vertex cover to the minimum set cover [2, Theorem 10.11] and Reduction B from the minimum set cover to the minimum domination number (an adaptation of [33, Theorem A.1]).

Reduction A. Given a graph G and its vertex v, let $I(v)$ denote the set of the edges of G incident to v. Consider the set $X = E(G)$ and the system of its subsets $\mathcal{Y} = \{\, I(v) : v \in V(G)\}$. Then $S \subseteq V(G)$ is an optimal vertex cover for G iff $\{\, I(v) : v \in S\}$ is an optimal set cover for $(X, \mathcal{Y})$.

Reduction B. Given a set $X = \{x_1, \ldots, x_m\}$ and a system of sets $\mathcal{Y} = \{Y_1, \ldots, Y_n\}$ such that $X = \bigcup_{j=1}^{n} Y_j$, we construct a graph H as follows. $V(H)$ contains each element x_i in duplicate, namely, x_i itself and its clone x_i'. There is no edge between these $2m$ vertices. Other vertices of H are indices $1, \ldots, n$, with all possible $\binom{n}{2}$ edges between them. Iff $x_i \in Y_j$, both x_i and x_i' are adjacent to j. There are no more vertices and edges.

Observe that any optimal dominating set $D \subset V(H)$ is included in $[n]$. Indeed, if D contains both x_i and x_i', then it can be reduced by replacing these two vertices by only one vertex j such that $x_i \in Y_j$. If D contains exactly one of x_i and x'_i, say x_i, then it should contain some j such that $x_i \in Y_j$ to dominate x'_i. But then D can be reduced just by removing x_i.

It is also clear that a set $D \subseteq [n]$ is dominating in H iff $\{\, Y_j : j \in D\}$ is a set cover for $(X, \mathcal{Y})$. Thus, there is a one-to-one correspondence between optimal set covers for $(X, \mathcal{Y})$ and optimal dominating sets in H. □

Thus, UO_ω and UO_γ are both US-hard.

LEMMA 7.4. *Let* $\pi \in \{\omega, \gamma\}$. *Then* $\textsc{Force}_\pi(k) \le^P_m \textsc{Force}_\pi(k+1)$ *for any* $k \ge 0$.

Proof. Given a graph G, we have to construct a graph H such that $F_\pi(G) \le k$ iff $F_\pi(H) \le k+1$. It suffices to ensure that

$$F_\pi(H) = F_\pi(G) + 1. \tag{5}$$

Let $\pi = \omega$. Let H be the result of adding to G two new vertices u and v and the edges $\{w, u\}$ and $\{w, v\}$ for all $w \in V(G)$. Any optimal clique in H consists of an optimal clique in G and of either u or v. Hence any forcing set in H consists of a forcing set in G and of either u or v (we use the terminology of section 1). This implies (5).

If $\pi = \gamma$, we obtain H from G by adding a new isolated edge. □

Putting it all together, we make the following conclusion.

THEOREM 7.5. *Let* $\pi \in \{\omega, \gamma\}$. *Then*

$$\mathrm{US} \le^P_m \mathrm{UO}_\pi = \textsc{Force}_\pi(0) \le^P_m \textsc{Force}_\pi(k) \le^P_m \textsc{Force}_\pi(k+1)$$

for any $k \ge 0$.

8. Complexity of $\textsc{Force}_\pi(k)$: An upper bound. We first state a simple general property of the class US.

LEMMA 8.1. *Every US-complete set has an* AND_2 *function.*[2]

Proof. It suffices to prove the lemma for any particular US-complete set, for example, USAT. Given two Boolean formulas Φ and Ψ, rename the variables in Ψ so that the formulas are over disjoint sets of variables, and consider the conjunction $\Phi \wedge \Psi$. The conjunction is in USAT iff both Φ and Ψ are in USAT. □

In section 6 with a nonempty set $U \subseteq \{0,1\}^l$ we associated the number $\mathit{force}(U)$. Additionally, let us set $\mathit{force}(\emptyset) = \infty$.

THEOREM 8.2. *Let* π *be a polynomially bounded NP optimization problem. Then* $\textsc{Force}_\pi(k) \le^P_{dtt} \mathrm{US}$ *for each* $k \ge 0$.

Proof. We will assume that π is a minimization problem (the case of maximization problems is quite similar). Suppose that $v(x, y) \le |x|^c$ for a constant c. Given $1 \le m \le |x|^c$, we define

$$\mathit{sol}^m(x) = \{\, y \in \mathit{sol}(x) : v(x, y) = m\}$$

and

$$F^m_\pi(x) = \mathit{force}(\mathit{sol}^m(x)).$$

In particular, $F^m_\pi(x) = F_\pi(x)$ if $m = \pi(x)$.

Let k be a fixed integer. Notice that

$$F_\pi(x) \le k \ \text{ iff } \bigvee_{m=1}^{|x|^c} (F^m_\pi(x) \le k \wedge \pi(x) \ge m) \tag{6}$$

[2] In fact, a stronger fact is true: Every US-complete set has an AND function of unbounded arity.

(actually, only the clause for which $m = \pi(x)$ can be true). The set of pairs (x, m) with $\pi(x) \geq m$ is in coNP and hence in US. Let us now show that the set of (x, m) with $F^m_\pi(x) \leq k$ is dtt-reducible to US.

Recall that $\mathit{sol}(x) \subseteq \{0,1\}^{l(x)}$, where $l(x) \leq |x|^d$ for a constant d. Define T to be the set of quadruples (x, m, l, D) such that m and l are positive integers, $D \subseteq [l]$, and there is a unique $y \in \mathit{sol}^m(x)$ of length l with all 1's in positions from D. It is easy to see that T is in US and

$$F^m_\pi(x) \leq k \quad \text{iff} \quad \bigvee_{\substack{l,D\colon l \leq |x|^d,\\ D\subseteq[l],\, |D|\leq k}} (x, m, l, D) \in T.$$

Combining this equivalence with (6), we conclude that $F_\pi(x) \leq k$ iff there are numbers $m \leq |x|^c$ and $l \leq |x|^d$ and a set $D \subseteq [l]$ of size at most k such that

$$(x, m, l, D) \in T \;\wedge\; \pi(x) \geq m.$$

By Lemma 8.1, this conjunction is expressible as a proposition about membership of the quadruple (x, m, l, D) in a US-complete set. Thus, the condition $F_\pi(x) \leq k$ is equivalent to a disjunction of less than $|x|^{c+d(k+1)}$ propositions, each verifiable in US. □

Corollary 8.3. *Let* $\pi \in \{\chi, \omega, \gamma\}$. *Then* $\textsc{Force}_\pi(k) \leq^{P}_{dtt} \mathrm{US}$ *for each* $k \geq 0$.

Remark 8.4. Using (3) and Theorem 8.2, we can easily show that for the vertex cover number τ we also have

$$\mathrm{US} \leq^{\mathrm{P}}_{\mathrm{m}} \mathrm{UO}_\tau = \textsc{Force}_\tau(0) \leq^{\mathrm{P}}_{\mathrm{m}} \textsc{Force}_\tau(k) \leq^{\mathrm{P}}_{\mathrm{m}} \textsc{Force}_\tau(k+1) \leq^{\mathrm{P}}_{\mathrm{dtt}} \mathrm{US}.$$

9. Concluding discussion and open questions.

1. We have considered forcing versions of the three most popular graph invariants: the chromatic, the clique, and the domination numbers (F_χ, F_ω, and F_γ respectively). We have shown that the slice decision problems for each of F_χ, F_ω, and F_γ are as hard as US under many-one reducibility and as easy as US under dtt-reducibility. The latter upper bound is actually true for the forcing variant of any polynomially bounded NP optimization problem. The lower bound in the case of F_ω and F_γ is provable by using standard reductions due to a close connection with the unique optimum problems UO_ω and UO_γ. However, in the case of F_χ we use somewhat more elaborate reductions involving graph products. We point out two simple reasons for the distinction between F_χ and F_ω, F_γ. First, unlike the case of ω and γ, the unique colorability of a graph is not a function of F_χ (cf. Lemma 2.7.3). Second, we currently do not know any reductions between F_χ, F_ω, and F_γ as optimization problems that would allow us to relate their complexities (cf. further discussion).

2. We have shown that the slice decision problems $\textsc{Force}_\pi(k)$ for $\pi \in \{\chi, \omega, \gamma\}$ are close to each other in the complexity hierarchy. Furthermore, let $\textsc{Force}_\pi(*) = \{\, (x, k) : F_\pi(x) \leq k\}$. Hatami [27] has recently shown that, like $\textsc{Force}_\chi(*)$, the decision problem $\textsc{Force}_\omega(*)$ is Σ^{P}_2-complete. Consequently, the problems of computing F_χ and F_ω are polynomial-time Turing equivalent. It would be also interesting to compare the complexities of F_χ, F_ω, and F_γ using weaker reducibility concepts for optimization problems. (As is well known, the similarity of decision versions does not imply the similarity of the underlying optimization problems; for example, the decision versions of χ, ω, and γ are all NP-complete and parsimoniously equivalent but have pairwise different parameterized complexities, and the corresponding optimization problems have pairwise different approximation complexities.)

3. Is $\textsc{Force}_\pi(k)$ NP-hard under $\le_{\mathrm{m}}^{\mathrm{P}}$-reductions for any π under consideration and constant k? It should be noted that the affirmative answer would settle a long-standing open problem if $\mathrm{C}_=\mathrm{P}$ contains NP in the affirmative.

4. Let UCOL be the set of all uniquely colorable graphs (with no restriction on the chromatic number). Is it true that UCOL $\le_{\mathrm{m}}^{\mathrm{P}}$ $\textsc{Force}_\chi(2)$? It is not hard to show that UCOL is US-hard.

Acknowledgments. This work was initiated when the authors met at the Department of Information Systems at Vienna University of Technology. They are indebted to Georg Gottlob for his hospitality. Also, thanks to Hamed Hatami for useful discussions, and to the anonymous referees for their comments.

In Memoriam. The journal version of this paper has been prepared without Frank Harary. He passed away a month before the conference presentation at STACS'05 [26]. Frank's enthusiasm and inspiration were fundamental to this work.

W.S., O.V.

REFERENCES

[1] P. Adams, M. Mahdian, and E. S. Mahmoodian, *On the forced matching numbers of bipartite graphs*, Discrete Math., 281 (2004), pp. 1–12.

[2] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison–Wesley, Reading, MA, 1976.

[3] P. Afshani, H. Hatami, and E. S. Mahmoodian, *On the spectrum of the forced matching number of graphs*, Austral. J. Combin., 30 (2004), pp. 147–160.

[4] A. Blass and Y. Gurevich, *On the unique satisfiability problem*, Inform. Control, 55 (1982), pp. 80–88.

[5] R. Chang, J. Kadin, and P. Rohatgi, *On unique satisfiability and the threshold behaviour of randomized reductions*, J. Comput. System Sci., 50 (1995), pp. 359–373.

[6] G. Chartrand, H. Gavlas, R. C. Vandell, and F. Harary, *The forcing domination number of a graph*, J. Combin. Math. Combin. Comput., 25 (1997), pp. 161–174.

[7] G. Chartrand, F. Harary, M. Schultz, and C. E. Wall, *Forced orientation numbers of a graph*, Congr. Numer., 100 (1994), pp. 183–191.

[8] G. Chartrand and P. Zhang, *The forcing geodetic number of a graph*, Discuss. Math. Graph Theory, 19 (1999), pp. 45–58.

[9] G. Chartrand and P. Zhang, *The forcing hull number of a graph*, J. Combin. Math. Combin. Comput., 38 (2001), pp. 81–94.

[10] G. Chartrand and P. Zhang, *The forcing dimension of a graph*, Math. Bohem., 126 (2001), pp. 711–720.

[11] C. J. Colbourn, M. J. Colbourn, and D. R. Stinson, *The computational complexity of recognizing critical sets*, in Graph Theory, Proceedings of the 1st Southeast Asian Colloquium, Singapore, 1983, Lecture Notes in Math. 1073, Springer, New York, 1984, pp. 248–253.

[12] P. Crescenzi, *A short guide to approximation preserving reductions*, in Proceedings of the 12th Annual Conference on Computational Complexity, Ulm, Germany, IEEE Computer Society Press, Los Alamitos, CA, 1997, pp. 262–272.

[13] P. Crescenzi, C. Fiorini, and R. Silvestri, *A note on approximation of the MAX CLIQUE problem*, Inform. Process. Lett., 40 (1991), pp. 1–5.

[14] D. Curran and G. H. J. van Rees, *Critical sets in latin squares*, Congr. Numer., 22 (1979), pp. 165–168.

[15] D. P. Dailey, *Uniqueness of colorability and colorability of planar 4-regular graphs are NP-complete*, Discrete Math., 30 (1980), pp. 289–293.

[16] T. Easton and R. Gary Parker, *On completing Latin squares*, Discrete Appl. Math., 113 (2001), pp. 167–181.

[17] M. El-Zahar and N. Sauer, *The chromatic number of the product of two 4-chromatic graphs is 4*, Combinatorica, 5 (1985), pp. 121–126.

[18] M. R. Garey, D. S. Johnson, and L. J. Stockmeyer, *Some simplified NP-complete graph problems*, Theoret. Comput. Sci., 1 (1976), pp. 237–267.

[19] K. Gray, *On the minimum number of blocks defining a design*, Bull. Austral. Math. Soc., 41 (1990), pp. 97–112.

[20] D. L. Greenwell and L. Lovász, *Applications of product colouring*, Acta Math. Acad. Sci. Hung., 25 (1974), pp. 335–340.

[21] H. Hajiabolhassan, M. L. Mehrabadi, R. Tusserkani, and M. Zaker, *A characterization of uniquely vertex colorable graphs using minimal defining sets*, Discrete Math., 199 (1999), pp. 233–236.

[22] F. Harary, *Three new directions in graph theory*, in Proceedings of the First Estonian Conference on Graphs and Applications, Tartu-Kääriku, 1991, Tartu University Press, Tartu, Estonia, 1993, pp. 15–19.

[23] F. Harary, S. T. Hedetniemi, and R. W. Robinson, *Uniquely colorable graphs*, J. Combin. Theory, 6 (1969), pp. 264–270.

[24] F. Harary, S. T. Hedetniemi, and R. W. Robinson, *Errata: Uniquely colorable graphs*, J. Combin. Theory, 6 (1970), pp. 221.

[25] F. Harary, D. Klein, and T. Živković, *Graphical properties of polyhexes: Perfect matching vector and forcing*, J. Math. Chem., 6 (1991), pp. 295–306.

[26] F. Harary, W. Slany, and O. Verbitsky, *On the computational complexity of the forcing chromatic number*, in Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science, V. Diekert, B. Durand, eds., Lecture Notes in Comput. Sci. 3404, Springer-Verlag, Berlin, 2005, pp. 182–193.

[27] H. Hatami, Department of Computer Science, University of Toronto, *personal communication*, 2005.

[28] H. Hatami and H. Maserrat, *On the computational complexity of defining sets*, Discrete Appl. Math., 149 (2005), pp. 101–110.

[29] L. Hemachandra, *Structure of complexity classes: Separations, collapses, and completeness*, in Proceedings of the 13th MFCS Conference, Lecture Notes in Comput. Sci. 324, Springer, New York, 1988, pp. 59–73.

[30] L. A. Hemaspaandra and M. Ogihara, *The Complexity Theory Companion*, Springer, Berlin, 2002.

[31] P. Horák, R. E. L. Aldred, and H. Fleischner, *Completing Latin squares: Critical sets*, J. Combin. Designs, 10 (2002), pp. 419–432.

[32] W. Imrich and S. Klavzar, *Product graphs. Structure and Recognition*, Wiley-Intersci. Ser. Discrete Math. Optim., Wiley, Chichester, UK, 2000.

[33] V. Kann, *On the Approximability of NP-complete Optimization Problems*, Ph.D. thesis, Royal Institute of Technology, Stockholm, Sweden, 1992.

[34] J. Köbler, U. Schöning, and J. Torán, *The Graph Isomorphism Problem: Its Structural Complexity*, Birkhäuser, Boston, Cambridge, MA, 1993.

[35] C. Lund and M. Yannakakis, *On the hardness of approximating minimization problems*, J. ACM, 41 (1994), pp. 960–981.

[36] M. Mahdian, E. S. Mahmoodian, R. Naserasr, and F. Harary, *On defining sets of vertex colorings of the Cartesian product of a cycle with a complete graph*, in Combinatorics, Graph Theory, and Algorithms, Y. Alavi, D. R. Lick, and A. Schwenk, eds., New Issues Press, Kalamazoo, MI, 1999, pp. 461–467.

[37] E. S. Mahmoodian, R. Naserasr, and M. Zaker, *Defining sets in vertex colorings of graphs and Latin rectangles*, Discrete Math., 167–168 (1997), pp. 451–460.

[38] J. Nelder, *Critical sets in Latin squares*, CSIRO Division of Math. and Stats, Newsletter, 38 (1977), pp. 4.

[39] C. Papadimitriou, *On the complexity of unique solutions*, J. ACM, 31 (1984), pp. 392–400.

[40] C. Papadimitriou, *Computational Complexity*, Addison–Wesley, Reading, MA, 1994.

[41] C. Papadimitriou and M. Yannakakis, *Optimization, approximation, and complexity classes*, J. Comput. System Sci., 43 (1991), pp. 425–440.

[42] D. Ranjan, S. Chari, and P. Rohatgi, *Improving known solutions is hard*, Comput. Complexity, 3 (1993), pp. 168–185.

[43] G. Sabidussi, *Graphs with given group and given graph-theoretical properties*, Canad. J. Math., 9 (1957), pp. 515–525.

[44] A. L. Selman, *Promise problems and complexity classes*, Inform. and Comput., 78 (1988), pp. 87–98.

[45] L. J. Stockmeyer, *Planar* 3*-colorability is NP-complete*, SIGACT News, 5 (1973), pp. 19–25.

[46] T. Thierauf, *The Computational Complexity of Equivalence and Isomorphism Problems*, Lecture Notes in Comput. Sci. 1852, Springer, New York, 2000.

[47] L. G. Valiant and V. V. Vazirani, *NP is as easy as detecting unique solutions*, Theoret. Comput. Sci., 47 (1986), pp. 287–300.

[48] D. Zuckerman, *On unapproximable versions of NP-complete problems*, SIAM J. Comput., 25 (1996), pp. 1293–1304.

SIAM J. COMPUT.
Vol. 37, No. 1, pp. 20–46

LOWER BOUNDS FOR QUANTUM COMMUNICATION COMPLEXITY*

HARTMUT KLAUCK†

Abstract. We prove lower bounds on the bounded error quantum communication complexity. Our methods are based on the Fourier transform of the considered functions. First we generalize a method for proving classical communication complexity lower bounds developed by Raz [*Comput. Complexity,* 5 (1995), pp. 205–221] to the quantum case. Applying this method, we give an exponential separation between bounded error quantum communication complexity and nondeterministic quantum communication complexity. We develop several other lower bound methods based on the Fourier transform, notably showing that $\sqrt{\bar{s}(f)/\log n}$, for the average sensitivity $\bar{s}(f)$ of a function f, yields a lower bound on the bounded error quantum communication complexity of $f((x \wedge y) \oplus z)$, where x is a Boolean word held by Alice and y, z are Boolean words held by Bob. We then prove the first large lower bounds on the bounded error quantum communication complexity of functions, for which a polynomial quantum speedup is possible. For all the functions we investigate, the only previously applied general lower bound method based on discrepancy yields bounds that are $O(\log n)$.

Key words. communication complexity, quantum computing, lower bounds, computational complexity

AMS subject classifications. 68Q17, 68Q10, 81P68, 03D15

DOI. 10.1137/S0097539702405620

1. Introduction.

Quantum mechanical computing and communication have been studied extensively during the last decade. Communication has to be a physical process, so an investigation of the properties of physically allowed communication is desirable, and the fundamental theory of physics available to us for this investigation is quantum mechanics.

The theory of communication complexity deals with the question of how efficiently communication problems can be solved, and it has various applications to lower bound proofs for other resources (an introduction to (classical) communication complexity can be found in Kushilevitz and Nisan's excellent monograph [29]).

In a quantum protocol (as defined by Yao [41]) two players Alice and Bob receive an input and have to cooperatively compute some function defined on the pair of inputs. To this end they exchange messages consisting of qubits until the result can be produced from some measurement done by one of the players (for surveys about quantum communication complexity see [39, 10, 25]).

It is known that quantum communication protocols can sometimes be substantially more efficient than classical probabilistic protocols: the most prominent example of such a function is the disjointness problem $DISJ_n$, in which the players receive incidence vectors x, y of subsets of $\{1, \ldots, n\}$, and have to decide whether or not the sets are disjoint: $\neg \bigvee (x_i \wedge y_i)$. By an application of Grover's search algorithm [19] to communication complexity given by Buhrman, Cleve, and Wigderson [11], an upper

*Received by the editors April 15, 2002; accepted for publication (in revised form) August 16, 2006; published electronically March 30, 2007. Most results in this paper have previously appeared in its conference version [26] at FOCS'01.

http://www.siam.org/journals/sicomp/37-1/40562.html

†Institut für Informatik, Goethe-Universität Frankfurt, 60054 Frankfurt am Main, Germany (hklauck@gmail.com). Supported by DFG grant KL1470/1. This work was partially done at CWI, The Netherlands, by supported by the EU 5th framework program QAIP IST-1999-11234 and by NWO grant 612.055.001.

bound of $O(\sqrt{n}\log n)$ holds for the bounded error quantum communication complexity of $DISJ_n$. This upper bound has been improved to $O(\sqrt{n}c^{\log^* n})$ for a constant c by Høyer and de Wolf [22] and, finally, to $O(\sqrt{n})$ by Aaronson and Ambainis [1]. The classical bounded error communication complexity of $DISJ_n$, on the other hand, is $\Omega(n)$ by a bound due to Kalyanasundaram and Schnitger [24]. The quantum protocol for $DISJ_n$ yields the largest gap between quantum and classical communication complexities known so far for a total function. For partial functions and so-called sampling problems, even exponential gaps between quantum and classical communication complexities are known; see [36, 11, 3].

Unfortunately only a few lower bound methods for quantum communication complexity are known: the logarithm of the rank of the communication matrix is known as a lower bound for exact (i.e., errorless) quantum communication [11, 12]; the (often weakly applied) discrepancy method can be used to give lower bounds for protocols with error as shown by Kremer [28]. Buhrman and de Wolf [12] observed that lower bounds on the minimum rank of matrices approximating the communication matrix give bounded error quantum lower bounds, but were not able to apply this method to any explicit function.[1] In this paper we introduce several lower bound methods for bounded error quantum communication complexity exploiting algebraic properties of the communication matrix.

Let IP_n denote the inner product modulo 2 function, i.e.,

$$IP_n(x, y) = \bigoplus_{i=1}^{n} (x_i \wedge y_i).$$

Known results about the discrepancy of the inner product function under the uniform distribution then imply that quantum protocols for IP_n with error $1/2-\epsilon$ have complexity $\Omega(n/2 - \log(1/\epsilon))$; see [28] (actually, only a linear lower bound assuming constant error is proved there, but minor modifications give the stated result). The inner product function appears to be the only explicit function for which a large lower bound on the bounded error quantum communication complexity has been published prior to this paper.

We prove new lower bounds on the bounded error quantum communication complexity of several functions. These bounds are exponentially bigger than the bounds obtainable by the discrepancy method. Note that we do not consider the model of quantum communication with prior entanglement here (as defined by Cleve and Buhrman [13]).

Our results are as follows. First we generalize a lower bound method developed by Raz [35] for classical bounded error protocols to the quantum case. The lower bound is given in terms of the sum of absolute values of selected Fourier coefficients of the function. To be able to generalize this method we have to decompose the quantum protocol into a "small" set of weighted monochromatic rectangles, so that the sum of these approximates the communication matrix. Opposed to the classical case the weights may be negative, but all weights have absolute value at most 1.

Applying the method, we get a lower bound of $\Omega(n/\log n)$ for the bounded error

[1] A result by Razborov [37] (published subsequently to the conference version of the present paper) implies such lower bounds for a limited class of functions and gives a tight characterization of the bounded error quantum communication complexity of functions $f(x, y) = g(x \wedge y)$, also settling the complexity of the disjointness problem to $\Theta(\sqrt{n})$.

quantum communication complexity of the Boolean function $HAM_n^{n/2}$, where

$$HAM_n^t(x,y) = 1 \iff dist(x,y) \neq t \iff \sum_i (x_i \oplus y_i) \neq t,$$

for binary strings x, y of length n and the Hamming distance *dist*. We then show, using methods of de Wolf [40], that the nondeterministic (i.e., one-sided unbounded error) quantum communication complexity of $HAM_n^{n/2}$ is $O(\log n)$. So we get an exponential gap between the nondeterministic quantum and bounded error quantum complexities. Since it is also known that the equality function EQ_n has (classical) bounded error protocols with $O(\log n)$ communication [29], while its nondeterministic quantum communication complexity is $\Theta(n)$ [40], we get the following separation.

Let BQP denote the bounded error quantum communication complexity and NQP denote the nondeterministic quantum communication complexity (see section 2.2 for definitions).

Corollary 1.1. *There are total Boolean functions $HAM_n^{n/2}, EQ_n$ on $2n$ inputs each, such that*

1. $NQC(HAM_n^{n/2}) = O(\log n)$ *and* $BQC(HAM_n^{n/2}) = \Omega(n/\log n)$,
2. $BQC(EQ_n) = O(\log n)$ *and* $NQC(EQ_n) = \Omega(n)$.

Furthermore we give quite tight lower and upper bounds for HAM_n^t for general values of t, establishing that bounded error quantum communication does not give a significant speedup compared to classical bounded error communication for these problems. The same bounds hold for testing whether the Hamming distance is at most t.

Corollary 1.2. *Let $t : \mathbb{N} \to \mathbb{N}$ be any monotone increasing function with $t(n) \leq n/2$. Then*

1. $BQC(HAM_n^{t(n)}) \geq \Omega\left(\frac{t(n)}{\log t(n)} + \log n\right)$.
2. $BPC(HAM_n^{t(n)}) = O(t(n) \log n)$.

We then turn to several other techniques for proving lower bounds, which are also based on the Fourier transform. We concentrate on functions $f(x,y) = g(x \diamond y)$, for $\diamond \in \{\wedge, \oplus\}$, which are the bitwise conjunction and parity operators. We prove that for $\diamond = \wedge$, if we choose any Fourier coefficient $\hat{g}_z$ of g, then $|z|/(1 - \log|\hat{g}_z|)$ yields a lower bound on the bounded error quantum communication complexity of f. Averaging over all coefficients leads to a bound given by the average sensitivity of g divided by the entropy of the squared Fourier coefficients. We then show another bound for $\diamond = \oplus$ in terms of the entropy of the Fourier coefficients and obtain a result solely in terms of the average sensitivity by combining both results.

Corollary 1.3. *For all functions f, so that both $g(x \wedge y)$ and $g(x \oplus y)$ with $g : \{0,1\}^n \to \{0,1\}$ reduce to f,*

$$BQC(f) = \Omega\left(\sqrt{\frac{\bar{s}(g)}{\log n}}\right).$$

If, e.g., $f(x,y,z) = g((x \wedge y) \oplus z)$, with x held by Alice and y, z held by Bob, the required reductions are trivial. For many functions, e.g., for g being the majority function, it is easy to reduce $g(x \oplus y)$ on $2 \cdot n$ inputs directly to $g(x \wedge y)$ on more inputs using $x_i \oplus y_i = \neg x_i \wedge y_i + x_i \wedge \neg y_i$ (plus the addition of a few dummy variables), and so the lower bound of Corollary 1.3 can sometimes be used for $g(x \wedge y)$. Note that unlike in Razborov's bounds in [37] the function g does not need to be symmetric.

We then modify the lower bound methods and show how we may replace the Fourier coefficients with the singular values of the communication matrix (divided by 2^n). This means that we may replace the Fourier transform with other unitary transforms and sometimes get much stronger lower bounds.

Application of the new methods to the Boolean function

$$MAJ_n(x,y) = 1 \iff \sum_i (x_i \wedge y_i) \geq n/2$$

yields a lower bound of $\Omega(n/\log n)$ for its bounded error quantum communication complexity. MAJ_n is a function for which neither bounded error quantum nor nondeterministic quantum protocols are efficient, while the discrepancy bound is still only $O(\log n)$.

We then apply the same approach to

$$COUNT_n^t(x,y) = 1 \iff \sum_i (x_i \wedge y_i) = t.$$

These functions have a classical complexity of $\Theta(n)$ for all $t \leq n/2$, since one can easily reduce the disjointness problem to these functions ($DISJ_n$ is the complement of $COUNT_n^0$). We show the following.

Corollary 1.4.

1. $\Omega(n^{1-\epsilon}/\log n) \leq BQC(COUNT_n^{n^{1-\epsilon}}) \leq O(n^{1-\epsilon/2}\log n)$.
2. $BPC(COUNT_n^t) = \Theta(n)$ *for all* $t \leq n/2$.

These are the first lower bounds for functions which allow a polynomial quantum speedup.

Prior to this paper the only known general method for proving lower bounds for the bounded error quantum communication complexity has been the discrepancy method. We show that for any application of the discrepancy bound to HAM_n^t, MAJ_n, and $COUNT_n^t$, the result is only $O(\log n)$. To show this we characterize the discrepancy bound within a constant multiplicative factor and an additive log-factor as the classical weakly unbounded error communication complexity PC (see sections 2.2 and 2.4 for definitions).

Corollary 1.5. *For all* $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$,

$$\max_\mu \log(1/disc_\mu(f)) \leq O(UPC(f)) \leq O\left(\max_\mu \log(1/disc_\mu(f)) + \log n\right),$$

where μ *denotes distributions on* $\{0,1\}^n \times \{0,1\}^n$.

This explains why the discrepancy bound in applications is usually not a good lower bound for bounded error communication complexity, since the weakly unbounded error complexity is always asymptotically at most as large as, e.g., the classical nondeterministic complexity. For our examples the new lower bound methods are exponentially better than the discrepancy bound. In light of Corollary 1.5, it becomes clear that, actually, lower bounds using discrepancy follow the approach of simulating quantum bounded error protocols by classical unbounded error protocols and subsequent application of a classical lower bound.

We conclude also that the discrepancy bound subsumes other methods for proving lower bounds on the weakly unbounded error communication complexity [16]. Furthermore we investigate quantum protocols with weakly unbounded error and show that quantum and classical weakly unbounded error communication complexities are asymptotically equivalent.

The organization of the paper is as follows. In section 2 we describe the necessary technical background. Section 3 shows how we can decompose quantum protocols into weighted rectangle covers of the communication matrix. Sections 4 and 6 then describe our main lower bound techniques, while sections 5 and 7 show how to apply these to specific functions and derive Corollaries 1.1, 1.2, and 1.4. Section 8 is concerned with the power of classical and quantum weakly unbounded error protocols. Section 9 discusses recent developments and open problems.

2. Preliminaries. Note that we consider functions with range $\{0,1\}$ as well as with range $\{-1,1\}$. If a result is stated for functions with range $\{0,1\}$, then it also holds for $\{-1,1\}$. Some results are stated only for functions with range $\{-1,1\}$. The communication complexity does not depend on that choice, so this means that certain parameters in the lower bounds are dependent on the range.

2.1. Quantum states and transformations. Quantum mechanics is usually formulated in terms of states and transformations of states. See [32] for general information on this topic with an orientation on quantum computing.

In quantum mechanics, pure states are unit norm vectors in a Hilbert space, usually the space $\mathbb{C}^k$. We use the Dirac notation for pure states. So a pure state is denoted $|\phi\rangle$ or $\sum_{x\in\{0,\ldots,k-1\}}\alpha_x|x\rangle$ with $\sum_{x\in\{0,\ldots,k-1\}}|\alpha_x|^2=1$ and with $\{\,|x\rangle\,|x\in\{0,\ldots,k-1\}\}$ being an orthonormal basis of $\mathbb{C}^k$.

Inner products in the Hilbert space are denoted $\langle\phi|\psi\rangle$.

If $k=2^l$, then the basis is also denoted $\{\,|x\rangle\,|x\in\{0,1\}^l\}$. In this case the space $\mathbb{C}^{2^l}$ is the l-wise tensor product of the space $\mathbb{C}^2$. The latter space is called a qubit, and the former space consists of l qubits.

As usual, measurements of observables and unitary transformations are considered as basic operations on states; see [32] for definitions.

2.2. The communication model. Now we provide definitions of the computational models considered in the paper. We begin with the model of classical communication complexity.

Definition 2.1. *Let $f: X\times Y\rightarrow\{0,1\}$ be a function. In a classical communications protocol, players Alice and Bob receive $x\in X$ and $y\in Y$, resp., and compute $f(x,y)$. The players exchange binary encoded messages.*

In a deterministic protocol all computations of Alice and Bob are deterministic. The communication complexity of a protocol is the worst case number of bits exchanged for any input. The deterministic communication complexity $DC(f)$ of f is the complexity of an optimal protocol for f.

In a randomized protocol both players have access to private random bits. In the bounded error model the output is required to be correct with probability $1-\epsilon$ for some constant $1/2>\epsilon\geq 0$. The bounded error randomized communication complexity of a function $BPC_\epsilon(f)$ is then defined analogously to the deterministic communication complexity. The worst case communication is taken over both the inputs and the random bits. We set $BPC(f)=BPC_{1/3}(f)$.

In a weakly unbounded error protocol the output has to be correct with probability exceeding $1/2$. If the worst case error of the protocol (over all inputs and coin tosses) is $1/2-\delta$ and the worst case communication is c, then the cost of the protocol is defined as $c-\lfloor\log\delta\rfloor$. The cost of an optimal weakly unbounded error protocol for a function is called $UPC(f)$.

Definition 2.2. *Let us note that the communication matrix of a function f : $X\times Y\rightarrow Z$ is the matrix with rows labeled by $x\in X$, columns labeled by $y\in Y$, and the* *entry in row x and column y equal to $f(x, y) \in Z$. A rectangle in the communication matrix is a product set of inputs labeled by $A \times B$ with $A \subseteq X$ and $B \subseteq Y$. Such a rectangle is monochromatic iff all its entries are equal.*

It is easy to see that a deterministic protocol partitions the communication matrix into a set of monochromatic rectangles, each corresponding to the set of inputs sharing the same communication string produced in the run of the protocol.

The above notion of weakly unbounded error protocols coincides with another type of protocol, namely, majority nondeterministic protocols, which accept an input whenever there are more nondeterministic computations leading to acceptance than to rejection. For a proof see Theorem 10 in [20]. So, weakly unbounded error protocols correspond to certain majority covers for the communication matrix as follows.

FACT 2.3. *There is a weakly unbounded error protocol with cost $O(c)$ iff there is a set of $2^{O(c)}$ rectangles, each labeled either 1 or 0, such that for every input at least one half of the adjacent rectangles have the label $f(x, y)$.*

Note that there is another type of protocol, the truly unbounded error protocol, in which the cost is not dependent on the error, defined by Paturi and Simon [34]. Recently a linear lower bound for the unbounded error communication complexity of IP_n has been obtained in [17]. It is not hard to see that the same bound holds for quantum communication as well. An interesting observation is that the lower bound method of [17] is actually equivalent to the discrepancy lower bound restricted to the uniform distribution.

Now we turn to quantum communication protocols. For a more formal definition of quantum protocols see [41].

DEFINITION 2.4. *In a quantum protocol both players have a private set of qubits. Some of the qubits are initialized to the input before the start of the protocol, while the other qubits are in state $|0\rangle$. In a communication round, one of the players performs some unitary transformation on the qubits in his or her possession and then sends one of these qubits to the other player (the latter step does not change the global state but rather the possession of individual qubits). The choices of the unitary operations and of the qubit to be sent are fixed in advance by the protocol.*

At the end of the protocol the state of some qubit belonging to one player is measured and the result is taken as the output and communicated to the other player. The communication complexity of the protocol is the number of qubits exchanged.

In a (bounded error) quantum protocol the correct answer must be given with probability $1 - \epsilon$ for some $1/2 > \epsilon \geq 0$. The (bounded error) quantum complexity of a function, called $BQC_\epsilon(f)$, is the complexity of an optimal protocol for f. $BQC(f) = BQC_{1/3}(f)$.

In a weakly unbounded error quantum protocol the output has to be correct with probability exceeding $1/2$. If the worst case error of the protocol (over all inputs) is $1/2 - \delta$ and the worst case communication is c, then the cost of the protocol is defined as $c - \lfloor \log \delta \rfloor$. The cost of an optimal weakly unbounded error protocol for a function is called $UQC(f)$.

In a nondeterministic quantum protocol for a Boolean function f all inputs in $f^{-1}(0)$ have to be rejected with certainty, while all other inputs have to be accepted with positive probability. The corresponding complexity is denoted $NQC(f)$.

We have to note that in the defined model no intermediate measurements are allowed to control either the choice of qubits to be sent or the time of the final measurement. Thus for all inputs the same amount of communication and the same number of message exchanges are used. As a generalization one could allow intermediate measurements, whose results could be used to choose (several) qubits to be sent and possibly when to stop the communication protocol. One would have to make sure that the receiving player knows when a message ends. While the model in our definition is in the spirit of the "interacting quantum circuits" definition given by Yao [41], the latter definition would more closely resemble "interacting quantum Turingmachines." Obviously the latter model can be simulated by the former such that in each communication round exactly one qubit is communicated. All measurements can then be deferred to the end by standard techniques. This increases the overall communication by a factor of 2 (but leads to an increase in the number of message exchanges).

2.3. Fourier analysis. We consider functions $f : \{0,1\}^n \to \mathbb{R}$. Define

$$\langle f, g\rangle = \frac{1}{2^n} \sum_{x\in\{0,1\}^n} f(x)\cdot g(x)$$

as an inner product and use the norm $||f||_2 = \sqrt{\langle f, f\rangle}$. We identify $\{0,1\}^n$ with $\mathbb{Z}_2^n$ and describe the Fourier transform. A basis for the space of functions from $\mathbb{Z}_2^n \to \mathbb{R}$ is given by

$$\chi_z(x) = (-1)^{IP_n(x,z)}$$

for all $z \in \mathbb{Z}_2^n$. Then the Fourier transform of f with respect to that basis is

$$\sum_z \hat{f}_z \chi_z,$$

where the $\hat{f}_z = \langle f, \chi_z\rangle$ are called the Fourier coefficients of f. If the functions are viewed as vectors, this is closely related to the Hadamard transform used in quantum computing.

The following facts are well known.

Fact 2.5 (Parseval). *For all f, $||f||_2^2 = \sum_z \hat{f}_z^2$.*

Fact 2.6 (Cauchy–Schwarz).

$$\sum_z \hat{f}_z^2 \cdot \sum_z \hat{g}_z^2 \geq \left(\sum_z |\hat{f}_z \cdot \hat{g}_z|\right)^2.$$

When we consider (communication) functions $f : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{R}$, we use the basis functions

$$\chi_{z,z'}(x,x') = (-1)^{IP_n(x,z)+IP_n(x',z')}$$

for all $z, z' \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$ in Fourier transforms. The Fourier transform of f with respect to that basis is

$$\sum_{z,z'} \hat{f}_{z,z'} \chi_{z,z'},$$

where the $\hat{f}_{z,z'} = \langle f, \chi_{z,z'}\rangle$ are the Fourier coefficients of f.

We will decompose communication protocols into sets of weighted rectangles. For each rectangle $R_i = A_i \times B_i \subseteq \{0,1\}^n \times \{0,1\}^n$ let R_i, A_i, B_i also denote the characteristic functions associated with the rectangle. Then let $\alpha_i = |A_i|/2^n$ be the uniform probability of x being in the rectangle, and let $\beta_i = |B_i|/2^n$ be the uniform probability of y being in the rectangle. Let $\hat{\alpha}_{z,i}$ denote the Fourier coefficients of A_i, and let $\hat{\beta}_{z,i}$ denote the Fourier coefficients of B_i. It is easy to see that $\hat{\alpha}_{z,i} \cdot \hat{\beta}_{z',i}$ is the z, z'-Fourier coefficient of the rectangle function R_i.

For technical reasons we will sometimes work with functions f, whose range is $\{-1, 1\}$. Note that we can set $f = 2g - 1$ for a function g with range $\{0, 1\}$. Since the Fourier transform is linear, the effect on the Fourier coefficients is that they get multiplied by 2, except for the coefficient of the constant basis function, which is also decreased by 1.

2.4. Discrepancy, sensitivity, and entropy.

We now define the discrepancy bound.

DEFINITION 2.7. *Let μ be any distribution on $\{0,1\}^n \times \{0,1\}^n$ and let f be any function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$. Then let*

$$disc_\mu(f) = \max_R |\mu(R \cap f^{-1}(0)) - \mu(R \cap f^{-1}(1))|,$$

where R runs over all rectangles in the communication matrix of f.

Then denote $disc(f) = \min_\mu disc_\mu(f)$.

The application to communication complexity is as follows (see [28] for a less general statement; we also provide a proof for completeness at the end of section 3).

FACT 2.8. *For all f,*

$$BQC_{1/2-\epsilon}(f) = \Omega(\log(\epsilon/disc(f))).$$

A quantum protocol which computes a function f correctly with probability $1/2 + \epsilon$ over a distribution μ on the inputs (and over its measurements) needs at least $\Omega(\log(\epsilon/disc_\mu(f)))$ communication.

We will prove a lower bound on quantum communication complexity in terms of average sensitivity. The average sensitivity of a function measures how many of the n possible bit flips in a random input change the function value. We define this formally for functions with range $\{-1, 1\}$.

DEFINITION 2.9. *Let $f : \{0,1\}^n \to \{-1,1\}$ be a function. For $a \in \{0,1\}^n$ let $s_a(f) = \sum_{i=1}^n \frac{1}{2}|f(a) - f(a \oplus e_i)|$ for the vector e_i containing a one at position i and zeroes elsewhere. $s_a(f)$ is the sensitivity of f at a. Then the average sensitivity of f is defined as $\bar{s}(f) = \sum_{a \in \{0,1\}^n} \frac{1}{2^n} s_a(f)$.*

The connection to Fourier analysis is made by the following fact first observed in [23].

FACT 2.10. *For all $f : \{0,1\}^n \to \{-1,1\}$,*

$$\bar{s}(f) = \sum_{z \in \{0,1\}^n} |z| \cdot \hat{f}_z^2.$$

So the average sensitivity can be expressed in terms of the expected "height" of Fourier coefficients under the distribution induced by the squared coefficients.

One more notion we will use in lower bounds is entropy.

DEFINITION 2.11. *The entropy of a vector $(f_1, \ldots, f_m)$ with $f_i \geq 0$ for all i and $\sum f_i \leq 1$ is $H(f) = -\sum_{i=1}^m f_i \log f_i$.*

We follow the convention $0 \log 0 = 0$. We will consider the entropy of the vector of squared Fourier coefficients $H(\hat{f}^2) = -\sum_z \hat{f}_z^2 \log(\hat{f}_z^2)$. This quantity has the following useful property.

LEMMA 2.12. *For any $f : \{0,1\}^n \to \mathbb{R}$ with $||f||_2 \leq 1$,*

$$H(\hat{f}^2) \leq 2\log\left(1 + \sum_{z\in\{0,1\}^n} |\hat{f}_z|\right).$$

Proof.

$$\begin{aligned} H(\hat{f}^2) &= \sum_z \hat{f}_z^2 \log\frac{1}{|\hat{f}_z|^2} \\ &= 2\left(\sum_z \hat{f}_z^2 \log\frac{1}{|\hat{f}_z|} + \left(1 - \sum_z \hat{f}_z^2\right)\cdot \log 1\right) \\ &\leq 2\log\left(\sum_z \hat{f}_z^2 \frac{1}{|\hat{f}_z|} + \left(1 - \sum_z \hat{f}_z^2\right)\cdot 1\right) \qquad \text{(by Jensen's inequality)} \\ &\leq 2\log\left(1 + \sum_z |\hat{f}_z|\right). \qquad \square \end{aligned}$$

3. Decomposing quantum protocols. In this section we show how to decompose a quantum protocol into a set of weighted rectangles, whose sum approximates the communication matrix.

LEMMA 3.1. *For all Boolean functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, and for all $1/2 > \epsilon > 0$, if there is a quantum protocol for f with communication c and error $1/3$, then there is a real $\alpha \in [0,1]$, and a set of $2^{O(c\log(1/\epsilon))}/\epsilon^4$ rectangles R_i with weights $w_i \in \{-\alpha, \alpha\}$, so that*

$$\sum_i w_i R_i[x,y] \in \begin{cases} [1-\epsilon, 1] & \textit{for } f(x,y) = 1, \\ [0,\epsilon] & \textit{for } f(x,y) = 0. \end{cases}$$

Proof. First we perform the usual success amplification to boost the success probability of the quantum protocol to $1 - \epsilon/4$, increasing the communication to $c' = O(c\log(1/\epsilon))$ at most. Using standard techniques [7] we can assume that all amplitudes used in the protocol are real. Now we employ the following fact proved in [28] and [41].

FACT 3.2. *The final state of a quantum protocol exchanging c' qubits on an input (x,y) can be written*

$$\sum_{m\in\{0,1\}^{c'}} \alpha_m(x)\beta_m(y)|A_m(x)\rangle|m_{c'}\rangle|B_m(y)\rangle,$$

where $|A_m(x)\rangle, |B_m(y)\rangle$ are pure states and $\alpha_m(x), \beta_m(y)$ are real numbers from the interval $[-1,1]$.

Now let the final state of the protocol on (x,y) be

$$\sum_{m\in\{0,1\}^{c'}} \alpha_m(x)\beta_m(y)|A_m(x)\rangle|m_{c'}\rangle|B_m(y)\rangle,$$

and let

$$\phi(x,y) = \sum_{m\in\{0,1\}^{c'-1}} \alpha_{m1}(x)\beta_{m1}(y)|A_{m1}(x)\rangle|1\rangle|B_{m1}(y)\rangle$$

be the part of the state which yields output 1. The acceptance probability of the protocol on (x, y) is now the inner product $\langle\phi(x, y)|\phi(x, y)\rangle$. Using the convention

$$a_{mp}(x) = \alpha_{m1}(x)\alpha_{p1}(x)\langle A_{m1}(x)|A_{p1}(x)\rangle,$$

$$b_{mp}(y) = \beta_{m1}(y)\beta_{p1}(y)\langle B_{m1}(y)|B_{p1}(y)\rangle,$$

this can be written as $\sum_{m,p} a_{mp}(x)b_{mp}(y)$. Viewing a_{mp} and b_{mp} as 2^n-dimensional vectors, and summing their outer products over all m, p yields a sum of $2^{2c'}$ rank 1 matrices containing reals between -1 and 1. Rewrite this sum as $\sum_i \alpha_i\beta_i^T$ with $1 \leq i \leq 2^{2c'}$ to save notation. The resulting matrix is an approximation of the communication matrix within componentwise error $\epsilon/4$.

In the next step define for all i a set $P_{\alpha,i}$ of the indices of positive entries in α_i, and the set $N_{\alpha,i}$ of the indices of negative entries of α_i. Define $P_{\beta,i}$ and $N_{\beta,i}$ analogously. We want all rank 1 matrices to have either only positive or only negative entries. For this we split the matrices into four matrices each, depending on the positivity/negativity of α_i and β_i. Let

$$\alpha'_i(x) = \begin{cases} 0 & \text{if } x \in N_{\alpha,i}, \\ \alpha_i(x) & \text{if } x \in P_{\alpha,i}, \end{cases}$$

and analogously for β'_i; then set the positive entries in α_i and β_i to 0. Consider the sum $\sum_i(\alpha_i\beta_i^T) + \sum_i(\alpha'_i\beta_i^T) + \sum_i(\alpha_i\beta_i\prime^T) + \sum_i(\alpha_i\prime\beta_i\prime^T)$. This sum equals the previous sum, but here all matrices are either nonnegative or nonpositive. Again rename the indices so that the sum is written $\sum_i \alpha_i\beta_i^T$ (to save notation).

At this point we have a set of $C = 2^{2c'+2}$ rank one matrices which are either nonnegative or nonpositive with the above properties. We want to round entries and split matrices into uniformly weighted matrices.

Consider the intervals $[0, \epsilon/(16C)]$ and $[\epsilon/(16C)\cdot k, \epsilon/(16C)\cdot(k+1)]$, for all k up to the least k, so that the last interval includes 1. Obviously there are $O(C/\epsilon)$ such intervals. Round every positive $\alpha_i(x)$ and $\beta_i(x)$ to the upper bound of the first interval it is included in, and change the negative entries analogously by rounding to the upper bounds of the corresponding negative intervals. The overall error introduced on an input (x, y) in the approximating sum $\sum_i \alpha_i(x)\beta_i(y)$ is at most

$$\begin{aligned} &\sum_i \alpha_i(x)\cdot\epsilon/(16C) \\ &\quad + \sum_i \beta_i(y)\cdot\epsilon/(16C) + C\cdot\epsilon^2/(16C)^2 \\ &\leq \epsilon/4. \end{aligned}$$

The sum of the matrices is now between $1 - \epsilon/2$ and $1 + \epsilon/4$ for inputs in $f^{-1}(1)$ and between $-\epsilon/4$ and $\epsilon/2$ for inputs in $f^{-1}(0)$. Add a rectangle with weight $\epsilon/4$ covering all inputs. Dividing all weights by $1 + \epsilon/2$ renormalizes again without increasing the error beyond ϵ.

Now we are left with C rank 1 matrices $\alpha_i\beta_i^T$ containing entries from an $O(C^2/\epsilon^2)$ size set only. Splitting the rank 1 matrices into rectangles containing only the entries with one of the values yields $O(C^3/\epsilon^2)$ weighted rectangles, whose (weighted) sum approximates the communication matrix within error ϵ.

In a last step we replace any rectangle of weight ${}^2/(256C^2(1+\epsilon/2))\cdot k\cdot l$ with kl rectangles of weights $\pm\alpha$ for $\alpha=\epsilon^2/(256C^2(1+\epsilon/2))$. The rectangle weighted $\epsilon/4$ can be replaced with a set of rectangles of weight α each, introducing a negligible error. So the overall number of rectangle is at most $O(C^5/\epsilon^4)=O(2^{10c'}/\epsilon^4)$. □

At first glance the covers obtained in this section seem to be very similar to majority covers: we have a set of rectangles with either negative or positive weights of absolute value α, and if the weighted sum of rectangles adjacent to some input exceeds a threshold, then it is a 1-input. But we have one more property, namely, that summing the weights of the adjacent rectangles approximates the function value. Actually the lower bounds in the next sections and the characterization of majority covers (and weakly unbounded error protocols and the discrepancy bound) in section 8 show that there is an exponential difference between the sizes of the two types of covers.

Now we state another form of the lemma: this time, if the error is close to 1/2, the proof is essentially the same as for Lemma 3.1, omitting the success amplification at the beginning.

LEMMA 3.3. *For all Boolean functions $f:\{0,1\}^n\times\{0,1\}^n\to\{0,1\}$, and for all $1/2>\epsilon>0$, if there is a quantum protocol for f with communication c and error $1/2-\epsilon$, then there is a real $\alpha\in[0,1]$, and a set of $2^{O(c)}/\epsilon^4$ rectangles R_i with weights $w_i\in\{-\alpha,\alpha\}$, so that*

$$\sum_i w_iR_i[x,y]\in\begin{cases}[1/2+\epsilon/2,1] & \textit{for } f(x,y)=1,\\ [0,1/2-\epsilon/2] & \textit{for } f(x,y)=0.\end{cases}$$

Note that all results of this section easily generalize to functions with range $\{-1,+1\}$. Furthermore all the results generalize to partial functions, i.e., the functions may be undefined on some inputs. For those inputs the weighted covers produce an arbitrary weight between 0 and 1.

As an application of the decomposition results we now prove Fact 2.8. A proof of this result seems to be available only in the thesis of Kremer [28] and is stated in less generality there, so we include a proof here.

Proof of Fact 2.8. Obviously it suffices to prove the second statement. Let μ be any distribution on the inputs. Assume there is a protocol with communication c so that the average correctness probability over μ and the measurements of the protocol are at least $1/2+\epsilon$.

Let $P(x,y)$ denote the probability that the protocol accepts x,y and let $K(x,y)$ denote the probability that the protocol is correct on x,y. W.l.o.g. we assume that $\mu(f^{-1}(1))\geq\mu(f^{-1}(0))$. Then we have

$$\begin{aligned}&\sum_{x,y\in f^{-1}(1)}\mu(x,y)P(x,y)\\&\quad-\sum_{x,y\in f^{-1}(0)}\mu(x,y)P(x,y)\\&=\sum_{x,y\in f^{-1}(1)}\mu(x,y)K(x,y)\\&\quad+\sum_{x,y\in f^{-1}(0)}\mu(x,y)K(x,y)-\mu(f^{-1}(0))\\&\geq 1/2+\epsilon-1/2=\epsilon.\end{aligned}$$

Following the construction of Lemma 3.3 we get a set of $C = 2^{O(c)}/\epsilon^4$ rectangles R_i with weights w_i so that the sum of these approximates the acceptance probability of the protocol with componentwise additive error $\epsilon/2$. Then

$$\begin{aligned}&\sum_{x,y\in f^{-1}(1)} \mu(x,y) \sum_{1\leq i\leq C} w_i R_i(x,y)\\ -&\sum_{x,y\in f^{-1}(0)} \mu(x,y) \sum_{1\leq i\leq C} w_i R_i(x,y) \geq \epsilon - \epsilon/2.\end{aligned}$$

Exchanging sums gives us

$$\sum_{1\leq i\leq C} w_i \left(\sum_{x,y\in f^{-1}(1)} \mu(x,y)R_i(x,y) - \sum_{x,y\in f^{-1}(0)} \mu(x,y)R_i(x,y) \right) \geq \epsilon/2$$

and

$$\sum_{1\leq i\leq C} w_i(\mu(f^{-1}(1)\cap R_i) - \mu(f^{-1}(0)\cap R_i)) \geq \epsilon/2.$$

Thus there is a rectangle R_i with $\mu(f^{-1}(1)\cap R_i) - \mu(f^{-1}(0)\cap R_i) \geq (\epsilon/2)/C$, since $|w_i| \leq 1$. But for all rectangles we have $\mu(f^{-1}(1)\cap R_i) - \mu(f^{-1}(0)\cap R_i) \leq disc_\mu(f)$, hence $disc_\mu(f) \geq (\epsilon/2)/C$, and finally

$$\frac{2^{O(c)}}{\epsilon^4} = C \geq (\epsilon/2)/disc_\mu(f) \Rightarrow c \geq \Omega\left(\log \frac{\epsilon}{disc_\mu(f)}\right). \qquad \square$$

4. A Fourier bound. In this section we describe a lower bound method first developed by Raz [35] for classical bounded error communication complexity. We prove that the same method is applicable in the quantum case, using the decomposition results from the previous section. The lower bound method is based on the Fourier transform of the function.

As in section 2.3 we consider the Fourier transform of a communication function. The basis functions are labeled with pairs of strings (z, z'). Denote by V the set of all pairs (z, z). Let $E \subseteq V$ denote some subset of indices of Fourier coefficients.

The basic idea of the lower bound is that the communication must be large when the sum of the absolute values of a small set of Fourier coefficients is large.

THEOREM 4.1. *Let f be a total Boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$. Let $E \subseteq V$. Denote $\kappa_0 = |E|$ (the number of coefficients considered) and $\kappa_1 = \sum_{(z,z)\in E} |\hat{f}_{z,z}|$ (the absolute value sum of coefficients considered). Then*

- *if $\kappa_1 \geq \Omega(\sqrt{\kappa_0})$, then $BQC(f) = \Omega(\log(\kappa_1))$.*
- *if $\kappa_1 \leq O(\sqrt{\kappa_0})$, then $BQC(f) = \Omega(\log(\kappa_1)/(\log(\sqrt{\kappa_0}) - \log(\kappa_1) + 1))$.*

Proof. We are given any quantum protocol for f with error $1/3$ and some worst case communication c. We have to put the stated lower bound on c. Following Lemma 3.1 we can find a set of $2^{O(cd)}$ weighted rectangles, so that the sum of these approximates the communication matrix up to error $1/2^d$ for any $d \geq 1$, where the weights are either α or $-\alpha$ for some real α between 0 and 1. We will fix d later. Let $\{(R_i, w_i) | 1 \leq i \leq 2^{O(cd)}\}$ denote that set. Furthermore let $g(x,y)$ denote the function that maps (x,y) to $\sum_i w_i R_i(x,y)$.

First we give a lower bound on the sum of absolute values of the Fourier coefficients in E for g, in terms of the respective sum for f, using the fact that g approximates f. Obviously $||f-g||_2 \leq 1/2^d$. The identity of Parseval then gives us

$$\sum_{(z,z)\in E} (\hat{f}_{z,z} - \hat{g}_{z,z})^2 \leq ||f-g||_2^2 \leq 2^{-2d}.$$

We make use of the following simple consequence of Fact 2.6.

Fact 4.2. *Let* $|||v|||_2 = \sqrt{\sum_{i=1}^m v_i^2}$, *and* $|||v|||_1 = \sum_{i=1}^m |v_i|$. *Then* $|||v-w|||_2 \geq |||v-w|||_1/\sqrt{m} \geq (|||v|||_1 - |||w|||_1)/\sqrt{m}$.

Hence

$$\begin{aligned}\sum_E |\hat{g}_{z,z}| &\geq \sum_E |\hat{f}_{z,z}| - \sqrt{|E| \cdot \sum_E (\hat{f}_{z,z} - \hat{g}_{z,z})^2} \\ &\geq \kappa_1 - \sqrt{\kappa_0} \cdot 2^{-d}.\end{aligned}$$

Thus the sum of absolute values of the chosen Fourier coefficients of g must be large, if there are not too many such coefficients, or if the error is small enough to suppress their number in the above expression. Call $P = (\kappa_1 - \sqrt{\kappa_0} \cdot 2^{-d})$, so $\sum_E |\hat{g}_{z,z}| \geq P$.

Now due to the decomposition of the quantum protocol used to obtain g, the function is the weighted sum of $C = 2^{O(cd)}$ rectangles. Since the Fourier transform is a linear transformation, the Fourier coefficients of g are weighted sums of the Fourier coefficients of the rectangles. Furthermore the Fourier coefficients of a rectangle are the products of the Fourier coefficients of the characteristic functions of the sets constituting the rectangle, as argued in section 2.3. So $\hat{g}_{z,z} = \sum_i w_i \cdot \hat{\alpha}_{z,i} \cdot \hat{\beta}_{z,i}$ and

$$\sum_E |\hat{g}_{z,z}| \leq \sum_E \sum_i |w_i \cdot \hat{\alpha}_{z,i} \cdot \hat{\beta}_{z,i}|. \tag{4.1}$$

For all rectangles R_i we have $\sum_E \hat{\alpha}_{z,i}^2 \leq ||A_i||_2^2 \leq 1$ by the identity of Parseval. Using the Cauchy–Schwarz inequality (Fact 2.6) we get $\sum_E |\hat{\alpha}_{z,i}\hat{\beta}_{z,i}| \leq 1$. But according to (4.1) the weighted sum of these values, with weights between -1 and 1, adds up to at least P, and so at least $C \geq P$ rectangles are there, and thus $cd = \Omega(\log P)$.

If now $\kappa_1 \geq \Omega(\sqrt{\kappa_0})$, then let $d = O(1)$, and we get the lower bound $c = \Omega(\log(\kappa_1))$. Otherwise set $d = O(\log\sqrt{\kappa_0} - \log\kappa_1 + 1)$ to get $P = \kappa_1/2$ as well as $c = \Omega(\log(P)/d) = \Omega(\log(\kappa_1)/(\log(\sqrt{\kappa_0}) - \log(\kappa_1) + 1))$. □

Let us note one lemma that is implicit in the above proof, and which will be used later.

Lemma 4.3. *Let* $g : \{0,1\}^n \times \{0,1\}^n \to [-1,1]$ *be any function such that there is a set of* Q *rectangles* R_i *with weights* $w_i \in [-1,1]$ *so that* $g(x,y) = \sum_{i=1}^Q w_i R_i(x,y)$ *for all* x, y. *Then*

$$\sum_{z\in\{0,1\}^n} |\hat{g}_{z,z}| \leq Q.$$

5. Applications.

In this section we give applications of the lower bound method.

5.1. Quantum nondeterminism versus bounded error. We first use the lower bound method to prove that nondeterministic quantum protocols may be exponentially more efficient than bounded error quantum protocols. Raz has shown the following [35].

FACT 5.1. *For the function $HAM_n^{n/2}$ consider the set of Fourier coefficients with labels from a set E containing those strings z, z with z having $n/2$ ones. Then*

$$\kappa_0 = \binom{n}{n/2}, \kappa_1 = \binom{n}{n/2}\binom{n/2}{n/4}1/2^n.$$

Thus $\log(\sqrt{\kappa_0}) - \log(\kappa_1) = O(\log n)$. Also $\kappa_1 = \Theta(2^{n/2}/n)$, and thus $\log \kappa_1 = \Theta(n)$.

Applying the lower bound method we get the following:

THEOREM 5.2. *$BQC(HAM_n^{n/2}) = \Omega(n/\log n)$.*

Now we prove that the nondeterministic quantum complexity of $HAM_n^{n/2}$ is small. We use the following technique by de Wolf [40, 22].

FACT 5.3. *Let the nondeterministic rank (denoted* nrank*) of a Boolean function f be the minimum rank of a matrix that contains 0 at positions corresponding to inputs (x, y) with $f(x, y) = 0$ and nonzero reals elsewhere. $NQC = \log \mathrm{nrank}(f) + 1$.*

THEOREM 5.4. *$NQC(HAM_n^{n/2}) = O(\log n)$.*

Proof. It suffices to prove that the nondeterministic rank is polynomial. Define rectangles M_i, which include inputs with $x_i = 1$ and $y_i = 0$, and N_i, which include inputs with $x_i = 0$ and $y_i = 1$. Let E denote the all one matrix. Then let $M = \sum_i(M_i + N_i) - n/2 \cdot E$. This is a matrix which is 0 exactly at those inputs with $\sum_i(x_i \oplus y_i) = n/2$. Furthermore M is composed of $2n + 1$ weighted rectangles and thus the nondeterministic rank of $HAM_n^{n/2}$ is $O(n)$. □

5.2. The complexity of the Hamming distance problem. Now we determine the complexity of HAM_n^t and show that quantum bounded error communication does not allow a significant speedup.

THEOREM 5.5. *Let $t : \mathbb{N} \to \mathbb{N}$ be any monotone increasing function with $t(n) \leq n/2$. Then*

$$BQC(HAM_n^{t(n)}) \geq \Omega\left(\frac{t(n)}{\log t(n)} + \log n\right).$$

Proof. We already know that the complexity of $HAM_n^{n/2}$ is $\Omega(n/\log n)$. Now consider functions HAM_n^t for smaller t. The logarithmic lower bound is obvious from the at most exponential speedup obtainable by quantum protocols [28] compared to deterministic protocols.

Fixing $n - 2t$ pairs of inputs variables to the same values leaves us with $2t$ pairs of free variables, and the function accepts if HAM_{2t}^t accepts on these inputs. Thus the lower bound follows. □

THEOREM 5.6.

$$BPC(HAM_n^t) = O(t \log n).$$

Proof. The protocol determines (and removes) positions in which x, y are different, until no more such positions are present, or until $t + 1$ such positions are found; in both cases the function value can be decided.

Nisan [33] has given a protocol in which Alice and Bob, given n-bit strings x, y, compute the leftmost bit in which x, y differ. The protocol needs communication $O(\log n - \log \epsilon)$ to solve this problem with error ϵ. Hence we can find such a position with error $1/(3t)$ and communication $O(\log n)$, since $t \leq n$. So Alice and Bob can determine, with error $1/3$, whether there are exactly t differences between x and y, using communication $O(t \log n)$ as claimed. □

Let us note that there is another way to prove this upper bound, based on the fingerprinting protocol for EQ_n (see [29]): Alice sends a fingerprint for input x to Bob that allows him to check equality between x and strings z with success probability $1 - 1/n^{2t}$. Such fingerprints can have length $O(t \log n)$. Bob can then go through all z strings in Hamming distance t from y and check whether $z = x$. With high probability all the tests are performed correctly and Bob knows the result. Note that this protocol needs only one message exchange.

6. More Fourier bounds.

In this section we develop more methods for proving lower bounds on quantum communication complexity in terms of properties of their Fourier coefficients. Combining them yields a bound in terms of average sensitivity.

6.1. A bound employing one Fourier coefficient.

Consider functions of the type $f(x, y) = g(x \wedge y)$. The Fourier coefficients of g measure how well the parity function on a certain set of variables is approximated by g. But if g is correlated with a parity (we hope on a large set of variables), then f should be correlated with an inner product function. Then we hope the hardness result stated in Fact 2.8 is applicable (even though f might have low discrepancy).

THEOREM 6.1. *For all total functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ with $f(x, y) = g(x \wedge y)$ and all $z \in \{0,1\}^n$,*

$$BQC(f) = \Omega\left(\frac{|z|}{1 - \log|\hat{g}_z|}\right).$$

Proof. We prove the bound for g with range $\{-1, 1\}$. Obviously the bound itself changes only by a constant factor with this change, and the communication complexity is unchanged.

Let z be the index of any Fourier coefficient of g. Let $|z| = m$. Basically $\hat{g}_z$ measures how well g approximates χ_z, the parity function on the m variables which are 1 in z. Consider the following distribution μ_m on $\{0,1\}^m \times \{0,1\}^m$: Each variable is set to one with probability $\sqrt{1/2}$ and to zero with probability $1 - \sqrt{1/2}$. Then every $x_i \wedge y_i$ is one resp zero with probability $1/2$. So under this distribution on the inputs (x, y) to f we get the uniform distribution on the inputs $z = x \wedge y$ to g.

We will get an approximation of IP_m under μ_m with error $1/2 - |\hat{g}_z|/4$ by taking the outputs of a protocol for f under a suitable distribution. We then use a hardness result for IP_m given by the following lemma.

LEMMA 6.2. *Let μ_m be the distribution on $\{0,1\}^m \times \{0,1\}^m$, that is, the $2m$-wise product of the distribution on $\{0,1\}$, in which 1 is chosen with probability $\sqrt{1/2}$. Then*

$$disc_{\mu_m}(IP_m) \leq O(2^{-m/4}).$$

Clearly with Fact 2.8 we get that computing IP_m with error $1/2 - \epsilon$ under the distribution μ_m needs quantum communication $\Omega(m/4 + \log \epsilon)$.

Let us prove the lemma. Lindsey's lemma (see, e.g., [4]) states the following.

FACT 6.3. *Let R be any rectangle with $a \times b$ entries in the communication matrix of IP_m. Then let*

$$\big||R \cap IP_m^{-1}(1)| - |R \cap IP_m^{-1}(0)|\big| \leq \sqrt{ab2^m}.$$

The above fact allows us to compute the discrepancy of IP_n under the uniform distribution, and will also be helpful for μ_m.

μ_m is uniform on the subset of all inputs x, y containing k ones. Consider any rectangle R. There are at most $\binom{2m}{k}$ inputs with exactly k ones in that rectangle. Furthermore if we intersect the rectangle containing all inputs x, y containing i ones in x and j ones in y with R we get a rectangle containing at most $\binom{m}{i} \cdot \binom{m}{j} \leq \binom{2m}{i+j}$ inputs. In this way R is partitioned into m^2 rectangles, on which μ_m is uniform and Lindsey's lemma can be applied. Note that we partition the set of inputs with overall k ones into up to m rectangles.

Let $\alpha = \sqrt{1/2}$. The probability of any input with k ones is $(1-\alpha)^{2m-k} \cdot \alpha^k$. We get the following upper bound on discrepancy under μ_m:

$$\begin{aligned}
&\sum_{i,j=0}^{m} \alpha^{i+j} \cdot (1-\alpha)^{2m-i-j} \cdot \sqrt{\binom{m}{i}\binom{m}{j}2^m} \\
&\leq m2^{m/2} \cdot \sum_{k=0}^{2m} \alpha^k \cdot (1-\alpha)^{2m-k} \cdot \sqrt{\binom{2m}{k}} \\
&\leq m2^{m/2} \cdot \sqrt{2m+1} \cdot \sqrt{\sum_{k=0}^{2m} \alpha^{2k} \cdot (1-\alpha)^{4m-2k} \cdot \binom{2m}{k}} \\
&\leq m\sqrt{2m+1}2^{m/2}(\alpha^2 + (1-\alpha)^2)^m \\
&\leq m\sqrt{2m+1}2^{m/2}(2-\sqrt{2})^m \\
&\leq O(2^{-m/4}).
\end{aligned}$$

This concludes the proof of Lemma 6.2. □

To describe the way we use this hardness result, first we assume that the quantum protocol for f is errorless. The Fourier coefficient for z measures the correlation between g and the parity function χ_z on the variables that are ones in z. We first show that χ_{1^m} can be computed with error $1/2 - |\hat{g}_z|/2$ from g (or its complement). To see this, consider $\hat{g}_z = \langle g, \chi_z \rangle = \sum_a \frac{1}{2^n} g(a) \cdot \chi_z(a)$. $\hat{g}_z$ as W.l.o.g. assume that the first m variables of z are its ones. So we can rewrite $\hat{g}_z$ as

$$\hat{g}_z = \sum_{b \in \{0,1\}^{n-m}} \frac{1}{2^{n-m}} \sum_{a \in \{0,1\}^m} \frac{1}{2^m} g(ab) \cdot \chi_z(ab).$$

Note that χ_z depends only on the first m variables. In other words, if we fix a random b, the output of g has an expected advantage of $|\hat{g}_z|$ over a random choice in computing parity on the cube spanned by the first m variables. Consequently there must be some b realizing that advantage. We fix that b, and use $g(ab)$ (or $-g(ab)$) to approximate χ_{1^m}. The error of this approximation is $1/2 - |\hat{g}_z|/2$.

Next we show that IP_m, resp., $\chi_{1^m}(x \wedge y) = \chi_z((x \wedge y) \circ b)$ is correlated with $g((x \wedge y) \circ b)$ under some distribution.

Let μ'_n be a distribution resulting from μ_n if all x_i and y_i for $i = m+1, \ldots, n$ are fixed so that $x_i \wedge y_i = b_{i-m}$ and all other variables are chosen as for μ_n. Then

$$\begin{aligned}
&\left| \sum_{(x,y)\in\{0,1\}^{2\cdot n}} \mu'_n(x,y) \cdot g(x \wedge y) \cdot \chi_z(x \wedge y) \right| \\
= &\left| \sum_{a\in\{0,1\}^m} g(ab) \cdot \chi_z(ab) \cdot \sum_{x,y:x\wedge y=ab} \mu'_n(x,y) \right| \\
= &\left| \sum_{a\in\{0,1\}^m} g(ab) \cdot \chi_z(ab) \cdot \frac{1}{2^m} \right| \geq |\hat{g}_z|.
\end{aligned}$$

Hence computing f on μ'_n with no error is at least as hard as computing IP_m on distribution μ_m with error $1/2 - |\hat{g}_z|/2$, which needs at least $\Omega(|z|/4 + \log|\hat{g}_z|)$ qubits communication due to the discrepancy bound.

We assumed previously that f is computed without error. Now assume the error of a protocol for f is $1/3$. Then reduce the error probability to $|\hat{g}_z|/4$ by repeating the protocol $d = O(1 - \log|\hat{g}_z|)$ times and taking the majority output. Computing f on μ'_n with error $|\hat{g}_z|/4$ is at least as hard as computing IP_m on distribution μ_m with error $1/2 - |\hat{g}_z|/2 + |\hat{g}_z|/4$, which needs at least $\Omega(|z|/4 + \log|\hat{g}_z|)$ qubits communication. The error introduced by the protocol is smaller than the advantage of the function f in computing IP_m.

So a lower bound of $\Omega(|z|/4 + \log|\hat{g}_z|)$ holds for the task of computing f with error $|\hat{g}_z|/4$. This implies a lower bound of

$$\frac{\Omega(|z|/4 + \log|\hat{g}_z|)}{d} = \Omega\left(\frac{|z|}{1 - \log|\hat{g}_z|}\right)$$

for the task of computing f with error $1/3$. □

Note that the discrepancy of f in the above theorem may be much higher than the discrepancy of IP_m (leading to weak lower bounds for f), but that f approximates IP_m well enough to transfer the lower bound known for IP_m (which happens to be provable via low discrepancy).

6.2. A sensitivity bound. A weaker, averaged form of the bound in the above subsection is the following.

LEMMA 6.4. *For all functions $f : \{0,1\}^n \times \{0,1\}^n \to \{-1,1\}$ with $f(x,y) = g(x \wedge y)$,*

$$BQC(f) = \Omega\left(\frac{\bar{s}(g)}{H(\hat{g}^2) + 1}\right).$$

Proof. First note that $\bar{s}(g) = \sum_z \hat{g}_z^2 |z|$ by Fact 2.10. Thus we can read the bound

$$BQC(f) = \Omega\left(\frac{\sum_z \hat{g}_z^2 |z|}{\sum_z \hat{g}_z^2 (1 - 2\log|\hat{g}_z|)}\right).$$

The $\hat{g}_z^2$ define a probability distribution on the $z \in \{0,1\}^n$. If we choose a z randomly, then the expected Hamming weight of z is $\bar{s}(g)$. Also the expectation of $1 - 2\log|\hat{g}_z|$ is $1 + H(\hat{g}^2)$. We use the following lemma.

LEMMA 6.5. *Let $a_1, \dots, a_m$ be nonnegative and $b_1, \dots, b_m$ be positive numbers and let $p_1, \dots, p_m$ be a probability distribution. Then there is an i with*

$$\frac{a_i}{b_i} \geq \frac{\sum_j p_j a_j}{\sum_j p_j b_j}.$$

To see the lemma let $a = \sum_j p_j a_j$ and $b = \sum_j p_j b_j$ and assume that for all i we have $a_i b < b_i a$. Then also for all i with $p_i > 0$ we have $p_i a_i b < p_i b_i a$, and hence $b \sum_i p_i a_i < a \sum_i p_i b_i$, which is a contradiction.

So there must be one z, such that $|z|/(1 - \log \hat{g}_z^2) \geq \bar{s}(g)/(1 + H(\hat{g}^2))$. Using that particular z in the bound derived in Theorem 6.1 finishes our proof. □

The above bound decreases with the entropy of the squared Fourier coefficients. This seems unnecessary, since the method of Theorem 4.1 suggests that functions with highly disordered Fourier coefficients should be hard. This leads us to the next bound.

LEMMA 6.6. *For all functions $f : \{0,1\}^n \times \{0,1\}^n \to \{-1,1\}$,*

$$BQC(f) = \Omega\left(\frac{H_D(\hat{f}^2)}{\log n}\right),$$

where $H_D(\hat{f}^2) = -\sum_z \hat{f}_{z,z}^2 \log \hat{f}_{z,z}^2$.

Proof. Consider any quantum protocol for f with communication c. As described in Lemma 3.1, we can find a set of $2^{O(c \log n)}$ weighted rectangles so that their sum yields a function $h(x,y)$ that approximates f entrywise within error $1/n^2$.

Consequently, due to Lemma 4.3, the sum of certain Fourier coefficients of h is bounded:

$$\log \sum_{z \in \{0,1\}^n} |\hat{h}_{z,z}| \leq O(c \log n).$$

Also $-\sum_{z \in \{0,1\}^n} \hat{h}_{z,z}^2 \log \hat{h}_{z,z}^2 \leq 2 \log(1 + \sum_{z \in \{0,1\}^n} |\hat{h}_{z,z}|) \leq O(c \log n)$ due to Lemma 2.12.

But on the other hand $||f - h||_2 \leq 1/n^2$, which we will use to relate $H_D(\hat{f}^2)$ to $H_D(\hat{h}^2)$. We employ the following lemma.

LEMMA 6.7. *Let $f, h : \{0,1\}^n \times \{0,1\}^n \to \mathbb{R}$ with $||f||_2, ||h||_2 \leq 1$. Then*

$$\sum_{z \in \{0,1\}^n} |\hat{f}_{z,z}^2 - \hat{h}_{z,z}^2| \leq 3||f - h||_2.$$

Let us prove the lemma. Define

$$\mathrm{Min}_z = \begin{cases} \hat{f}_{z,z} & \text{if } |\hat{f}_{z,z}| \leq |\hat{h}_{z,z}|, \\ \hat{h}_{z,z} & \text{if } |\hat{h}_{z,z}| < |\hat{f}_{z,z}| \end{cases}$$

and

$$\mathrm{Max}_z = \begin{cases} \hat{f}_{z,z} & \text{if } |\hat{f}_{z,z}| > |\hat{h}_{z,z}|, \\ \hat{h}_{z,z} & \text{if } |\hat{h}_{z,z}| \geq |\hat{f}_{z,z}|. \end{cases}$$

Then $\sum_{z \in \{0,1\}^n} |\hat{f}_{z,z}^2 - \hat{h}_{z,z}^2| = \sum_z \mathrm{Max}_z^2 - \mathrm{Min}_z^2$ and

$$||f - h||_2^2 \geq \sum_z (\hat{f}_{z,z} - \hat{h}_{z,z})^2 = \sum_z (\mathrm{Min}_z - \mathrm{Max}_z)^2.$$

Due to the triangle inequality, we have

$$\sqrt{\sum_z \mathrm{Min}_z^2} + ||f-h||_2 \geq \sqrt{\sum_z \mathrm{Max}_z^2}$$

and

$$\sqrt{\sum_z \mathrm{Min}_z^2} \geq \sqrt{\sum_z \mathrm{Max}_z^2} - ||f-h||_2,$$

which implies

$$\sum_z \mathrm{Min}_z^2 \geq \sum_z \mathrm{Max}_z^2 - 2\sqrt{\sum_z \mathrm{Max}_z^2} \cdot ||f-g||_2$$

and

$$\begin{aligned}\sum_z \mathrm{Max}_z^2 - \mathrm{Min}_z^2 &\leq 2\sqrt{\sum_z \mathrm{Max}_z^2} \cdot ||f-h||_2\\ &\leq 2\sqrt{\sum_z \hat{f}_{z,z}^2 + \hat{h}_{z,z}^2} \cdot ||f-h||_2\\ &\leq 2\sqrt{2}||f-h||_2.\end{aligned}$$

Lemma 6.7 is proved.

So the distribution given by the squared z,z-Fourier coefficients of f is close to the vector of the squared z,z-Fourier coefficients of h. Then also the entropies are quite close, by the following fact (see Theorem 16.3.2 in [15]).

FACT 6.8. *Let p,q be distributions on $\{0,1\}^n$ with $d = \sum_z |p_z - q_z| \leq 1/2$. Then $|H(p) - H(q)| \leq d \cdot n - d \log d$.*

Actually the fact also holds if p,q are subdistributions, i.e., if they consist of nonnegative numbers summing up to at most 1.

So we get

$$H_D(\hat{h}^2) \geq H_D(\hat{f}^2) - O(1/n).$$

Remembering that $H_D(\hat{h}^2) = O(c \log n)$ we get

$$H_D(\hat{f}^2) \leq O(c \log n + 1/n).$$

This concludes the proof. □

If $f(x,y) = g(x \oplus y)$, then $H_D(\hat{f}^2) = H(\hat{f}^2) = H(\hat{g}^2)$. Now we would like to get rid of the entropies in our lower bounds at all, since the entropy of the squared Fourier coefficients is, in general, hard to estimate. Therefore we would like to combine the bounds of Lemmas 6.4 and 6.6. The first holds for functions $g(x \wedge y)$; the second for functions $g(x \oplus y)$.

DEFINITION 6.9. *A communication problem $f : \{0,1\}^n \times \{0,1\}^n \to \{-1,1\}$ can be reduced to another problem $h : \{0,1\}^m \times \{0,1\}^m \to \{-1,1\}$ if there are functions a,b so that $f(x,y) = h(a(x), b(y))$ for all x,y.*

In this case the communication complexity of h is at least as large as the communication complexity of f. Note that if m is much larger than n, a lower bound which is a function of n translates into a lower bound, which is a function of m, and is thus "smaller." For more general types of reductions in communication complexity see [4].

If we can reduce $g(x \wedge y)$ and $g(x \oplus y)$ to some f, then combining the bounds of Lemmas 6.4 and 6.6 gives a lower bound of $\Omega(\bar{s}(g)/(1+H(\hat{g}^2))+H(\hat{g}^2)/\log n)$, which yields Corollary 1.3.

6.3. A bound involving singular values. We return to the technique of Lemma 6.6. For many functions, like IP_m, the entropy of the squared diagonal Fourier coefficients is small, because these coefficients are all very small. We consider the entropy of a vector of values that sum to something much smaller than 1 in many cases. Consequently it may be useful to consider other unitary transformations instead of the Fourier transform.

It is well known that any quadratic matrix M can be brought into diagonal form by multiplying with unitary matrices; i.e., there are unitary U, V so that $M = UDV^*$ for some positive diagonal D. The entries of D are the singular values of M, i.e., they are unique and equal to the eigenvalues of $\sqrt{MM^*}$; see [8].

Consider a communication matrix for a function $f : \{0,1\}^n \times \{0,1\}^n \to \{-1,1\}$. Then let M_f denote the communication matrix divided by 2^n. Let $\sigma_1(f), \dots, \sigma_{2^n}(f)$ denote the singular values of M_f in some decreasing order. In the case when M_f is symmetric these are just the absolute values of its eigenvalues. Let $\sigma^2(f)$ denote the vector of squared singular values of M_f. Note that the sum of the squared singular values is 1. The following theorem is a modification of Lemma 6.6 and Theorem 4.1.

THEOREM 6.10. *Let $f : \{0,1\}^n \times \{0,1\}^n \to \{-1,1\}$ be a total Boolean function. Then $BQC(f) = \Omega(H(\sigma^2(f))/\log n)$.*

Let $\kappa_k = \sigma_1(f) + \cdots + \sigma_k(f)$.

If $\kappa_k \geq \Omega(\sqrt{k})$, then $BQC(f) = \Omega(\log(\kappa_k))$.

If $\kappa_k \leq O(\sqrt{k})$, then $BQC(f) = \Omega(\log(\kappa_k)/(\log(\sqrt{k}) - \log(\kappa_k) + 1))$.

Proof. We first consider the entropy bound and proceed similarly as in the proof of Lemma 6.6. Let f be the considered function and let h be the function computed by a protocol decomposition with error $1/n^2$ consisting of P rectangles with $\log P = O(c \log n)$ for the communication complexity c of some protocol computing f with error 1/3.

M_f denotes the communication matrix of f divided by 2^n; let M_h be the corresponding matrix for h. Using the Frobenius norm on the matrices, we have $||M_f - M_h||_F = ||f-h||_2 \leq 1/n^2$. Then also the singular values of the matrices are close due to the Hoffmann–Wielandt theorem for singular values; see Corollary 7.3.8 in [21].

FACT 6.11. *Let A, B be two square matrices with singular values $\sigma_1 \geq \cdots \geq \sigma_m$ and $\mu_1 \geq \cdots \geq \mu_m$. Then*

$$\sqrt{\sum_i (\sigma_i - \mu_i)^2} \leq ||A - B||_F.$$

As in Lemma 6.6 we can use Lemma 6.7 to show that the L_1-distance between the vector of squared singular values of M_f and the corresponding vector for M_h is bounded and use Fact 6.8 to show that the entropies of the squared singular values of M_f and M_h are at most $o(1)$ apart.

It remains to show that $H(\sigma^2(h))$ is upper bounded by $\log P$. Due to Lemma 2.12 $H(\sigma^2(h)) \leq 2\log(1+\sum_i \sigma_i(h))$. Due to the Cauchy–Schwarz inequality, we have

$$\begin{aligned}
&2\log\left(1+\sum_i \sigma_i(h))\right)\\
&\leq 2\log\sqrt{\sum_i \sigma_i^2(h)}\sqrt{\operatorname{rank}(M_h)}+O(1)\\
&\leq \log\operatorname{rank}(M_h)+O(1)\leq \log P+O(1).
\end{aligned}$$

The last step holds since M_h is the sum of P rank 1 matrices. We get the desired lower bound.

To prove the remaining part of the theorem we argue as in the proof of Theorem 4.1 that the sum of the selected singular values of M_h is large compared to the sum of the selected singular values of M_f, then upper bound the former as above by the rank of M_h and thus by P. The remaining argument is as in the proof of Theorem 4.1. □

Note that for IP_n all singular values are $1/2^{n/2}$, so the entropy of their squares is n, while the entropy of the squared diagonal Fourier coefficients is close to 0, since all these are $\langle IP_n, \chi_{z,z}\rangle^2 = 1/2^{2n}$. The log of the sum of all singular values yields a linear lower bound. In this case the bounds of Lemma 6.6 and Theorem 4.1 are very small, while Theorem 6.10 gives large bounds.

Ambainis [2] has observed that Theorem 6.10 can also be deduced from a lower bound on the quantum communication complexity of sampling [3], using success amplification and an argument relating the smallest number of singular values whose sum is at least $1-\kappa_k^2/(4k)$ to the sum of the first k singular values in the presence of a sufficiently small error.

Note that Theorem 6.10 does not necessarily generalize our other bounds in the sense that the results obtained by using Theorem 6.10 are better for all functions.

We mention that the quantity $\sigma_1+\cdots+\sigma_k$ is known as the Ky Fan k-norm of a matrix [8]. Well-known examples of such norms are the case $k=1$, which is the spectral norm, and the case of maximal k, known as the trace norm. The Ky Fan norms are unitarily invariant for all k, and there is a remarkable fact saying that if matrix A has a smaller Ky Fan k-norm than B for all k, then the same holds for *every* unitarily invariant norm. This leads to the interesting statement that the Raz-type bound in Theorem 6.10 for a function g is smaller than the respective bound for f for all k, iff for all unitarily invariant matrix norms $|||M_g||| \leq |||M_f|||$. Under the same condition the distribution $(\sigma_1^2(f),\ldots,\sigma_{2^n}^2(f))$ induced by the singular values of M_f majorizes the distribution $(\sigma_1^2(g),\ldots,\sigma_{2^n}^2(g))$ induced by M_g. This implies that $H(\sigma^2(f)) \leq H(\sigma^2(g))$. Conversely we get an observation regarding the bounds in Theorem 6.10: if the entropy bound for g is smaller than the entropy bound for f, then there is a k, so that the Raz-type bound for k applied to g is bigger than the corresponding bound for f.

6.4. Examples. To conclude this section we give examples of lower bounds provable using the methods described by Theorem 6.1 and Corollary 1.3.

THEOREM 6.12. $BQC(MAJ_n)=\Omega(n/\log n)$.

Proof. We change the range of MAJ_n to $\{-1,+1\}$. Now consider the Fourier coefficient with index $z=1^n$. $MAJ_n = g(x\wedge y)$ for a function g that is 1, if at least $n/2$ of its inputs are one. W.l.o.g. let $n/2$ be an odd integer. Thus any input to g with $n/2$ ones is accepted by both g and χ_z. Call the set of these inputs I. Similarly every input to g with an odd number of ones larger than $n/2$ is accepted by both d and χ_z, and every input to g with an even number of ones smaller than $n/2$ is rejected by both d and χ_z. On all other inputs g and χ_z disagree. Thus there are $|I|$ inputs more being classified correctly by χ_z than are being classified incorrectly. The Fourier coefficient $\hat{g}_z$ is $2\binom{n}{n/2}/2^n = \Omega(1/\sqrt{n})$. So the method of Theorem 6.1 gives the claimed lower bound. □

Note also that the average sensitivity of the function g with $MAJ_n(x,y) = g(x \wedge y)$ is $\Theta(\sqrt{n})$.

As another example we consider a function $g((x \wedge y) \oplus z)$ with a nonsymmetric g. Let $MED(a)$ be the middle bit of the median of $n/(2\log n)$ numbers of $2\log n$ bits given in a. Let us compute a lower bound on the average sensitivity of MED. For all inputs a there are $\Theta(n/\log n)$ numbers bigger than the median and $\Theta(n/\log n)$ smaller than the median. For each number p different from the median we can switch a single bit to put the changed number below, resp., above the median, shifting the median in the sorted sequence by one position. For a random a such a bit flip entails a change of the middle bit of the median with constant probability. Hence the average sensitivity of MED is at least $\Omega(n/\log n)$. With Corollary 1.3 this gives us a lower bound of $\Omega(\sqrt{n}/\log n)$ on the bounded error quantum communication complexity of $MED((x \wedge y) \oplus z)$.

7. Application: Limits of quantum speedup. Consider $COUNT_n^t(x,y)$. These functions do admit some speedup by quantum protocols; this follows from a black box algorithm given in [9] (see also [5]) and the results of [11] connecting the black box and the communication model.

LEMMA 7.1. *$BQC(COUNT_n^t) = O(\sqrt{nt}\log n)$.*

Note that the classical bounded error communication complexity of all $COUNT_n^t$ is $\Theta(n)$ by a reduction from $DISJ_n$.

THEOREM 7.2. *Let $t : \mathbb{N} \to \mathbb{N}$ be any monotone increasing function with $t(n) \leq n/2$. Then*

$$BQC(COUNT_n^{t(n)}) \geq \Omega\left(\frac{t(n)}{\log t(n)} + \log n\right).$$

Proof. First consider $COUNT_n^{n/2}$. This function is equivalent to a function $g(x \wedge y)$, in which g is 1 if the number of ones in its input is $n/2$ and -1 otherwise. Consider the Fourier coefficient for $z = 1^n$. For simplicity assume that n is even and $n/2$ is odd. Then clearly $\hat{g}_z = 2\binom{n}{n/2}/2^n = \Omega(1/\sqrt{n})$. Thus the method of Theorem 6.1 gives us the lower bound $\Omega(n/\log n)$. Note that finding this lower bound is much easier than the computations in section 5 for $HAM_n^{n/2}$, since we have only to consider one coefficient.

Now consider functions $COUNT_n^t$ for smaller t. The logarithmic lower bound is obvious from the at most exponential speedup obtainable by quantum protocols [28].

Fixing $n/2-t$ pairs of inputs variables to ones and $n/2-t$ pairs of input variables to zeroes leaves us with $2t$ pairs of free variables and the function accepts if $COUNT_{2t}^t$ accepts on these inputs. Thus the lower bound follows. □

Computing the bounds for $t = n^{1-\epsilon}$ yields Corollary 1.4.

8. Discrepancy and weakly unbounded error. Prior to this work, the only general method for proving lower bounds on the quantum bounded error communication complexity has been the discrepancy method. We now characterize the parameter

$disc(f)$ in terms of the communication complexity of f. Due to Fact 2.8 we get for all $\epsilon > 0$,

$$\begin{aligned} &BQC_{1/2-\epsilon}(f) = \Omega(\log(\epsilon/disc(f))) \\ \Rightarrow\ &BQC_{1/2-\epsilon}(f) - \log(\epsilon) = \Omega(\log(1/disc(f))). \end{aligned}$$

Thus $UPC(f) \geq UQC(f) = \Omega(\log(1/disc(f)))$.

THEOREM 8.1. *For all $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, $UPC(f) = O(\log(1/disc(f)) + \log n)$.*

Proof. Let $disc(f) = 1/2^c$. We first construct a protocol with public randomness, constant communication, and error $1/2 - 1/2^{c+1}$, using the Yao principle, and then switch to a usual weakly unbounded protocol (with private randomness) with communication $O(c + \log n)$ and the same error using a result of Newman.

We know that for all distributions μ there is a rectangle with discrepancy at least $1/2^c$. Then the weight of ones is $\alpha + 1/2^{c+1}$ and the weight of zeroes is $\alpha - 1/2^{c+1}$, or vice versa, on that rectangle (for some $\alpha \in [0, 1/2]$).

We take that rectangle and partition the rest of the communication matrix into 2 more rectangles. Assign to each rectangle the label 0 or 1, depending on the majority of function values in that rectangle according to μ. The error of the rectangles is at most 1/2. If a protocol outputs the label of the adjacent rectangle for every input, the error according to μ is only $1/2 - 1/2^{c+1}$.

This holds for all μ. Furthermore the rectangle partitions lead to deterministic protocols with $O(1)$ communication and error $1/2 - 1/2^{c+1}$: Alice sends the names of the rectangles that are consistent with her input. Bob then picks the label of the only rectangle consistent with both inputs.

We now invoke the following lemma due to Yao (as in [29]).

FACT 8.2. *The following statements are equivalent for all f:*

1. *For each distribution μ there is a deterministic protocol for f with error ϵ and communication d.*
2. *There is a randomized protocol in which both players can access a public source of random bits, so that f is computed with error probability ϵ (over the random coins), and the communication is d.*

So we get an $O(1)$ communication randomized protocol with error probability $1/2 - 1/2^{O(c)}$ using public randomness. We employ the following result from [31] to get a protocol with private randomness.

FACT 8.3. *Let f be computable by a probabilistic protocol with error ϵ that uses public randomness and d bits of communication. Then $BPC_{(1+\delta)\epsilon}(f) = O(d + \log(\frac{n}{\epsilon\delta}))$.*

We may now choose $\delta = 1/2^{O(c)}$ small enough to get a weakly unbounded error protocol for f with cost $O(c + \log n)$. □

Let us also consider the quantum version of weakly unbounded error protocols.

THEOREM 8.4. *For all f, $UPC(f) = \Theta(UQC(f))$.*

Proof. The lower bound is trivial, since the quantum protocol can simulate the classical protocol.

For the upper bound we have to construct a classical protocol from a quantum protocol. Consider a quantum protocol with error $1/2 - \epsilon \leq 1/2 - 1/2^c$ and communication c. Due to Lemma 3.3 this gives us a set of $2^{O(c)}$ weighted rectangles, such that the sum of the rectangles approximates the communication matrix entrywise within error $1/2 - \epsilon/2$. The weights are real $\pm\alpha$ with absolute value smaller than 1. Label the $-\alpha$ weighted rectangles with 0 and the other rectangles with 1, and add $(1/2)/\alpha$

rectangles covering all inputs and bearing label 0. This clearly yields a majority cover of size $2^{O(c)}$, which is equivalent to a classical weakly unbounded error protocol using communication $O(c)$ due to Fact 2.3. □

It is easy to see that there are weakly unbounded error protocols for MAJ_n, HAM_n^t, and $COUNT_n^t$ with cost $O(\log n)$. For MAJ_n consider the protocol where Alice picks a random i from 1 to n and sends i, x_i. If $x_i = y_i = 1$ they accept. Clearly, if n is odd, this protocol is correct with probability $1/2 + 1/(2n)$. For even $n > 2$ the protocol must be modified by accepting every input with probability $1/n$ beforehand. Other threshold predicates can be computed similarly.

For HAM_n^t we have w.l.o.g. that $t \leq n/2$, since otherwise we can just complement x and use a protocol for $t' = n - t$. If we have a protocol that works for $t = n/2$ and even n, we can just add $n-2t$ dummy inputs (which are all different for Alice and Bob) to solve the problem for other t, since $t + (n - 2t) = n - t = (n + n - 2t)/2$. The protocol for $HAM_n^{n/2}$ goes as follows: Alice rejects unconditionally with probability $1/3 + 1/(8n^2)$, and otherwise picks i_1, i_2 from 1 to n and sends them along with the corresponding x_i. Bob now accepts if $x_{i_1} \neq y_{i_1} \vee x_{i_2} = y_{i_2}$. For inputs with Hamming distance d the acceptance probability is $(2/3 - 1/(8n^2)) \cdot (1 - (d/n) \cdot (1 - d/n))$. So inputs with $d = n/2$ are accepted with probability $1/2 - 1/O(n^2)$; all other inputs are accepted with probability at least $1/2 + 1/O(n^2)$. The protocol for $COUNT_n^t$ is similar.

MAJ_n is even a complete problem for the class of problems computable with polylogarithmic cost by weakly unbounded error protocols. To see this note that this class is equal to the class of majority nondeterministic protocols with polylogarithmic communication [20], and so MAJ_n is complete by the techniques of [4]. So all these problems allow only small discrepancy bounds.

LEMMA 8.5. *For* $f \in \{MAJ_n, HAM_n^t, COUNT_n^t\}$, $\max_\mu \log(1/disc_\mu(f)) = O(\log n)$.

9. Discussion. In this paper we have investigated the problem of proving lower bounds on the bounded error quantum communication complexity. As opposed to previous approaches our methods both are general and make use of the quantum properties of the protocols (i.e., do not implicitly follow the pattern of simulating a bounded error quantum protocol by an unbounded error classical protocol and employing a lower bound method for the latter). Our results are strong enough to show separations between unbounded error classical and bounded error quantum communications, resp., between quantum nondeterministic and quantum bounded error communications.

Our results do not address the more powerful model of quantum communication complexity with prior entanglement [13, 14]. It would be interesting to obtain similar results for this model. Recently an improved lower bound (compared to [14]) for the complexity of IP_n in this model has been obtained in Nayak and Salzman [30]. However, these bounds do not show hardness under a distribution like in the second statement of Fact 2.8. So constructions similar to that of Theorem 6.1 remain unknown for the model with prior entanglement.

More recently Razborov [37] has obtained much stronger lower bounds on the quantum communication complexity of $g(x \wedge y)$ for *symmetric* functions g, almost tightly characterizing the quantum bounded error communication complexity of these functions, even in the model with prior entanglement. This gives a $\Omega(\sqrt{n})$; lower bound for $DISJ_n$; previously superlogarithmic bounds for this function were known only for the cases when strong restrictions on the interaction are imposed [27] or when the error probability is extremely small [12]. Razborov's techniques are based on showing good lower bounds on the minimal trace norm (sum of singular values) of matrices approximating the communication matrix, similar to the approach in Theorem 6.10. These new results can be used to show that in our Corollary 1.4 actually the upper bounds for $COUNT_n^t$ are tight.

The lower bound methods of this paper can also be applied to other types of functions; see sections 5 and 6.4. It would be interesting to find tighter lower bounds for these functions and to extend our results to the model with prior entanglement.

A major open problem in the area is to determine whether quantum bounded error communication can ever be more than quadratically smaller than classical bounded error communication for total functions. A first step to resolve this problem would be to show a lower bound in terms of (one-sided) block sensitivity on the quantum bounded error complexity of all functions $g(x \wedge y)$ (with nonsymmetric g).

Regarding unbounded error protocols, a result of Forster [17] can easily be extended to show that the discrepancy bound restricted to the uniform distribution is a lower bound on the unbounded error quantum communication complexity (not its weak variant as considered in this paper; i.e., the communication of protocols with error smaller than 1/2 is measured).

Finally, let us mention that the known quantum protocols that give a polynomial speedup compared to randomized protocols for total functions need much interaction, i.e., many communication rounds. It has been shown by Jain, Radhakrishnan, and Sen [38] that this in inevitable for $DISJ_n$. Is there a function $g(x \oplus y)$ which cannot be computed optimally quantum by a 1-round protocol?

Acknowledgments. The author wishes to thank Ronald de Wolf for bringing [35] to his attention and for lots of valuable discussions, Andris Ambainis for pointing out a mistake in an earlier version of the paper, and the referees for helpful suggestions.

REFERENCES

[1] S. Aaronson and A. Ambainis, *Quantum search of spatial regions,* in Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, IEEE, Piscataway, NJ, 2003, pp. 200–209.

[2] A. Ambainis, *Personal communication*, 2001.

[3] A. Ambainis, L. J. Schulman, A. Ta-Shma, U. Vazirani, and A. Wigderson, *The quantum communication complexity of sampling,* in Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science, IEEE, Piscataway, NJ, 1998, pp. 342–351.

[4] L. Babai, P. Frankl, and J. Simon, *Complexity classes in communication complexity theory,* in Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science, IEEE, Piscataway, NJ, 1986, pp. 303–312.

[5] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf, *Quantum lower bounds by polynomials,* in Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science, IEEE, Piscataway, NJ, 1998, pp. 352–361. Available online at http://arxiv.org/abs/quant-ph/9802049.

[6] C. H. Bennett and S. J. Wiesner, *Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states,* Phys. Rev. Lett., 69 (1992), pp. 2881–2884.

[7] E. Bernstein and U. Vazirani, *Quantum complexity theory,* SIAM J. Comput., 26 (1997), pp. 1411–1473.

[8] R. Bhatia, *Matrix Analysis,* Springer-Verlag, New York, 1997.

[9] G. Brassard, P. Høyer, and A. Tapp, *Quantum counting,* in Proceedings of the 25th Annual International Colloquium on Automata, Languages, and Programming, Springer-Verlag, Berlin, 1998, pp. 820–831. Available online at http://arxiv.org/abs/quant-ph/9805082.

[10] H. Buhrman, *Quantum computing and communication complexity,* Theor. Comput. Sci. EATCS, 70 (2000), pp. 131–141.

[11] H. Buhrman, R. Cleve, and A. Wigderson, *Quantum vs. classical communication and computation*, in Proceedings of the 30th Annual ACM Symposium on Theory of Computing, ACM, New York, 1998, pp. 63–68. Available online at http://arxiv.org/abs/quant-ph/9802040.

[12] H. Buhrman and R. de Wolf, *Communication complexity lower bounds by polynomials,* in Proceedings of the 16th Annual IEEE Conference on Computational Complexity, IEEE, Piscataway, NJ, 2001, pp. 120–130. Available online at http://arxiv.org/abs/cs.CC/9910010.

[13] R. Cleve and H. Buhrman, *Substituting quantum entanglement for communication,* Phys. Rev. A, 56 (1997), pp. 1201–1204. Available online at http://arxiv.org/abs/quant-ph/9704026.

[14] R. Cleve, W. van Dam, M. Nielsen, and A. Tapp, *Quantum entanglement and the communication complexity of the inner product function,* in Proceedings of the 1st NASA International Conference on Quantum Computing and Quantum Communications, Springer, Berlin, 1999, pp. 61–74. Available online at http://arxiv.org/abs/quant-ph/9708019.

[15] T. M. Cover and J. A. Thomas, *Elements of Information Theory,* Wiley Ser. Telecom., John Wiley, New York, 1991.

[16] C. Damm, M. Krause, C. Meinel, and S. Waack, *Separating counting communication complexity classes,* in Proceedings of the 9th Annual Symposium on Theoretical Aspects of Computer Science, Springer, Berlin, 1992, pp. 281–292.

[17] J. Forster, *A linear lower bound on the unbounded error probabilistic communication complexity,* in Proceedings of the 16th Annual IEEE Conference on Computational Complexity, IEEE, Piscataway, NJ, 2001, pp. 100–106.

[18] R. M. Gray, *Entropy and Information Theory,* Springer-Verlag, New York, 1990.

[19] L. K. Grover, *A fast quantum mechanical algorithm for database search,* in Proceedings of the 28th Annual ACM Symposium on Theory of Computing, ACM, New York, 1996, pp. 212–219. Available online at http://arxiv.org/abs/quant-ph/9605043.

[20] B. Halstenberg and R. Reischuk, *Relations between communication complexity classes,* J. Comput. Systems Sci., 41 (1990), pp. 402–429.

[21] R. Horn and C. Johnson, *Matrix Analysis,* Cambridge University Press, Cambridge, UK, 1985.

[22] P. Høyer and R. de Wolf, *Improved quantum communication complexity bounds for disjointness and equality,* in Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science, Springer, Berlin, 2002, pp. 299–310. Available online at http://arxiv.org/abs/quant-ph/0109068.

[23] J. Kahn, G. Kalai, and N. Linial, *The influence of variables on Boolean functions,* in Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science, IEEE, Piscataway, NJ, 1988, pp. 68–80.

[24] B. Kalyanasundaram and G. Schnitger, *The probabilistic communication complexity of set intersection,* SIAM J. Discrete Math., 5 (1992), pp. 545–557.

[25] H. Klauck, *Quantum communication complexity,* Workshop on Boolean Functions and Applications at the 27th International Colloquium on Automata, Languages, and Programming, Springer, Berlin, 2000, pp. 241–252. Available online at http://arxiv.org/abs/quant-ph/0005032.

[26] H. Klauck, *Lower Bounds for Quantum Communication Complexity,* in Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science, IEEE, Piscataway, NJ, 2001, pp. 288–297. Available online at http://arxiv.org/abs/quant-ph/0106160.

[27] H. Klauck, A. Nayak, A. Ta-Shma, and D. Zuckerman, *Interaction in quantum communication and the complexity of set disjointness,* in Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, 2001, pp. 124–133.

[28] I. Kremer, *Quantum Communication,* Master's thesis, Hebrew University, Jerusalam, Israel, 1995.

[29] E. Kushilevitz and N. Nisan, *Communication Complexity,* Cambridge University Press, Cambridge, UK, 1997.

[30] A. Nayak and J. Salzman, *On communication over an entanglement-assisted quantum channel,* in Proceedings of the 34th Annual ACM Symposium on Theory of Computing, 2002, pp. 698–704.

[31] I. Newman, *Private vs. common random bits in communication complexity,* Inform. Process. Lett., 39 (1991), pp. 67–71.

[32] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information,* Cambridge University Press, Cambridge, UK, 2000.

[33] N. Nisan, *The communication complexity of threshold gates,* in Combinatorics, Paul Erdős is Eighty, János Bolyai Math. Soc., Budapest, 1993, pp. 301–315.

[34] R. Paturi and J. Simon, *Probabilistic communication complexity,* in Proceedings of the 25th Annual IEEE Symposium on Foundations of Computer Science, IEEE, Piscataway, NJ, 1984, pp. 118–126.

[35] R. Raz, *Fourier analysis for probabilistic communication complexity,* Comput. Complexity, 5 (1995), pp. 205–221.

[36] R. Raz, *Exponential separation of quantum and classical communication complexity,* in Proceedings of the 31st Annual ACM Symposium on Theory of Computing, ACM, New York, 1999, pp. 358–367.

[37] A. A. Razborov, *On the quantum communication complexity of symmetric predicates,* Izv. Russ. Akad. Nayk. Ser. Mat. 67 (2003), pp. 159–176 (in Russian). Available online at http://arxiv.org/abs/quant-ph/0204025 (in English).

[38] R. Jain, J. Radhakrishnan, and P. Sen, *A lower bound for the bounded round quantum communication complexity of set disjointness,* in Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, IEEE, Piscataway, NJ, 2003, pp. 220–229.

[39] A. Ta-Shma, *Classical versus quantum communication complexity,* SIGACT News, 30 (1999), pp. 25–34.

[40] R. de Wolf, *Characterization of nondeterministic quantum query and quantum communication complexity,* in Proceedings of the 15th Annual IEEE Conference on Computational Complexity, IEEE, Piscataway, NJ, 2000, pp. 271–278. Available online at http://arxiv.org/abs/cs.CC/0001014.

[41] A. C. Yao, *Quantum circuit complexity,* in Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science, IEEE, Piscataway, NJ, 1993, pp. 352–361.

SIAM J. COMPUT.
Vol. 37, No. 1, pp. 47–82

ADIABATIC QUANTUM STATE GENERATION[*]

DORIT AHARONOV[†] AND AMNON TA-SHMA[‡]

Abstract. The design of new quantum algorithms has proven to be an extremely difficult task. This paper considers a different approach to this task by studying the problem of *quantum state generation*. We motivate this problem by showing that the entire class of statistical zero knowledge, which contains natural candidates for efficient quantum algorithms such as graph isomorphism and lattice problems, can be reduced to the problem of quantum state generation. To study quantum state generation, we define a paradigm which we call *adiabatic state generation* (ASG) and which is based on adiabatic quantum computation. The ASG paradigm is not meant to replace the standard quantum circuit model or to improve on it in terms of computational complexity. Rather, our goal is to provide a natural theoretical framework, in which quantum state generation algorithms could be designed. The new paradigm seems interesting due to its intriguing links to a variety of different areas: the analysis of spectral gaps and ground-states of Hamiltonians in physics, rapidly mixing Markov chains, adiabatic computation, and approximate counting. To initiate the study of ASG, we prove several general lemmas that can serve as tools when using this paradigm. We demonstrate the application of the paradigm by using it to turn a variety of (classical) approximate counting algorithms into efficient quantum state generators of nontrivial quantum states, including, for example, the uniform superposition over all perfect matchings in a bipartite graph.

Key words. quantum computation, quantum algorithm, adiabatic theorem, Hamiltonians, Markov chains, quantum sampling, state generation, statistical zero knowledge, Zeno effect, spectral gap

AMS subject classification. 81P68

DOI. 10.1137/060648829

1. Introduction.

Quantum computation carries the hope of solving classically intractable tasks in quantum polynomial time. The most notable success so far is Shor's quantum algorithm for factoring integers and for finding the discrete log [45]. Following Shor's algorithm several other algorithms were discovered, such as Hallgren's algorithm for solving Pell's equation [31], Watrous's algorithms for the group black box model [48], and the Legendre symbol algorithm by van Dam and Hallgren [17]. Except for [17] all of these algorithms fall into the framework of the hidden subgroup problem and in fact use exactly the same quantum circuitry; the exception, [17], is a different algorithm but also heavily uses Fourier transforms and exploits the special algebraic structure of the problem. Recently, a beautiful new algorithm by Childs et al. [13] was found which gives an exponential speed-up over classical algorithms using an entirely different approach, namely quantum random walks. The algorithm, however, works in the black box model and solves a fairly contrived problem.

In order to develop new quantum algorithms, it is crucial that we have a larger

[*] Received by the editors January 2, 2006; accepted for publication (in revised form) October 6, 2006; published electronically March 30, 2007. A preliminary version of this paper appeared in *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, San Diego, CA, 2003, pp. 20–29.

http://www.siam.org/journals/sicomp/37-1/64882.html

[†] Department of Computer Science and Engineering, Hebrew University, 91904 Jerusalem, Israel (doria@cs.huji.ac.il). This author's research was supported by an Alon fellowship (The Council of High Education in Israel) under grant 033-7233, and by the Israel Science Foundation under grants 032-9738 and 039-7549.

[‡] Department of Computer Science, Tel Aviv University, 69978 Tel-Aviv, Israel (amnon@post.tau.ac.il). This author's research was supported by the Binational Science Foundation, by the EU Integrated Project QAP, and by the Israel Science Foundation.

variety of quantum algorithmic techniques and approaches. In this paper we attempt to make a step in that direction by studying the problem of quantum algorithm design from a different point of view, that of "quantum state generation."

It has been folklore knowledge for almost a decade already that the problem of graph isomorphism, which is considered hard classically [37], has an efficient quantum algorithm as long as a certain state, namely the superposition of all graphs isomorphic to a given graph,

$$|\alpha_G\rangle = \sum_{\sigma \in S_n} |\sigma(G)\rangle, \tag{1.1}$$

can be generated efficiently by a quantum Turing machine (here and in the rest of the paper we ignore normalizing constants for the sake of brevity). The reason that generating $|\alpha_G\rangle$ suffices is very simple: for two isomorphic graphs G_1 and G_2, the states $|\alpha_{G_1}\rangle$ and $|\alpha_{G_2}\rangle$ are identical, whereas for two nonisomorphic graphs they are orthogonal. Using a simple quantum circuit known as the SWAP test (see section 3.2), one can approximate the inner product between two given states and thus can distinguish between the two cases of orthogonal and parallel $|\alpha_G\rangle$'s.

One is tempted to assume that such a state, $|\alpha_G\rangle$, is easy to construct, since the equivalent classical distribution, namely the uniform distribution over all graphs isomorphic to a certain graph, can be sampled from efficiently. Indeed, the state $|\beta_G\rangle = \sum_{\sigma \in S_n} |\sigma\rangle \otimes |\sigma(G)\rangle$ can easily be generated by this reasoning. However, $|\beta_G\rangle$ is inadequate for our needs, as $|\beta_{G_1}\rangle$ and $|\beta_{G_2}\rangle$ are always orthogonal. It is a curious (and disturbing) fact of quantum mechanics that though $|\beta_G\rangle$ is an easy state to generate, so far no one knows how to generate $|\alpha_G\rangle$ efficiently, because we cannot *forget* the value of $|\sigma\rangle$.

In this paper we systematically study the problem of quantum state generation. We are interested in a restricted version of quantum state generation, that of generating states corresponding to *efficiently samplable* classical probability distributions. To be specific, let C be a classical circuit with n inputs and m outputs. We define the probability distribution D_C to be the distribution over the outputs of the classical circuit C when its inputs are uniformly distributed, i.e., $D_C(z) = \Pr_{x \in \{0,1\}^n}[C(x) = z]$. We denote

$$|C\rangle \overset{\text{def}}{=} \sum_{z \in \{0,1\}^m} \sqrt{D_C(z)}\, |z\rangle$$

and define the quantum sampling (QS) problem.

Definition 1.1 (quantum sampling (QS_δ)).

Input: *A description of a classical circuit C and a constant $\delta \geq 0$.*

Output: *A description of a quantum circuit Q of size poly$(|C|)$, with a marked set of output qubits, such that on input $|0\rangle$ the final state ρ of the output qubits of the circuit Q is close to $|\phi\rangle = |C\rangle$, namely,*

$$\|\rho - |\phi\rangle\langle\phi|\,\|_{tr} \leq \delta.$$

The norm above is the trace norm (see section 2). We say Q quantum samples (or Qsamples) the output distribution of the circuit C. If δ is not specified, we take δ to be some fixed small constant, say 10^{-5}.

The problem of generating the graph isomorphism state from (1.1) is an instance of QS, that of Qsampling the uniform distribution over all isomorphic graphs. We proceed with the study of quantum state generation as follows:

- In section 3 we prove that any problem in the complexity class statistical zero knowledge (SZK) can be reduced to an instance of QS. SZK contains most problems which are considered good candidates for an efficient quantum algorithm, or for which such an algorithm already exists. Hence, this provides a strong motivation for the study of the QS problem. Additional results related to SZK and to the QS problem are given.
- In section 4 we define a new paradigm for quantum state generation, called adiabatic state generation (ASG). We show that the existence of ASG implies the existence of a standard quantum algorithm to generate the same state, that of polynomially related complexity. Thus, in order to design a quantum state generator, it is sufficient to design ASG for the same state. The ASG paradigm is strongly related in spirit to the framework of adiabatic quantum computation and the physical terminology used therein, such as Schrödinger's equation and the adiabatic theorem. Nevertheless, our definition and proofs do not require any knowledge of those notions and can be understood from first principles.
- Section 5 shows that a fairly general class of classical approximate counting algorithms (that use rapidly mixing Markov chains) can be transformed into ASG algorithms that Qsample from the final distributions of the Markov chains. This solves the QS problem for various interesting cases, such as the uniform distribution over all perfect matchings of a given graph. This section draws intriguing links between the ASG paradigm and Markov chains and spectral gap analysis.
- Section 6 collects lemmas that were used in previous sections and which might be useful when applying the ASG paradigm in other cases. These include the Hamiltonian-to-projection and the Hamiltonian-to-measurement lemmas, the jagged adiabatic path lemma, and the sparse-Hamiltonian lemma, and we explain their meaning below.

The problem of QS was also considered by Grover and Rudolph [30], without a name. They show how to apply standard techniques to construct the state $\sum_i \sqrt{p_i}\,|i\rangle$ for a probability distribution $\{p_i\}$ that is "integrable," i.e., for which $\sum_{i=k}^{\ell} p_i$ can be efficiently computed (approximated) given k and ℓ. One can apply these techniques to construct the states that we construct in section 5. This is done by exploiting the self-reducibility of the problems corresponding to these states. We stress, however, that the techniques we develop in this paper are qualitatively and significantly different from previous techniques for generating quantum states and, in particular, do not require self-reducibility. This can be important for extending our approach to other quantum states in which self-reducibility cannot be used.

In the remainder of the introduction we provide overviews of each of the different parts of the paper. We note that each one of these sections can be read in an almost self-contained way.

1.1. QS and SZK (section 3).

Our first observation is an interesting connection between the QS problem and the complexity class SZK (see section 3 for the definition and background on this class).

THEOREM 1.1. *If QS $\in$ BQP, then SZK $\subseteq$ BQP.*

The proof of Theorem 1.1 relies on a result of Sahai and Vadhan [44]. They defined a problem, called statistical difference, and proved it is SZK-complete. We provide a quantum algorithm for the statistical difference problem given a quantum algorithm for QS.

Theorem 1.1 shows that a general quantum algorithm for the problem of QS implies SZK $\subseteq$ BQP.[1] We note that most problems that were shown to be in BQP or are considered good candidates for BQP, such as discrete log, quadratic residuosity, approximating closest and shortest vectors in a lattice, graph isomorphism, and more, belong to SZK. Theorem 1.1 thus connects the problem of QS to all these algorithmic problems. This motivates our definition and study of the QS problem.

A possibly easier task than solving the general QS problem is to solve specific instances of the problem. To this end, one can apply the proof of Theorem 1.1 to a specific problem in SZK. This would lead to the discovery of the relevant QS instance to which the problem can be reduced. In the general case, this might be quite complicated to do, since the proof of Theorem 1.1 uses the nontrivial completeness result of [44]. In some cases, however, the specification of the relevant QS instance is much easier. Three such cases are discrete log, quadratic residuosity, and a certain lattice related problem. We provide in Appendix A explicit specifications of the QS instances (namely, the quantum superpositions) to which each one of these problems can be reduced. Note that we already know efficient quantum algorithms for the first two problems. The case of solving the Qsampling instance associated with the lattice problem is wide open.

It is interesting to ask whether Theorem 1.1 also holds in the other direction. In other words, is solving QS equivalent to solving the SZK problem, or is the QS problem harder? We show that at least for some cases, equivalence holds. It is easy to see that the QS instance corresponding to discrete log can be solved using the quantum algorithm for discrete log. We prove that the same is also true for the graph isomorphism problem; namely, by trying to solve the QS problem for graph isomorphism, we are not making the problem harder.

Finally, we also study the case of perfect Qsampling. One might hope that if $\mathrm{QS}_{\delta=0}$ can be solved in quantum polynomial time, this would imply that SZK lies in quantum polynomial time with the one-sided error, RQP. We do not know how to prove this, but we provide a slightly weaker result.

1.2. The adiabatic quantum state generation paradigm (section 4). In the past few years, a paradigm called *adiabatic quantum computation* which was defined in [22] attracted considerable attention. Adiabatic quantum computation is a framework for quantum algorithms which uses, instead of the unitary gates used in the standard quantum circuit model, the more physical language of Hamiltonians, spectral gaps, and ground-states, which we will soon explain.

Inspired by adiabatic quantum computation, we define a paradigm for designing quantum state generating algorithms (sometimes called quantum state generators) in the standard quantum circuit model. We call this paradigm ASG. Our goal in the definition of ASG is not to replace the quantum circuit model, or to improve on it, but rather to develop a paradigm, or a language, in which the problem of quantum state generation, and QS in particular, can be studied conveniently. The advantage in using the language of the adiabatic computation model is that the task of quantum state generation becomes more natural, since adiabatic evolution is cast into the language of quantum state generation. Furthermore, as we will see, it seems that this language lends itself more easily than the standard circuit model to developing general tools.

Our definition of ASG and results regarding this paradigm do not rely on knowledge of the physical terminology on which adiabatic computation is based, such as

[1] Note that there exists an oracle A relative to which $\mathrm{SZK}^A \not\subseteq \mathrm{BQP}^A$ [1].

Schrödinger's equation and the adiabatic theorem. Nevertheless, since these notions, and the adiabatic computation model in particular, provide so much of the intuition behind our definitions and proofs, we now provide some background regarding these notions to motivate our discussion. We refer the reader to [40, 22, 8] for more information regarding physical background, adiabatic computation, and the adiabatic theorem, respectively.

1.2.1. Adiabatic computation: The physical motivation for ASG. In the standard model of quantum computation, the state of n qubits evolves in *discrete* time steps by unitary operations. In contrast, the underlying physical description of this evolution is *continuous*. This evolution is described by Schrödinger's equation: $\frac{d}{dt}\left|\psi(t)\right\rangle = iH(t)\left|\psi(t)\right\rangle$, where $\left|\psi(t)\right\rangle$ is the state of the n qubits at time t, and $H(t)$ is a Hermitian $2^n \times 2^n$ matrix operating on the space of n qubits. This matrix is called the *Hamiltonian.* The term $\frac{d}{dt}\left|\psi(t)\right\rangle$ stands for $\lim_{\zeta\to 0}\frac{|\psi(t+\zeta)\rangle - |\psi(t)\rangle}{\zeta}$ and is a vector measuring the *direction* in which $\left|\psi(t)\right\rangle$ evolves at a given time t. Loosely speaking, the integration of Schrödinger's equation over time from time 0 to a later time t gives the discrete time evolution of the quantum state from time 0 to t; the fact that the Hamiltonian is Hermitian can be shown to be equivalent to the familiar fact that the discrete time evolution is unitary. When the Hamiltonian is independent of time, the solution of Schrödinger's equation is easy: one can verify that Schrödinger's equation is satisfied by

$$\left|\psi(t)\right\rangle = e^{iHt}\left|\psi(0)\right\rangle \tag{1.2}$$

(see section 2 for exponentiation of matrices). Moreover, the fact that H is Hermitian implies that the matrix e^{iHt} is unitary.

From the physicist's point of view, not every Hamiltonian can be used in the above equation, since not every Hamiltonian can be applied on a physical system. The physically realistic Hamiltonians are those that are *local*, namely, involve only interactions between a small number of particles. More formally, such a Hamiltonian H can be written as the sum $H(t) = \sum_m H_m(t)$, where m is small and each $H_m(t)$ is a tensor product of some Hermitian matrix on a small number of qubits with identity on the rest.

An important question in physics is the following. We are given a system which is in some initial state $\left|\psi(0)\right\rangle$ at time $t = 0$, and we let the system evolve according to a time-dependent Hamiltonian $H(t)$ from time $t = 0$ to $t = T$. This means that we set $\left|\psi(0)\right\rangle$ as the initial conditions for Schrödinger's equation and set the Hamiltonian to be $H(t)$. Our goal is to solve the equation and find out the state of the system at time T.

Adiabatic evolution is a special case of the above question in which an elegant solution exists. In adiabatic evolution, one considers a parameterized path in the Hamiltonian domain, $H(s)$ for $s \in [0,1]$, which starts at some Hamiltonian $H(0) = H_{init}$ and ends at another Hamiltonian $H(1) = H_{final}$. We require that the ground-state (the eigenstate of lowest eigenvalue) of the Hamiltonian $H(s)$ is unique for all $s \in [0,1]$. To specify the adiabatic evolution, one picks the duration of the process, namely T. The system is initialized at time $t = 0$ in the ground-state of $H(0)$. The system then evolves by Schrödinger's equation from time 0 to time T, under the Hamiltonian $\tilde{H}(t) = H(t/T)$. The term adiabatic means that the Hamiltonian is modified infinitely slowly along the path; in other words, we take $T \mapsto \infty$. In this limit we are guaranteed by the celebrated *adiabatic theorem* [34, 39] that the final state will be equal to the ground-state of the final Hamiltonian H_{final}.

Here we are interested in finite processes. Taking T to be finite introduces an error in the final state: it is no longer exactly the ground-state of $H(T)$ but only close to it in Euclidean distance. How large should T be in order for the error to be small? Two parameters turn out to be important. Denote by $\Delta(H(s))$ the spectral gap of $H(s)$, namely the difference between the Hamiltonian's lowest eigenvalue and the next one. The first parameter is Δ, the minimal spectral gap of the time-dependent Hamiltonian $H(s)$, along the path from $H(0)$ to $H(1)$. The other parameter is related to how fast the Hamiltonian changes in time; we set η to be the maximal norm of the first derivative of H with respect to s: $\eta = \max_s \|\frac{dH}{ds}(s)\|$. It turns out that for the final state to be within ϵ Euclidean distance from the final ground-state, T should be

$$T = poly\left(\frac{\eta}{\Delta\epsilon}\right). \tag{1.3}$$

Different versions of the theorem derive different (small degree) polynomials in the above parameters [43, 8, 10]. In [8] the second derivative of the Hamiltonian with respect to s also plays a role. We learn from (1.3) that if we would like to consider processes with polynomially bounded T, we need η to be polynomially bounded, and Δ to be nonnegligible, namely, bounded from below by a function which is inverse polynomial in the size of the system.

The proof of the adiabatic theorem [39] is rather nontrivial and is beyond the scope of this paper. We refer the reader to [8] for an elementary proof of the theorem and for further references. A very rough intuition about the proof is derived by considering a fixed Hamiltonian H, and observing how the solution to Schrödinger's equation behaves when the initial state is an eigenstate of the Hamiltonian H, with eigenvalue Λ. In this case, the matrix e^{iHt} applied on the eigenstate simply multiplies it by a scalar $e^{i(\Lambda t \mathrm{mod} 2\pi)}$. This complex number, of absolute value 1, can be viewed as a vector in the complex plane, which rotates in time faster when Λ is larger and slower when Λ is smaller. Hence, for the ground-state it rotates the least. The fast rotations essentially cancel the contributions of the vectors with the higher eigenvalues due to interference effects.

Farhi et al. considered the possibility of using adiabatic quantum evolutions to solve NP-hard optimization problems. The idea of Farhi et al. was to find the (unique) minimum of a given function $f : \{0,1\}^n \to \{0,1\}$ as follows: H_{init} is chosen to be some generic Hamiltonian. H_{final} is chosen to be the *problem Hamiltonian*, namely a $2^n \times 2^n$ matrix which has the values of f on its diagonal and zero everywhere else. The system is then initialized in the ground-state of H_{init} and evolves adiabatically on the convex line $H(s) = (1-s)H_{init} + sH_{final}$. By the adiabatic theorem, if the minimal spectral gap is lower bounded by some inverse polynomial, then T can be taken to be polynomially bounded, and the final state would be sufficiently close to the ground-state of H_{final} which is exactly the sought after minimum of f. Despite initial optimistic numerical results [20, 15, 21], there is now strong evidence that the spectral gap along the Hamiltonian path for the NP-hard problems considered is exponentially small [18, 19, 43].

The general model of adiabatic computation (see, e.g., [4]) does not require the final Hamiltonian to be diagonal as above, but it does require the Hamiltonians to be local. We now know that this model is in fact equivalent in computational power to the standard quantum circuit model. The fact that adiabatic computations with local Hamiltonians can be simulated efficiently by the standard model was shown in [22, 18]. The other direction, showing that any standard quantum computation can be simulated efficiently in the adiabatic model with local Hamiltonians, was recently shown by Aharonov et al. [4].[2] Adiabatic computation is thus a quantum algorithmic framework equivalent to the standard quantum computation model, in which computation is thought of as the process of generating (ground-) states. It is thus natural to draw intuition from it when attempting to design frameworks for quantum state generation.

1.2.2. An adiabatic framework for quantum state generation. We would now like to define a paradigm for quantum state generation that is based, in spirit, on similar ideas used in adiabatic computation. Our goal is to develop a tool that can be used when designing quantum algorithms that generate complicated quantum states. We therefore generalize adiabatic computation as much as we can, while maintaining its basic structure. First, we allow the path in the Hamiltonian domain to be a general path (with mild conditions such as smoothness). This is different from the choice often made in adiabatic computation literature, that the path be a straight line. Second, and very importantly, we relax the requirement that the Hamiltonians are local and require only that the Hamiltonians are *simulatable*. This means that the time evolution of the system governed by the Hamiltonian, namely the unitary matrix e^{iHt}, can be approximated by a *quantum circuit* to within any polynomial accuracy (for a rigorous definition see Definition 4.1). An *adiabatic state generator* is thus a specification of such a nicely behaved path of simulatable Hamiltonians in the Hamiltonians domain. The running time of the adiabatic state generator is taken to be exactly the time required for the adiabatic evolution to succeed, roughly given by (1.3) (again, for the exact condition see section 4).

We need to show that the existence of an adiabatic state generator implies the existence of a corresponding quantum state generator of the final state of the adiabatic state generator.

THEOREM 1.2 (informal). *Let A be an adiabatic state generator, initiated by a quantum state $|\psi(0)\rangle$, with a final state $|\psi(T)\rangle$, with a polynomially bounded time T. Then, if there exists an efficient quantum algorithm that generates $|\psi(0)\rangle$, then there exists an efficient quantum algorithm that generates $|\psi(T)\rangle$.*

This result means that one can use the framework of ASG as a quantum *reduction* from a (presumably, difficult to generate) quantum state to another quantum state (which is presumably easier to generate).

To prove the theorem we need to show how to simulate the adiabatic state generator efficiently using a quantum circuit, which is not too difficult, and we also need to show that the final state is indeed close to the ground-state of the final Hamiltonian. The second claim follows immediately from the adiabatic theorem. This provides a natural easy proof of Theorem 1.2, based on the adiabatic theorem. In this paper we prefer to avoid the use of the adiabatic theorem and instead provide an elementary proof which uses only the much simpler Zeno effect [42]. We now sketch the idea.

The Zeno effect considers the following situation. We start at some vector v_0 and apply a sequence of M projective measurements, each in a very close basis to the previous one. More precisely, if the jth measurement is in a basis which includes some vector v_j, we require the vectors v_j to be *slowly varying*; i.e., v_j is close to v_{j-1}. In this case, the Zeno effect states that with very high probability, after M measurements, we end up very close to v_M (even though v_0 and v_M might be very far away from each other). The Zeno effect resembles adiabatic computation in its

[2] In fact, the seeds for the result of [4] were planted in the preliminary version of the current paper.

slowly varying nature, with the important difference that in adiabatic computation we have a sequence of Hamiltonians, while in the Zeno effect we have a sequence of *measurements*. Our proof uses the useful Hamiltonian-to-projection lemma (Lemma 1.2), which we prove in section 6.

We thank Manny Knill [36] for pointing out to us the similarity between adiabatic evolution and the Zeno effect, which lead to this proof. A similar connection was used independently in [14].

1.3. ASG and Markov chains (section 5). We now proceed to show how the ASG paradigm can be used to Qsample from the limiting distributions of various Markov chains. This is done by converting classical approximate counting algorithms, based on rapidly mixing Markov chains, to adiabatic state generators (for a background on Markov chains see section 5).

It is well known that a Markov chain is rapidly mixing iff the second eigenvalue gap, namely the difference between the largest and second largest eigenvalue (in absolute values) of the transition matrix M, is nonnegligible [7]. This clearly bears resemblance to the adiabatic condition of a nonnegligible spectral gap and suggests looking at Hamiltonians of the form

$$H_M = I - M.$$

We use $I - M$ so that the spectrum is reversed; i.e., the largest eigenvector of M becomes the smallest of $H_M = I - H$, and the second eigenvalue gap of M turns into the spectral gap of H_M. H_M defined this way is a Hamiltonian if M is symmetric; if M is not symmetric but is a reversible Markov chain [38], we can still define the Hamiltonian corresponding to it (see section 5).

The first question is whether the resulting Hamiltonian can be used in our ASG framework. In other words, when is the Hamiltonian arising from a Markov chain simulatable? To this end we prove in section 6 a general lemma, called *the sparse Hamiltonian lemma*, which provides a very general condition for a Hamiltonian to be simulatable. Essentially, the condition is that the Hamiltonian be a sparse matrix. Based on this lemma, we show that for a (very natural) class of Markov chains, which we call *strongly samplable*, the Hamiltonian arising from the Markov chain is simulatable and can be used in the ASG paradigm.

In ASG one is interested not in a single Hamiltonian but in a path in the Hamiltonian domain. We recall that many approximate counting algorithms [33] use a sequence of Markov chains. Usually one starts with a simple Markov chain and slowly varies it until it gets close to a desired Markov chain. A notable example is the recent algorithm for approximating the permanent [32]. We show that such approximate counting algorithms naturally translate to adiabatic state generators. More precisely, but still informally, we have the following theorem.

Theorem 1.3 (informal). *Let A be an efficient randomized algorithm to approximately count a set Ω, possibly with weights. Suppose A uses slowly varying Markov chains starting from a Markov chain with a simple limiting distribution. Then there is an efficient quantum algorithm Q that Qsamples the final limiting distribution over Ω.*

We summarize the correspondence between Markov chains and adiabatic computation in Figure 1.1. We stress that it is *not* the case that we are interested in a quantum speed-up for sampling various distributions. Rather, we are interested in the *coherent* quantum state generation of the classical distribution, namely, in the solution for the QS problem.

A Markov chain	$\Leftrightarrow$	A Hamiltonian
A strongly samplable Markov chain	$\Leftrightarrow$	A simulatable Hamiltonian
Slowly varying strongly samplable Markov chains	$\Leftrightarrow$	ASG

FIG. 1.1. *The correspondence between Markov chains and adiabatic computation.*

The proof of Theorem 1.3 uses another general tool, which we call the *jagged adiabatic path lemma*. This lemma shows how we can connect the sequence of Hamiltonians resulting from the sequence of Markov chains, into a continuous path, such that if two subsequent Hamiltonians in the sequence are not too far, and all Hamiltonians in the sequence have nonnegligible spectral gaps, then all Hamiltonians along the path have nonnegligible spectral gaps. We state and prove this theorem in section 6.

We exploit this paradigm to Qsample the set of all perfect matchings of a bipartite graph using the recent algorithm by Jerrum, Sinclair, and Vigoda [32]. Using the same ideas we can also Qsample the set of all linear extensions of partial orders using an algorithm by Bubley and Dyer [12], all lattice points in a convex body satisfying certain restrictions using the Applegate–Kannan technique [9], and many more states.

1.4. Basic tools (section 6). In this section we collect several claims and lemmas that are used in the proofs inside the paper. We separate them from the rest of the paper, since these results are of a general flavor, and we believe they might be useful in other work related to adiabatic state generators, adiabatic computation, and computation with Hamiltonians in general.

We denote by $\alpha(H)$ the unique ground-state of a Hamiltonian H. The first claim shows that two close Hamiltonians have close ground-states, as long as their spectral gaps are big enough.

CLAIM 1.1. *Let A, B be two Hamiltonians of equal dimensions such that $\|A - B\| \leq \eta$. Moreover, assume that A, B have spectral gaps bounded from below: $\Delta(A), \Delta(B) \geq \Delta$. Then $|\langle\alpha(A)|\alpha(B)\rangle| \geq 1 - \frac{4\eta^2}{\Delta^2}$.*

The norm we use is the spectral norm, also called the operator norm (see section 2). The next basic but useful claim provides a lower bound on the spectral gap of a convex combination of two projections. For a vector $|\alpha\rangle$, the Hamiltonian $\Pi_\alpha = I - |\alpha\rangle\langle\alpha|$ is the projection onto the subspace orthogonal to α.

CLAIM 1.2. *Let $|\alpha\rangle, |\beta\rangle$ be two vectors in some Hilbert space. For any convex combination $H_\eta = (1-\eta)\Pi_\alpha + \eta\Pi_\beta,\ \eta \in [0,1]$, we have $\Delta(H_\eta) \geq |\langle\alpha|\beta\rangle|$.*

Both proofs use simple algebra.

Next, we prove the Hamiltonian-to-measurement lemma, which does the following. We are given a simulatable Hamiltonian with nonnegligible spectral gap. We design an efficient quantum circuit which essentially simulates a measurement in a basis which contains the ground-state of the given Hamiltonian.

LEMMA 1.1 (Hamiltonian-to-measurement lemma). *Assume H is a simulatable Hamiltonian on n qubits. For any constant d, there exists a poly$(n, \frac{1}{\Delta(H)})$-size circuit O_H which takes $|\alpha(H)\rangle$ to $|\alpha(H)\rangle \otimes |\gamma\rangle$, and for any eigenstate of H $|\alpha^\perp\rangle$ orthogonal to the ground-state, $O_H|\alpha^\perp\rangle = |\alpha^\perp\rangle \otimes |\beta(\alpha^\perp)\rangle$, where $|\langle\gamma|\beta(\alpha^\perp)\rangle| \leq O(n^{-d})$.*

The next lemma achieves a related task. It shows that if H is a simulatable Hamiltonian with nonnegligible spectral gap, then the Hamiltonian $\Pi_{\alpha(H)}$, which is the projection on the subspace orthogonal to the ground-state of H, is also simulatable.

Lemma 1.2 (Hamiltonian-to-projection lemma). *Assume H is a simulatable Hamiltonian on n qubits, with nonnegligible spectral gap $\Delta(H) \geq 1/n^c$ for some constant $c > 0$ and with a known ground-value. Then the Hamiltonian $\Pi_{\alpha(H)}$ is simulatable.*

The proof of both lemmas is a simple application of Kitaev's phase estimation algorithm [35].

The next lemma we prove allows connecting a sequence of Hamiltonians with not too far away ground-states into one adiabatic path.

Lemma 1.3 (the jagged adiabatic path lemma). *Let $\{H_j\}_{j=1}^{T=poly(n)}$ be a sequence of bounded norm, simulatable Hamiltonians on n qubits, with nonnegligible spectral gaps, $\Delta(H_j) \geq n^{-c}$, and with known ground-values, such that the inner product between the unique ground-states $\alpha(H_j), \alpha(H_{j+1})$ is at least n^{-c} for all j. Then there exists an adiabatic state generator with $\alpha(H_0)$ as its initial state and $\alpha(H_T)$ as its final state. In particular there exists an efficient quantum algorithm that takes $\alpha(H_0)$ to within arbitrarily small distance from $\alpha(H_T)$.*

The proof of this lemma is fairly simple, with one trick required. Our first attempt would be to consider the (jagged) path in the Hamiltonian domain that connects one Hamiltonian in the sequence to the next by a straight line. The main point is to show that the spectral gap along the lines is not too small. In fact, this does not hold in the general case (see section 6.4), but if instead of connecting the Hamiltonians we actually connect the projections $\Pi_{\alpha(H)}$'s, we can then use Claim 1.2 to prove that the convex combination of these projections has a nonnegligible spectral gap.

Finally, we ask which Hamiltonians can be used in the ASG framework, namely, which Hamiltonians are simulatable. We provide a very general condition under which we can simulate the Hamiltonian using a quantum circuit. We say that H on n qubits is *sparse* if it has at most polynomially many nonzero elements in each row (and column, as it is Hermitian). We say it is *explicit* if there exists an efficient classical algorithm that given an index of a row, j, outputs an *approximation* of *all* nonzero elements in the jth row of the Hamiltonian.

Definition 1.2 (an explicit matrix). *We say an $N \times N$ matrix A is explicit if for every $d > 0$ there exists an algorithm that on input $j \in N$ outputs an approximation of all nonzero elements in the jth row of A to within n^{-d} accuracy and whose running time is polynomial in $\log(N)$.*

Lemma 1.4 (the sparse Hamiltonian lemma). *If H is an explicit and sparse Hamiltonian on n qubits and $\|H\| \leq poly(n)$, then H is simulatable.*

We note that a local Hamiltonian is in particular sparse and explicit, but sparse and explicit Hamiltonians are not necessarily local.

The main idea of the proof is to write H as a sum of polynomially many bounded norm Hamiltonians H_m which are all block diagonal (in a combinatorial sense) and such that the size of the blocks in each matrix is at most 2×2. This is done using some combinatorial and number theoretical tricks. We then show that each Hamiltonian H_m is simulatable. To simulate the sum of the Hamiltonians we use standard techniques (namely, Trotter's formula—see section 4.1.1).

1.5. Conclusions. This paper sets the grounds for the general study of quantum state generation, using the paradigm of ASG. This direction points at interesting and intriguing connections between quantum computation and many different areas: the complexity class SZK and its complete problem SD [44], the notion of adiabatic evolution [34], the study of rapidly mixing Markov chains using spectral gaps [38], quantum random walks [13], and the study of ground-states and spectral gaps of Hamiltonians in physics. Hopefully, techniques from these areas can be borrowed to give more tools for ASG. Notably, the study of spectral gaps of Hamiltonians in physics is a lively area with various recently developed techniques (see [46] and the references therein).

It seems that a much deeper understanding of the adiabatic paradigm is required in order to solve the most interesting open question, namely to design interesting new quantum algorithms. As an intermediate task, it would be interesting to present known quantum algorithms, e.g., Shor's discrete log algorithm, or the quadratic residuosity algorithm, in the ASG paradigm in an insightful way.

1.6. Related work. The definition of ASG uses adiabatic evolutions along general paths in the Hamiltonian domain, and not just straight lines. Such adiabatic evolutions were also studied in [14].

The connection between adiabatic evolution, the Zeno effect, and measurements, which we use in our work, was observed before. We thank Manny Knill for pointing this out to us [36]. These connections were also considered, in a recent independent work, in [14].

We believe that the sparse Hamiltonian lemma might have other interesting implications, e.g., in the context of Hamiltonian based quantum random walks on graphs [16, 23, 13]. For example, Childs et al. [13] use quantum random walks to provide an exponential algorithmic speed-up over any possible classical algorithm for a certain graph reachability task. To do this, they define certain Hamiltonians and use a method of coloring to show that these Hamiltonians can be simulated efficiently by a quantum circuit. The sparse Hamiltonian lemma immediately implies that the Hamiltonians used in [13] are simulatable.

After the publication of the preliminary version of this article [3], the ideas presented in it were used to make progress in two different directions.

The first direction is the characterization of the computational complexity of the problem of approximating the shortest vector in a lattice up to $\sqrt{n}$ ($\mathrm{GapSVP}_{\sqrt{n}}$). Our reduction of this problem to a QS problem (section 3) was used in [5] to show that the problem lies in quantum NP. This gave the first nontrivial quantum complexity upper bound on a lattice problem. A following paper [6] improved this result and proved that $\mathrm{GapSVP}_{\sqrt{n}}$ lies in $\mathrm{NP} \cap co\mathrm{NP}$. Interestingly, this result initiated from an attempt to design an ASG algorithm for the relevant QS problem.

The second place where these results inspired further progress is in the study of adiabatic computation, where an important open question was the clarification of the computational power of the model. Our results raised the question of how powerful quantum adiabatic algorithms are and gave tools to prove some preliminary results about their universality [2]. These results were recently improved in [4] to show that the model of adiabatic computation using *local* Hamiltonians is equivalent to standard quantum computation.

1.7. Paper organization. The paper is organized as follows. We give some notation and general mathematical preliminaries in section 2. Background related to particular parts of the paper is given at the beginning of each section.

In section 3 we show that the QS problem is sufficient for solving all the languages in SZK, and we also discuss whether it is equivalent to solving SZK. The specific examples of the Qsampling instances associated with discrete log, quadratic residuosity, and a lattice problem are given in Appendix A. We define adiabatic quantum state generation in section 4. We also show (using measurements and the Zeno effect) that adiabatic state generators can be simulated by quantum circuits. In section 5 we show the connection to Markov chains, and prove that a host of approximating counting algorithms can be translated into adiabatic state generators, generating many interesting coherent states. Finally, in section 6, we prove several lemmas that serve as basic building blocks for our previous results, including the sparse Hamiltonian lemma, the jagged adiabatic path lemma, and the Hamiltonian-to-projection and Hamiltonian-to-measurement lemmas.

2. Preliminaries.

We assume the reader is familiar with the basic terminology of quantum computation: qubits, pure states, Hilbert space, density matrix, the class BQP, etc. For background on these notions, please consult [40]. We now give some preliminaries relevant for the entire paper. More specific preliminaries are given at the beginning of each section.

2.1. Distances between distributions: Fidelity and variational distance.

For two classical distributions $\{p(x)\}, \{q(x)\}$ we define their *variational distance* and their *fidelity* (this measure is known by many other names as well) to be, respectively,

$$|p-q| = \frac{1}{2}\sum_x |p(x)-q(x)|,$$
$$F(p,q) = \sum_x \sqrt{p(x)q(x)}.$$

The following fact is very useful.

Fact 2.1 (see [40]).

$$1 - F(p,q) \leq |p-q| \leq \sqrt{1-F(p,q)^2}$$

or, equivalently,

$$1 - |p-q| \leq F(p,q) \leq \sqrt{1-|p-q|^2}.$$

A distribution D is flat if for every z_1 and Z_2 for which $D(z_1), D(z_2) > 0$ we have $D(z_1) = D(z_2)$; i.e., D is uniform over all elements in its support.

2.2. Norms on matrices: Trace norm and operator norm.

The trace norm of a Hermitian matrix H with eigenvalues $\lambda_1, \ldots, \lambda_n$ is $\|H\|_{tr} = \sum |\lambda_i|$. Note that the trace norm of a density matrix is 1. The trace norm satisfies that $\|A \otimes B\|_{tr} = \|A\|_{tr}\|B\|_{tr}$.

The operator norm of a linear transformation T induced by the l_2 norm is called the *spectral norm* and is defined by

$$\|T\| = \max_{\psi \neq 0} \frac{|T\psi|}{|\psi|}.$$

The operator norm satisfies that for any two matrices, $\|AB\| \leq \|A\| \cdot \|B\|$.

If T is Hermitian or unitary (in general, if T is normal, namely, commutes with its adjoint), then $\|T\|$ equals the largest absolute value of its eigenvalues. Hence, if U is unitary, $\|U\| = 1$.

For any two unitary matrices A and B and any integer k, $\|A^k - B^k\| \leq k\|A-B\|$. This follows from the fact that $\|AB - CD\| \leq \|AB - CB\| + \|CB - CD\|$, which for unitary matrices is $\leq \|A-C\| + \|B-D\|$.

Finally, for a general $N \times N$ matrix $A = (a_{i,j})$ we have $\|A\|_\infty \leq \|A\| \leq N^2\|A\|_\infty$, where $\|A\|_\infty = \max_{i,j}|a_{i,j}|$.

2.3. Distances between density matrices. The variational distance and the fidelity can be generalized to density matrices, and Fact 2.1 also holds for density matrices (see [40]).

The generalization of the variational distance is the trace norm of the difference between the two matrices. It is a well-known fact that the two output distributions resulting from applying the same quantum measurement on two different density matrices, ρ_1 andρ_2, can have variational distance at most $\frac{1}{2}\|\rho_1 - \rho_2\|_{tr}$. For more details we refer the reader to [40, section 9.2].

In this paper we need only define fidelity for pure states. For two vectors ϕ_1, ϕ_2 in some Hilbert space, the fidelity is simply the absolute value of their inner product: $F(\phi_1, \phi_2) = |\langle\phi_1|\phi_2\rangle|$.

2.4. Power of a matrix. If M is a Hermitian matrix, then it has an orthonormal basis of eigenvectors $\{v_i\}$ with real eigenvalues $\{\lambda_i\}$. For a function $f : \mathbb{C} \to \mathbb{C}$, $f(M)$ is the linear transformation that has $\{v_i\}$ as an orthonormal basis of eigenvectors with eigenvalues $\{f(\lambda_i)\}$. In particular, this defines e^M.

2.5. Hamiltonian terminology. The set of Hamiltonians is the set of Hermitian matrices. The ground-state of a Hamiltonian H is the eigenstate with the smallest eigenvalue, and we denote it by $\alpha(H)$. The spectral gap of a Hamiltonian H is the difference between the smallest and second to smallest eigenvalues, and we denote it by $\Delta(H)$. If H is Hermitian, then its eigenvalues are real, and hence e^{-iH} is unitary.

3. Quantum state generation and SZK. In this section we connect the QS problem to the class SZK. We start with some background about SZK. We refer the interested reader to Vadhan's thesis [47] and to Sahai and Vadhan [44] for rigorous definitions, a discussion of their subtleties, and other results known about this elegant class. We then proceed to prove Theorem 1.1, and in Appendix A we provide explicit examples of interesting QS instances. We also prove that the task of QS for graph isomorphism is not harder than solving the graph isomorphism problem itself, and that if QS can be done with no error, then the graph isomorphism problem is in RQP$\bigcap$*co*RQP.

3.1. Background on SZK.

3.1.1. Interactive proofs. A pair $\Pi = (\Pi_{Yes}, \Pi_{No})$ is a promise problem if $\Pi_{Yes} \subseteq \{0,1\}^*$, $\Pi_{No} \subseteq \{0,1\}^*$, and $\Pi_{Yes} \cap \Pi_{No} = \emptyset$. We look at Π_{Yes} as the set of all *yes* instances and Π_{No} as the set of all *no* instances, and we do not care about all other inputs. If every $x \in \{0,1\}^*$ is in $\Pi_{Yes} \cup \Pi_{No}$, we call Π a language.

An interactive proof is a protocol in which a prover P tries to convince a verifier V of some fact through an exchange of messages. Formally, the prover and the verifier are described by probabilistic Turing machines which act on their private working spaces plus some interaction domain. The verifier is required to be polynomial time, and the prover is assumed to be all powerful. The interactive proof is denoted by (P, V).

We say that a promise problem Π has an interactive proof with soundness error ϵ_s and completeness error ϵ_c if there exist V, P such that we have the following:

- If $x \in \Pi_{Yes}$, V accepts with probability at least $1 - \epsilon_c$.
- If $x \in \Pi_{No}$, then *for every* prover P^*, V accepts with probability at most ϵ_s.

The class NP consists of one-message interactive proofs with $\epsilon_c, \epsilon_s = 0$.

When an interactive proof system (P, V) for a promise problem Π is run on an input x, it produces a distribution over *transcripts* that contains the conversation between the prover and the verifier; i.e., each possible transcript appears with some probability (depending on the random coin tosses of the prover and the verifier).

3.1.2. SZK. The class SZK consists of promise problems for which there are interactive proofs which exhibit the following remarkable property: for $x \in \Pi_{yes}$, the verifier learns (almost) nothing from the interaction with the prover P, other than the fact that x is a *yes* instance. It is remarkable that such proof systems in fact exist. This is captured mathematically by the concept of *simulation* as follows.

An interactive proof system (P, V) for a promise problem Π is said to be an honest verifier SZK, if there exists a probabilistic polynomial time *simulator* S that for every $x \in \Pi_{Yes}$ produces a distribution on transcripts that is close (in the variational distance sense; see section 2.1) to the distribution on transcripts that V and P would produce in their interaction. Note that the simulator has no access to the prover, and that we require only the simulator to produce a good distribution on inputs in Π_{Yes}, since for *no* instances there is no proof to learn anyway.

One might wonder whether it is possible for the verifier to deviate from the protocol (namely, to *cheat*) and by this to get information from an honest prover. Indeed, there are honest verifier SZK proofs which are not secure against a cheating verifier. However, it was shown in [27] that whenever there exists an honest verifier SZK proof, then there is also an interactive proof that is also secure against dishonest verifiers. By this we mean that a simulator also exists for verifiers that deviate from the protocol.

We denote by SZK the class of all promise problems which have interactive proof systems which are statistically zero knowledge against an honest (or, equivalently, a general) verifier. It is known that BPP $\subseteq$ SZK $\subseteq$ AM $\cap$ *co*AM [24, 11, 47] and that SZK is closed under complement [41, 47]. It follows that SZK does not contain any NP-complete language unless the polynomial time hierarchy collapses.

3.1.3. A complete problem for SZK. Sahai and Vadhan [44] found a natural complete problem for SZK. One nice thing about the problem is that it does not mention interactive proofs in any explicit or implicit way. We define the complete problem for SZK.

Definition 3.1 (statistical difference ($\mathrm{SD}_{\alpha,\beta}$)).

Input: *Two classical circuits C_0, C_1 with m Boolean outputs.*

Promise:

- *Yes:* $|D_{C_0} - D_{C_1}| \geq \alpha$.
- *No:* $|D_{C_0} - D_{C_1}| \leq \beta$.

Sahai and Vadhan [44] and Vadhan [47] show that for any two constants $0 < \beta < \alpha < 1$ such that $\alpha^2 > \beta$, $\mathrm{SD}_{\alpha,\beta}$ is complete for SZK.

3.2. A reduction from SZK to QS. We are now ready to prove Theorem 1.1. We first describe a very simple, standard building block in quantum computation, called the SWAP test.

Definition 3.2 (the SWAP test). *The algorithm operates on three quantum registers: A is a one qubit register, and B and C are two registers with the same number of qubits. The algorithm applies a Hadamard on the first qubit, then conditioned on the first control qubit swaps between the second and third registers, and, finally, applies a Hadamard on the control qubit and measures it.*

By a direct calculation, we have the following claim.

CLAIM 3.1. *Let v_1, v_2 be two vectors in the same Hilbert space. If the SWAP test is applied on $|0, v_1, v_2\rangle$, then the outcome of the SWAP test is* 0 *with probability* $\frac{1+|\langle v_1|v_2\rangle|^2}{2}$ *and* 1 *with probability* $\frac{1-|\langle v_1|v_2\rangle|^2}{2}$.

We now proceed to prove Theorem 1.1.

Proof of Theorem 1.1. We assume that QS is in BQP. It is enough to show that $\mathrm{SD}_{0.9,0.1}$, which is an SZK-complete problem, is in BQP.

Indeed, let C_0, C_1 be an input to $\mathrm{SD}_{0.9,0.1}$. By our assumption there is an efficient quantum algorithm that can generate states ρ_0, ρ_1 such that $\| \rho_i - |C_i\rangle \langle C_i| \|_{tr} \le \delta$ for $i = 0, 1$ and $\delta = 10^{-5}$. We can therefore apply the SWAP test on the two states ρ_0, ρ_1 efficiently. We now claim the test results in the outcome 1 with probability greater than 0.4 in case $|D_{C_0} - D_{C_1}| \ge 0.9$ and with probability smaller than 0.1 in case $|D_{C_0} - D_{C_1}| \le 0.1$.

The BQP algorithm follows from this claim easily: To achieve error ϵ, simply repeat the SWAP test $O(\log(\frac{1}{\epsilon}))$ times, generating the states each time from scratch. Then count the number of outcomes 1. If it is more than 0.25 of the tests, accept (the distributions are far); otherwise, reject (the distributions are close).

To prove the claim, we first write down the probability for 1 in the ideal case, in which the BQP algorithm outputs $|C_i\rangle$ exactly. We have

$$\langle C_0|C_1\rangle = \sum_{z\in\{0,1\}^m} \sqrt{D_{C_0}(z)D_{C_1}(z)} \;=\; F(D_{C_0}, D_{C_1}).$$

Claim 3.1 implies, therefore, that the SWAP test on the state $|0, C_0, C_1\rangle$ results in 1 with probability $\frac{1-F(D_{C_0},D_{C_1})^2}{2}$.

In fact, the state ρ_i is within δ trace distance from $|C_i\rangle$. This implies that the actual state on which we apply the swap test is $\rho_1 \otimes \rho_2$, which is 2δ-close in the trace norm to that of the pure state $|0, C_0, C_1\rangle$ (see section 2.2). By section 2.3, the variational distance between the distributions resulting from applying the SWAP test in the two cases is δ. This implies that the probability for 1 in the actual SWAP test is $\frac{1-F(D_{C_0},D_{C_1})^2}{2} \pm \delta$.

Using Fact 2.1, we have the following:

- If $|D_{C_0} - D_{C_1}| \ge \alpha$, we measure 1 with probability $\frac{1-F(D_{C_0},D_{C_1})^2}{2} \pm \delta \ge \frac{|D_{C_0}-D_{C_1}|^2}{2} - \delta \ge \frac{\alpha^2-2\delta}{2}$.
- If $|D_{C_0} - D_{C_1}| \le \beta$, we measure 1 with probability $\frac{1-F(D_{C_0},D_{C_1})^2}{2} \pm \delta \le \frac{2|D_{C_0}-D_{C_1}|-|D_{C_0}-D_{C_1}|^2}{2} + \delta \le \frac{2\beta-\beta^2+2\delta}{2}$.

Setting $\alpha = 0.9$ and $\beta = 0.1$, we get the desired results. □

3.3. Perfect QS and one-sided-error quantum algorithms. One might hope that if one could perfectly solve QS (i.e., $\mathrm{QS}_{\delta=0} \in$ BQP), then SZK $\subseteq$ RQP, where RQP is the one-sided variant of BQP. This, however, does not follow, because $\mathrm{SD}_{\alpha,\beta}$ is known to be SZK complete only when $\beta > 0$ and $\alpha < 1$. Instead, we can prove a weaker version of this general result, concerning the class honest verifier *perfect* zero knowledge (HVPZK), where the simulator can *exactly* simulate the transcripts distribution. This class contains the graph isomorphism and the graph nonisomorphism problems.

LEMMA 3.1. *If $QS_{\delta=0} \in BQP$, then coHVPZK $\subseteq$ RQP.*

Proof. The proof uses the fact that $\mathrm{SD}_{0.5,0}$ is complete for coHVPZK [47]. It is enough to show that $\mathrm{SD}_{0.5,0}$ is in RQP. Indeed, let C_0, C_1 be an input to $\mathrm{SD}_{0.5,0}$. By our assumption we can generate the superpositions $|C_i\rangle$ for $i = 0, 1$. The quantum algorithm proceeds as in the proof of Theorem 1.1 and accepts iff the result of one of the measurements is 1. For yes instances, $|D_{C_0} - D_{C_1}| \geq \alpha = 0.5$, and so we measure 1 with probability at least 0.12. For no instances, we never measure 1. Hence we get an RQP algorithm. □

As both graph isomorphism and graph nonisomorphism are in HVPZK, we get the following corollary.

Corollary 3.1. *If $QS_{\delta=0} \in BQP$, then $GI \in RQP \bigcap coRQP$.*

3.4. Specific examples. We saw that every problem L in SZK reduces to a pair of circuits $C_{L,0}, C_{L,1}$ such that if we can Qsample $|C_{L,i}\rangle$, we can solve L in quantum polynomial time. Unfortunately, we do not know how to solve the QS problem in general. We would like to specify explicitly interesting instances of the QS problem, associated with specific problems in SZK.

In theory, such an instance can be derived from the SZK proof of the promise problem in the following way. For every problem L in SZK, one can follow the reduction from L to $\mathrm{SD}_{0.9,0.1}$ (guaranteed by the SZK-completeness of $\mathrm{SD}_{0.9,0.1}$ [44]) and find two specific circuits $C_{L,i}$ corresponding to L. Qsampling from these circuits would be sufficient for solving L in quantum polynomial time. In practice, however, specifying the circuits is often not easy, as the reduction to $\mathrm{SD}_{0.9,0.1}$ is quite involved.

However, it is often possible to infer two such circuits $C_{L,i}$ directly from the zero-knowledge proof of L. We already saw in the introduction such a specific example for the graph isomorphism problem. In Appendix A we give three more examples of particular interest for quantum algorithms: discrete log, quadratic residuosity, and a gap version of closest vector in a lattice.

3.5. Is solving QS equivalent to solving SZK? We saw that QS $\in$ BQP implies that SZK $\subseteq$ BQP. A natural question is whether the QS problem is *equivalent* to solving SZK or *strictly harder*.

We start with the simplest case. Say L is a (promise) problem such that

- for any x, (L, x) can be efficiently reduced to solving the instance $|C_x\rangle$ of QS,
- C_x is one-to-one on its inputs, and
- there exists a procedure in BQP that using L as an oracle can invert C_x; i.e., given z, it computes a y such that $C_x(y) = z$.

For example, the discrete log problem gives rise to such a situation (see the problem DLP and the circuit C given in Appendix A). We claim the following.

Claim 3.2. *If L and C are as above, then $L \in BQP$ iff C is Qsamplable.*

Proof. We already know that (L, x) can be reduced to solving the instance $|C_x\rangle$ of QS. We show the other direction. Assume $L \in$ BQP. Fix some input x. Then, given $|y\rangle$, we can compute $|y, C_x(y)\rangle$ (because the circuit C_x is given to us), and, given $|C_x(y)\rangle$, we can compute $|C_x(y), y\rangle$ (because $L \in$ BQP and we assume we can invert C_x using L).[3] It then follows that there exists an efficient procedure that *replaces* $|y\rangle$ with $|C_x(y)\rangle$ (by undoing the computation). In particular we can build the superposition $\sum_y |y\rangle$ and transform it into the superposition $|C_x\rangle = \sum |C_x(y)\rangle$. □

Next, we consider the case where C_x is not one-to-one but rather *regular*; i.e., the distribution $D_C = D_{C_x}$ it induces is flat. Let us further assume that we have an

[3] In fact, we approximate only the state, since we run a BQP algorithm for the inversion procedure, and this algorithm may err.

efficient way to complete C_x to a one-to-one function. Formally, say L is a (promise) problem such that

- for any x, (L, x) can be efficiently reduced to solving the instance $|C_x\rangle$ of QS, and C_x is a circuit computing a function $C_x : \{0,1\}^n \to \Lambda_C$ for some domain Λ_C,
- there exists an efficient function $f_x : \{0,1\}^n \to \Lambda_f$, for some domain Λ_f, such that $C_x \otimes f_x : \{0,1\}^n \to \Lambda_C \times \Lambda_f$ (defined by $(C_x \otimes f_x)(y) = (C_x(y), f_x(y))$) is *one-to-one* and *onto*, and
- there exists an efficient procedure that using L as an oracle can invert $C_x \otimes f_x$; i.e., given z, it computes a y such that $(C_x \otimes f_x)(y) = z$.

We claim the following.

CLAIM 3.3. *If L and C are as above, then $L \in BQP$ iff C is Qsamplable.*

Proof. As before, we can create the state $\phi = \sum |C_x(y), f_x(y)\rangle$. As $C_x \otimes f_x$ is one-to-one and onto $\Lambda_C \times \Lambda_f$, we have that $\phi = \sum_{z \in \Lambda_C} |z\rangle \otimes \sum_{v \in \Lambda_f} |v\rangle$. Hence ϕ is in fact a product state, and we get the state $|C_x\rangle$ by just ignoring the second register. □

Graph isomorphism is an example to such a situation, as we now show. A key fact that we use is that there exists a deterministic search-to-decision reduction for graph isomorphism (see, e.g., [37, section 1.2]). Given any two isomorphic graphs G and G', the reduction R gives a permutation $\pi = R(G, G') \in \mathcal{S}_n$ such that $\pi(G) = G'$, where n is the number of vertices in G, and $\mathcal{S}_n$ is the set of all permutations on n elements.

Then the circuit $C_G : \mathcal{S}_n \to \mathcal{S}_n(G)$ gets $\pi \in \mathcal{S}_n$ as an input and outputs the permuted graph $\pi(G)$, $C_G(\pi) = \pi(G)$. The function $f_G : \mathcal{S}_n \to Aut(G)$ is defined by $f(\pi) = (R(G, \pi(G)))^{-1} \cdot \pi$ (where the product is in $\mathcal{S}_n$). We leave it to the reader to show that $f(\pi) \in Aut(G)$, that $C \otimes f$ is one-to-one and onto, and that the above three conditions are satisfied. Then we have the following lemma.

LEMMA 3.2. *GI $\in$ BQP iff $|\alpha_G\rangle = \sum_{\sigma \in S_n} |\sigma(G)\rangle$ can be generated in BQP.*

It is tempting to try extending the above approach in order to prove Lemma 3.2 for the SZK-complete problem $\mathrm{SD}_{\alpha,\beta}$. However, we face the following problems:

- C_x might not be regular; i.e., different elements $C_x(y)$ might have a different number of preimages.
- Even worse, even if we assume C_x is regular, in fact even if C_x is a permutation, it might be possible that C_x is hard to invert (and then C_x is a one-way function), and it is possible that it is hard to invert even given access to an oracle solving L.

  Thus, for this approach to work, it must be true that if $L =$ SD (and therefore also the whole of SZK) is easy (classically or quantumly), then there are no one-way functions in the quantum model. We note that the question of whether it is possible that SZK = BPP but yet one-way functions exist (in the classical model) is a major open problem (see [47, Open problem 4.8.10]).

We therefore do not know if, in general, solving QS in BQP is equivalent to solving SZK in BQP, and we leave it as an open problem.

4. The ASG paradigm.

In this section we define the paradigm of ASG. At the end of the section we formally state and prove Theorem 1.2, which states that any adiabatic state generator can be simulated efficiently by a quantum circuit. This is done using the Zeno effect. As mentioned before, our proof does not rely on the adiabatic theorem. We start with some background on the Trotter formula, which we need for our proofs in this section.

4.1. Preliminaries.

4.1.1. Trotter's formula. Consider the sum of two Hamiltonians A and B, $A+B$. We are interested in writing the unitary matrix $e^{i(A+B)t}$ in terms of e^{iAt} and e^{iBt}. If A and B commute, this is simple: we have $e^{i(A+B)t} = e^{iAt} \cdot e^{iBt}$. If the two matrices do not commute, Trotter's formula gives a way to do this:

$$\lim_{n\to\infty} (e^{iAt/n} e^{iBt/n})^n = e^{i(A+B)t}.$$

In other words, it says that if we interleave short executions of A and B, then in the limit we get an execution of $A+B$. For our purpose we need to quantify the error as a function of n, and for that we use the following variant from ([40, eq. 4.104]):

$$||e^{2\delta i(A+B)} - e^{\delta iA} e^{2\delta iB} e^{\delta iA}|| \leq O((\max\{||A||, ||B||\} \cdot \delta)^3). \tag{4.1}$$

We also need to deal with Hamiltonians of the form $H = \sum_m H_m$ that are sums of $m > 2$ Hamiltonians. We prove the following lemma (a very similar statement appears in [40, Exercise 4.50]).

Lemma 4.1. *Let H_m be Hermitian, $m = 1, \dots, M$, and let $H = \sum_{m=1}^M H_m$. Further, assume that for every $1 \leq k \leq \ell \leq M$ we have $\| \sum_{i=k}^{\ell} H_i \| \leq \Lambda$. Define*

$$U_\delta = [\ e^{\delta i H_1} \cdot e^{\delta i H_2} \cdot \dots \cdot e^{\delta i H_M}\ ] \cdot [\ e^{\delta i H_M} \cdot e^{\delta i H_{M-1}} \cdot \dots \cdot e^{\delta i H_1}\ ]. \tag{4.2}$$

Then $\|U_\delta - e^{2\delta i H}\| \leq O(M \cdot (\delta\Lambda)^3)$.

Proof. We prove by induction on M. The case $M = 2$ is (4.1). For the induction step, we notice that by (4.1)

$$||e^{2\delta i \sum_{i=1}^M H_i} - e^{\delta i H_1} e^{2\delta i \sum_{i=2}^M H_i} e^{\delta i H_1}|| \leq O((\delta\Lambda)^3).$$

Also, $U_\delta = e^{\delta i H_1} [e^{\delta i H_2} \cdot \dots \cdot e^{\delta i H_M}\ ] \cdot [\ e^{\delta i H_M} \cdot \dots \cdot e^{\delta i H_2}\ ] e^{\delta i H_1}$. Thus,

$$\begin{aligned} &||U_\delta - e^{2\delta i \sum_{i=1}^M H_i}|| \\ &\quad \leq ||[e^{\delta i H_2} \cdot \dots \cdot e^{\delta i H_M}\ ] \cdot [\ e^{\delta i H_M} \cdot \dots \cdot e^{\delta i H_2}\ ] - e^{2\delta i \sum_{i=2}^M H_i}|| + O((\delta\Lambda)^3), \end{aligned}$$

and by induction this is bounded by $O(M(\delta\Lambda)^3)$. □

Corollary 4.1. *Let H, H_m satisfy the conditions of Lemma 4.1. Then, for every $t > 4\delta$,*

$$\left\| U_\delta^{\lfloor \frac{t}{2\delta} \rfloor} - e^{-itH} \right\| \leq O(\Lambda \cdot \delta + M\Lambda^3 t \cdot \delta^2).$$

Notice that for every fixed t, M, and Λ, the error term goes down to zero with δ. In applications, we pick δ in such a way that the above error term is polynomially small. We now give the proof.

Proof.

$$\begin{aligned} \left\| U_\delta^{\lfloor \frac{t}{2\delta} \rfloor} - e^{-itH} \right\| &\leq \left\lfloor \frac{t}{2\delta} \right\rfloor \cdot \left\| U_\delta - e^{\frac{-it}{\lfloor \frac{t}{2\delta} \rfloor} H} \right\| \\ &\leq \frac{t}{2\delta} \cdot \left[\left\| U_\delta - e^{\frac{-it}{\frac{t}{2\delta}} H} \right\| + \left\| e^{\frac{-it}{\frac{t}{2\delta}} H} - e^{\frac{-it}{\lfloor \frac{t}{2\delta} \rfloor} H} \right\| \right]. \end{aligned}$$

The first term $\|U_\delta - e^{\frac{-it}{\frac{t}{2\delta}}H}\| = \|U_\delta - e^{-i2\delta H}\| \le O(M \cdot (\delta\Lambda)^3)$, by Lemma 4.1.

For the second term $\|e^{\frac{-it}{\frac{t}{2\delta}}H} - e^{\frac{-it}{\lfloor\frac{t}{2\delta}\rfloor}H}\|$, we notice that both matrices (and therefore also their difference) have the same eigenvector basis (that of H). As the norm is maximized at some eigenvector, $\|e^{\frac{-it}{\frac{t}{2\delta}}H} - e^{\frac{-it}{\lfloor\frac{t}{2\delta}\rfloor}H}\| = |e^{\frac{-it}{\frac{t}{2\delta}}\lambda} - e^{\frac{-it}{\lfloor\frac{t}{2\delta}\rfloor}\lambda}|$ for some eigenvalue λ with $|\lambda| \le \Lambda$ (because H has bounded norm). We now use the identities $|e^{-\theta i} - e^{-\theta' i}| = 2|\sin(\frac{\theta-\theta'}{2})| \le |\theta - \theta'|$. We see that the second term is bounded by $\frac{8\delta^2\lambda}{t}$.

Altogether,

$$\begin{aligned}\left\|U_\delta^{\lfloor\frac{t}{2\delta}\rfloor} - e^{-itH}\right\| &\le O\left(\frac{t}{\delta}\right) \cdot \left[(M \cdot (\delta\Lambda)^3) + \frac{\Lambda\delta^2}{t}\right] \\ &= O(M\Lambda^3 t\delta^2) + O(\Lambda\delta). \qquad \square\end{aligned}$$

4.2. Adiabatic quantum state generation. We now define our paradigm for *quantum state generation* inspired by the adiabatic theorem. As explained in the introduction, we would like to allow as much flexibility as possible and therefore allow any Hamiltonian which can be implemented efficiently by quantum circuits. We define the following.

Definition 4.1 (simulatable Hamiltonians). *We say that a Hamiltonian H on n qubits is simulatable if for every real value $t > 0$ and every accuracy $0 < \epsilon < 1$ the unitary transformation*

$$U(t) = e^{-iHt}$$

can be approximated by a quantum circuit of size $poly(n, t, 1/\epsilon)$ to within ϵ accuracy in the operator norm.

Corollary 4.1 implies that a local Hamiltonian is simulatable (but the other direction is not true). If H is simulatable, then by definition so is cH for any $0 \le c \le poly(n)$. It therefore follows by Trotter's formula that any convex combination of two simulatable, polynomially bounded norm Hamiltonians is simulatable. Also, if H is simulatable and U is a unitary matrix that can be efficiently applied by a quantum circuit, then $UHU^\dagger$ is also simulatable, because $e^{-itUHU^\dagger} = Ue^{-itH}U^\dagger$. We note that these rules cannot be applied unboundedly many times in a recursive way, because the simulation will then blow up. The interested reader is referred to [40, 13] for a more complete set of rules for simulating Hamiltonians.

We now describe an adiabatic path, which is an allowed path in the Hamiltonian space.

Definition 4.2 (adiabatic path). *A function H from $s \in [0,1]$ to the vector space of Hamiltonians on n qubits is an adiabatic path if*

- *$H(s)$ is continuous,*
- *$H(s)$ is differentiable, except for polynomially many points,*
- *for all s, $H(s)$ has a unique ground-state, and*
- *for all s, $H(s)$ is simulatable given s.*

Adiabatic quantum state generation is supposed to mimic the process of implementing Schrödinger's evolution along an adiabatic path, where the adiabatic condition holds.

In our case, we use simulatable Hamiltonians rather than local Hamiltonians. The time associated with ASG is defined using similar parameters to those used in the adiabatic theorem (as explained in the introduction). For an adiabatic path $H(s)$ we define

$$\eta(H(\cdot)) = \max_{s\in[0,1]\setminus\mathbf{D}} \left\| \frac{dH}{ds}(s) \right\| \text{ and}$$
$$\Delta(H(\cdot)) = \min_{s\in[0,1]} \Delta(H(s)),$$

where in the above, D is the set of at most polynomially many points where the derivative is not defined.

Definition 4.3 (adiabatic quantum state generation). *An adiabatic quantum state generator $H_x(s)$ is a function from $x \in \{0,1\}^n$ to adiabatic paths $\{H_x(s)\}_{s\in[0,1]}$. We require that the generator is* explicit*, i.e., that there is a quantum machine running in time polynomial in its input and output length, such that*

- *on input $x \in \{0,1\}^n$ outputs $\alpha(H_x(0))$, the ground-state of $H_x(0)$, and*
- *on input $x \in \{0,1\}^n$, $s \in [0,1]$, $t > 0$, and ϵ outputs a $poly(n, t, \frac{1}{\epsilon})$-size circuit $C_x(s)$ approximating $e^{-itH_x(s)}$ to within ϵ accuracy.*

We define $T(x,\epsilon) = \frac{\eta^2(H_x(\cdot))}{\epsilon\cdot\Delta^2(H_x(\cdot))}$. For $\epsilon > 0$ we let $T_\epsilon = \max_x\{T(x,\epsilon)\}$, and we say the adiabatic quantum state generator $H(\cdot)$ takes time T_ϵ (for the given ϵ).

4.3. Circuit simulation of adiabatic quantum state generation. We now prove that an adiabatic quantum state generator can be simulated efficiently by a quantum circuit.

Theorem 1.2 (formal). *Let $\epsilon > 0$. Let $H_x(s)$ be an adiabatic state generator taking time T_ϵ. There exists a quantum circuit of size $poly(T_\epsilon, \frac{1}{\epsilon}, n)$ such that for every input x, it generates $\alpha(H_x(1))$ to within ϵ accuracy.*

Proof. We start by an overview of the proof. The circuit is built by discretizing time to sufficiently small intervals of length $\delta = \frac{1}{R}$ for some large enough $R = poly(T_\epsilon, \frac{1}{\epsilon}, n)$. At each time step j, $j = 1, \ldots, R$, we apply a measurement in a basis which includes the ground-state $\alpha(H(s_j))$. In other words, we attempt to project $\alpha(H(s_{j-1}))$ onto $\alpha(H(s_j))$. This is done using the Hamiltonian-to-measurement lemma (Lemma 1.1). If R is sufficiently large, the subsequent Hamiltonians are very close in the spectral norm, and their ground-states are very close in the Euclidean norm (by Claim 1.1). Given that at time step j the state is the ground-state $\alpha(H(s_j))$, the next measurement results with very high probability in a projection onto the new ground-state $\alpha(H(s_{j+1}))$. The Zeno effect [42] guarantees that the error probability behaves like $1/R^2$, i.e., quadratically in R (and not linearly), and so the accumulated error after R steps is still small, which implies that the probability that the final state is the ground-state of $H(1)$ is very high, if R is taken to be large enough. We now give a formal treatment.

The description of the quantum circuit. For a given input x, the adiabatic state generator specifies an adiabatic path $H_x(s)$. Recall that $[0,1]$ can be decomposed into $m = poly(n)$ time intervals of the form $[s_j, s_{j+1}]$ where $H(\cdot)$ is continuous on $[s_j, s_{j+1}]$ and differentiable on (s_j, s_{j+1}). Let $\eta = \eta(H_x(\cdot)), \Delta = \Delta(H_x(\cdot))$. We divide each interval into R equal intervals, where we choose $R \geq \Theta(\frac{\eta^2}{\Delta^2}\frac{m}{\epsilon})$, and we set $t_{j,k} = s_j + (s_{j+1} - s_j)\frac{k}{R}$. For each interval, we apply the following R steps. At the kth step, $k = 1, \ldots, R$, we apply the operation $O_{H(t_{j,k})}$ defined in the statement of the Hamiltonian-to-measurement lemma (Lemma 1.1). Each of these applications of O_H takes time which is $poly(n, 1/\Delta)$, by Lemma 1.1. The complexity of the algorithm is therefore $O(\frac{\eta^2}{\Delta^2}\frac{m^2}{\epsilon})$ times the complexity of applying the measurement from Lemma 1.1. This is indeed $poly(T_\epsilon, n, 1/\epsilon)$.

Error analysis in the case that O_H is perfect. We first show the algorithm works when we assume that the O_H's are perfect; i.e., in Lemma 1.1, $\langle\gamma|\beta(\alpha^{\perp})\rangle = 0$. We show that starting with the state $\alpha(H(s_j))$, the state after the jth interval is, with high probability, $\alpha(H(s_{j+1}))$. We first bound the relative change of $H(s+\delta)$ with respect to $H(s)$. For $s, s+\delta \in [s_j, s_{j+1}]$,

$$\begin{aligned}\|H(s+\delta) - H(s)\| &= \left\| \int_s^{s+\delta} \frac{dH}{ds}(s)ds \right\| \\ &\leq \int_s^{s+\delta} \left\| \frac{dH}{ds}(s) \right\| ds \ \leq \ \eta \cdot \delta.\end{aligned}$$

Hence, $\|H(t_{j,k+1}) - H(t_{j,k})\| \leq \frac{\eta}{R}$. Claim 1.1 implies that

$$|\langle \alpha(H(t_{j,k+1})) \mid \alpha(H(t_{j,k})) \ \rangle| \ \geq \ 1 - 4\frac{\eta^2}{R^2\Delta^2}.$$

Hence the probability for successful projection at the $k'th$ measurement, i.e., the probability that the outcome is indeed the ground-state, is $(1 - \frac{4\eta^2}{R^2\Delta^2})^2 \geq 1 - \frac{8\eta^2}{R^2\Delta^2}$. The probability that we err at any of the R steps in the jth interval is therefore at most $O(\frac{\eta^2}{R\Delta^2})$. And the probability that we err at any of the intervals is therefore at most m times that. This is at most ϵ by our choice of R.

Including nonperfect O_H's. We now have to correct the fact that we can apply the measurements with only some exponentially good accuracy but not exactly. The above discussion showed that in the perfect measurement case, the output is within ϵ trace distance from the desired density matrix of the final ground-state. To analyze the nonperfect case, we keep track of the entire system (recall that O_H adds ancilla qubits to operate on). We now compare the overall state of the system after the application of O_H to the overall state after the application of an ideal O_H which simulates a perfect measurement. By Lemma 1.1, the Euclidean distance between the states is arbitrarily small. Summing up all these errors over polynomially many O_H's still results in an arbitrarily small distance from the state in the case of perfect measurements. When considering the reduced state to the original subspace, this results in a state which is arbitrarily close (in trace distance) to the state in the case of perfect measurements. $\square$

5. ASG for Markov chain states.

Finally, we show how to use our techniques to generate interesting quantum states related to Markov chains and approximate counting algorithms. We give some Markov chain background below; for more background, see [38] and the references therein. For background regarding approximate counting, see [33].

5.1. Markov chain background.

We consider a Markov chain on a graph, with nodes indexed by n bit strings. The bits strings are called *states* (not to be confused with quantum states). The Markov chain is characterized by a matrix M operating over the state space. The matrix M has eigenvalues between -1 and 1. Under mild conditions on M (namely, M is connected and aperiodic), for any p an initial probability distribution over the state space, the limit $\lim_{t\to\infty} pM^t = \pi$ exists, π is called the limiting distribution, it is independent of p, and it is a left eigenvector of M with eigenvalue 1. Also, $\pi_i > 0$ for all i.

A Markov chain is *reversible* if for the limiting distribution π, for every i and j, it holds that $M[i,j] \cdot \pi_i = M[j,i] \cdot \pi_j$, i.e., if every directed edge has the same weight under the stationary distribution. We note that any symmetric Markov chain M is reversible, and its limiting distribution must be the uniform distribution.

A Markov chain is said to be *rapidly mixing* if starting from any initial distribution, the distribution after $poly(n, \frac{1}{\epsilon})$ steps is within ϵ total variation distance from the limiting distribution π. A reversible Markov chain is rapidly mixing iff its second eigenvalue gap, namely, the difference between the first and second largest (in absolute value) eigenvalues, is nonnegligible, namely, bounded from below by $1/poly(n)$.

For the sake of simplicity, we restrict our attention in this paper to Markov chains with nonnegative eigenvalues. This is standardly done, by adding self-loops with probability 1/2, and makes sure that no absolute values are needed in the definition of the eigenvalue gap.

5.2. Reversible Markov chains and Hamiltonians. For a reversible Markov chain M with a limiting distribution π, we define

$$H_M = I - \mathrm{Diag}(\sqrt{\pi}) \cdot M \cdot \mathrm{Diag}\left(\frac{1}{\sqrt{\pi}}\right),$$

where $\mathrm{Diag}(\sqrt{\pi})$ is the diagonal matrix with $\sqrt{\pi_i}$ in its diagonal ith entry. Similarly, $\mathrm{Diag}(\frac{1}{\sqrt{\pi}})$ has $\frac{1}{\sqrt{\pi_j}}$ over its diagonal.

A direct calculation shows that M is reversible iff H_M is symmetric. Thus, for a reversible Markov chain, we denote by H_M the *Hamiltonian corresponding to M*. The properties of H_M and M are very much related.

CLAIM 5.1. *Suppose M is a reversible Markov chain with limiting distribution π. Then*

- *H_M is a Hamiltonian with $\|H_M\| \leq 2$;*
- *the spectral gap of H_M equals the second eigenvalue gap of M.*

Let us define $|\pi\rangle \overset{\text{def}}{=} \sum_i \sqrt{\pi_i}\,|i\rangle$. Then

- *the ground-state $\alpha(H_M)$ of H_M is $|\pi\rangle$ with ground-value* 0.

Proof. If M is reversible, H_M is Hermitian and hence has an eigenvector basis. It is easy to see that v is an eigenvector of H_M with eigenvalue λ iff $\mathrm{Diag}(\sqrt{\pi})^{-1}v$ is an eigenvector of M with eigenvalue $1-\lambda$. Also, $v^t\mathrm{Diag}(\sqrt{\pi})$ is a left eigenvector of M with the same eigenvalue. It follows that if the eigenvalues of H_M are $\{\lambda_r\}$, then the eigenvalues of M are $\{1-\lambda_r\}$. M is a reversible Markov chain and therefore has eigenvalues between -1 and 1, and the first two items of the claim follow. If we denote $v = (\pi_1, \ldots, \pi_n)$ to be the (unique) left eigenvector of M with eigenvalue 1, then $\mathrm{Diag}(\sqrt{\pi})^{-1}v$ is the (unique) eigenvector of H_M with eigenvalue 0. All other eigenvectors of M have eigenvalues strictly smaller than 1, and so all other eigenvectors of H_M have eigenvalues strictly larger than 0. It follows that $|\pi\rangle = \mathrm{Diag}(\sqrt{\pi})^{-1}v$ is the unique ground-state of H_M with ground-value 0. □

This gives a direct connection between Hamiltonians, spectral gaps, and ground-states on one hand and rapidly mixing reversible Markov chains and limiting distributions on the other hand.

5.3. Simulating $\boldsymbol{H_M}$. Even if M is sparse and explicit, its corresponding Hamiltonian H_M might not be explicit, because for approximating $H_M[i,j] = -\sqrt{\frac{\pi_i}{\pi_j}}M[i,j]$ we need to be able to approximate $\frac{\pi_i}{\pi_j}$. Special cases are easier. For example, it is easy to compute $\frac{\pi_i}{\pi_j}$ when M is symmetric. For general reversible Markov chains we define the following.

DEFINITION 5.1 (strongly samplable). *A reversible Markov chain on the state space Ω with limiting distribution π is called strongly samplable if it is*

- *sparse and explicit, and,*
- *given $i, j \in \Omega$, there is an efficient way to approximate $\frac{\pi_i}{\pi_j}$.*

Sparseness and explicitness hold in most interesting Markov chains. The second requirement is more restrictive. Still, we note that it holds in many interesting cases such as all Metropolis algorithms (see [29]). As $H_M[i,j] = -\sqrt{\frac{\pi_i}{\pi_j}} M[i,j]$ for $i \neq j$, we see that if M is strongly samplable, then H_M is sparse and explicit. As H_M has bounded norm, we can use the sparse Hamiltonian lemma (Lemma 1.4). This implies the following corollary.

Corollary 5.1. *If a Markov chain M is strongly samplable, then H_M is simulatable.*

5.4. From Markov chains to quantum state generation. We now consider sequences of Markov chains and define the following.

Definition 5.2 (slowly varying Markov chains). *Let $\{M_t^n\}_{t=1}^{T=T(n)}$ be a sequence of Markov chains on Ω_n, $|\Omega_n| = N = 2^n$. Let π_t^n be the limiting distribution of M_t^n. We say that the sequence is* slowly varying *if there exists some $c > 0$ such that for all large enough n, for all $1 \leq t \leq T(n)$ $|\pi_t^n - \pi_{t+1}^n| \leq 1 - 1/n^c$.*

We can now state Theorem 1.3 precisely.

Theorem 1.3 (formal). *Let $\{M_t^n\}_{t=1}^{T}$ be a slowly varying sequence of strongly samplable Markov chains which are all rapidly mixing, and let π_t^n be their corresponding limiting distributions. Then if there exists an efficient quantum state generator for $|\pi_0^n\rangle$, then there exists an efficient quantum state generator for $|\pi_{T(n)}^n\rangle$.*

Proof. By Corollary 5.1, the Hamiltonians $H_{M_t^n}$ are simulatable. Also, Claim 5.1 implies that $\|H_{M_t^n}\| \leq 2$ and that the ground-values of all these Hamiltonians are 0. Also, the Markov chains in the sequence are rapidly mixing, which means that they have nonnegligible spectral gaps, say $\geq \frac{1}{n^b}$, for some $b > 0$. This means that $\Delta(H_{M_t^n}) \geq \frac{1}{n^b}$. To complete the proof, we show that the inner product between the ground-states of subsequent Hamiltonians is nonnegligible. Indeed,

$$\begin{aligned}\langle \alpha(H_{M_t}) | \alpha(H_{M_{t+1}}) \rangle &= \langle \pi_t | \pi_{t+1} \rangle \\ &= \sum_i \sqrt{\pi_t(i) \pi_{t+1}(i)} \geq 1 - |\pi_t - \pi_{t+1}| \geq \frac{1}{n^c},\end{aligned}$$

where the second to last inequality follows from Fact 2.1. The theorem then follows from the jagged adiabatic path lemma (Lemma 1.3). □

Essentially all Markov chains that are used in approximate counting algorithms that we are aware of meet the criteria of the theorem. The following is a partial list of states we can generate by applying this theorem. The citations refer to the approximate counting algorithms that we use as the basis for the quantum state generation algorithm:

1. uniform superposition over all perfect matchings of a given bipartite graph [32],
2. all lattice points contained in a high-dimensional convex body satisfying the conditions of [9],
3. various Gibbs distributions over rapidly mixing Markov chains using the Metropolis filter [38],
4. log-concave distributions [9],
5. all linear extensions of a given partial order [12].

Remark 5.1. We note that the second requirement in the definition of strongly samplable Markov chains (Definition 5.1) is crucial. If this requirement can be relaxed, and one can prove Theorem 1.3 without it, then these techniques could have been used to solve the QS problem related to the graph isomorphism problem and thus to give a quantum algorithm for graph isomorphism.

We illustrate our technique with the example of how to Qsample all perfect matchings of a given bipartite graph.

5.5. An example: All perfect matchings of a bipartite graph. In this section we rely heavily on the work of Sinclair, Jerrum, and Vigoda [32], who recently showed how to efficiently approximate the permanent of any matrix with nonnegative entries. It is well known that this can be done if one can efficiently sample a random perfect matching of a given input bipartite graph. Sinclair et al. achieve the latter using a sequence of Markov chains $M_1, \ldots, M_T$ on the set of matchings of a bipartite graph. The details of this work are far too involved to fully explain here, and we refer the interested reader to [32] for further details.

In a nutshell, the idea in [32] is to start with M_1, which is a Metropolis random walk on the set of perfect and near perfect matchings (i.e., perfect matchings minus one edge) of the complete bipartite graph. Since [32] is interested in a given input bipartite graph which is a subgraph of the complete bipartite graph, they assign weights to the edges such that weights on the edges that do not participate in the input graph are slowly decreasing as we move from M_1 to M_T until their weights in the final Markov chain M_T practically vanish. The weights of the edges are updated from Markov chain M_t to the next M_{t+1} using data that is collected from running the Markov chain M_t with the current set of weights.

The final Markov chain M_T with the final set of parameters converges to a probability distribution which is essentially concentrated on the perfect and near perfect matchings of the input graph, where the probability of the perfect matchings is $1/n$ times that of the near perfect matching.

We would like to apply Theorem 1.3 in order to solve the quantum sampling problem for the limiting distribution of M_T, namely, to generate the coherent superposition over perfect and near perfect matchings with the correct weights.

We need to check that the conditions of the theorem hold. It is easy to check that the Markov chains M_t being used in [32] are all strongly samplable, since they are Metropolis chains, and so the corresponding Hamiltonians are simulatable by Corollary 5.1. Moreover, the sequence of Markov chains is slowly varying. To apply Theorem 1.3, it remains to show that we can Qsample the limiting distribution of M_1, the first Markov chain in the sequence.

The limiting distribution of the initial Markov chain M_1 is a distribution over all perfect and near perfect matchings in the complete bipartite graph, with each near perfect matching having weight n times that of a perfect matching, where n is the number of nodes of the given graph. To generate this superposition we do the following:

- We generate in the first register $\sum_{\pi \in S_n} |m_\pi\rangle$, where m_π is the matching on the complete bipartite graph induced by $\pi \in S_n$. We can efficiently generate this state because we can generate a superposition over all permutations in S_n, and there is an easy computation from a permutation to a perfect matching in a complete bipartite graph, and vice versa.
- In the second register, we generate the state $|0\rangle + \sqrt{n} \sum_{i=1}^{n} |i\rangle$ normalized on a $\log(n)$-dimensional register. This can be done efficiently because any unitary transformation on $\log(n)$ qubits can be performed efficiently.
- We apply a transformation that maps $|m, i\rangle$ to $|0, m\rangle$ when $i = 0$, and to

$|0, m-\{e_i\}\rangle$ for $i>0$, where $m-\{e_i\}$ is the matching m minus the $i'th$ edge in the matching. There is an easy computation from $m-\{e_i\}$ to m, i, and vice versa, and so this transformation can be done efficiently. We are now at the desired state.

Thus we can apply Theorem 1.3 to generate the limiting distribution of the final Markov chain. We then measure to see whether or not the matching is perfect, and with nonnegligible probability we project the state onto the uniform distribution over all perfect matchings of the given graph.

6. The basic tools.

6.1. A lemma about ground-states of close Hamiltonians.

CLAIM 6.1 (Claim 1.1 repeated). *Let A, B be two Hamiltonians of equal dimensions such that $\|A-B\| \leq \eta$. Moreover, assume that A, B have spectral gaps bounded from below: $\Delta(A), \Delta(B) \geq \Delta$. Then $|\langle\alpha(A)|\alpha(B)\rangle| \geq 1-\frac{4\eta^2}{\Delta^2}$.*

Proof. Adding $g \cdot I$ to both matrices, for any constant g, does not affect the spectral norm of the difference, the spectral gaps, or the inner product between the ground-states. We can therefore assume without loss of generality that A and B are positive, A's ground-value is 0, and B's ground-value, denoted by λ_B, is larger than 0.

Since $\lambda_B = \min_{v:|v|=1} |Bv|$, we have $\lambda_B \leq |B\alpha(A)|$. Also,

$$|B\alpha(A)| \leq |A\alpha(A)| + |(B-A)\alpha(A)| \ \leq \ \eta, \tag{6.1}$$

and so $\lambda_B \leq \eta$. We now express $\alpha(A) = c\alpha(B) + c^{\perp}\alpha(B)^{\perp}$, with $\alpha(B)^{\perp} \perp \alpha(B)$. Then

$$\begin{aligned} |B\alpha(A)| &= |cB\alpha(B) + c^{\perp}B\alpha(B)^{\perp}| \\ &\geq |c^{\perp}| \cdot \Delta - |c| \cdot \lambda_B \\ &\geq |c^{\perp}| \cdot \Delta - |c| \cdot \eta. \end{aligned} \tag{6.2}$$

Equations (6.1) and (6.2) together imply that $\eta \geq |c^{\perp}| \cdot \Delta - |c| \cdot \eta$. We see that $|c| \geq \frac{\Delta}{\eta}|c^{\perp}| - 1$. Equivalently, $\sqrt{1-|c^{\perp}|^2} \geq \frac{\Delta}{\eta}|c^{\perp}| - 1$.

Denoting $r = \frac{\Delta}{\eta} > 0$, we see that if the right-hand side is negative, then $|c^{\perp}| \leq \frac{1}{r}$; otherwise, solving the inequality we get

$$|c^{\perp}| \leq \frac{2r}{r^2+1} \leq \frac{2r}{r^2} = \frac{2}{r}.$$

We get $|\langle\alpha(A)|\alpha(B)\rangle| = |c| = \sqrt{1-|c^{\perp}|^2} \geq 1 - \frac{4}{r^2} = 1 - \frac{4\eta^2}{\Delta^2}$ as desired. □

6.2. The spectral gap of a convex combination of projections.

We now prove the basic but useful Claim 1.2 regarding the convex combination of two projections. Recall that for a vector $|\alpha\rangle$, the Hamiltonian $\Pi_\alpha = I - |\alpha\rangle\langle\alpha|$ is the projection onto the subspace orthogonal to α.

CLAIM 6.2 (Claim 1.2 repeated). *Let $|\alpha\rangle, |\beta\rangle$ be two vectors in some Hilbert space. For any convex combination $H_\eta = (1-\eta)\Pi_\alpha + \eta\Pi_\beta$, $\eta \in [0,1]$, we have $\Delta(H_\eta) \geq |\langle\alpha|\beta\rangle|$.*

Proof. We observe that the problem is two-dimensional. We write $|\beta\rangle = a|\alpha\rangle + b|\alpha^{\perp}\rangle$. We now express the matrix H_η in a basis which contains $|\alpha\rangle$ and $|\alpha^{\perp}\rangle$. The eigenvalues of this matrix are all 1, except for a two-dimensional subspace, where the matrix is exactly

$$\left(\begin{array}{cc} \eta|a|^2 + (1-\eta) & \eta ab^* \\ \eta a^*b & \eta|b|^2 \end{array}\right).$$

Diagonalizing this matrix we find that the spectral gap is exactly $\sqrt{1-4(1-\eta)\eta|b|^2}$, which is minimized for $\eta = 1/2$ where it is exactly $|a| = |\langle\alpha|\beta\rangle|$. □

6.3. The Hamiltonian-to-measurement and projection lemmas. Consider a simulatable Hamiltonian H with ground-state $|\alpha\rangle$. It is sometimes desirable to apply a measurement in the basis of eigenstates of the Hamiltonian. The Hamiltonian-to-measurement lemma provides an *approximation* of this procedure in the case where the spectral gap of the Hamiltonian is nonnegligible. The lemma is based on Kitaev's phase estimation procedure, which we now recall.

LEMMA 6.1 (phase estimation, adapted from [35]; see also [46, section 5.2]). *Let U be a unitary transformation on n qubits, and assume U can be implemented by a poly(n)-size circuit. Let $\epsilon > 0$. Then there exists a quantum procedure Q running in time poly$(n, \frac{1}{\epsilon})$ which on input v, that is an eigenvector of U with eigenvalue $e^{i\lambda}$, outputs $Q\,|v,0\rangle = |v,\psi\rangle$ such that the following conditions hold. $|\psi\rangle$ is exponentially close in fidelity to another vector $|\psi'\rangle$, $F(|\psi\rangle, |\psi'\rangle) \geq 1-2^{-\Omega(n)}$, where $|\psi'\rangle = |\lambda'\rangle \otimes |0\rangle$, and λ' is a real number such that $|\lambda'-\lambda| \leq \epsilon$. The right register in $|\psi'\rangle$ consists of the ancilla bits of the algorithm.*

We can now proceed to proving Lemma 1.1.

LEMMA 6.2 (Lemma 1.1 repeated). *Assume H is a simulatable Hamiltonian on n qubits. For any constant d, there exists a poly$(n, \frac{1}{\Delta(H)})$-size circuit O_H which takes $|\alpha(H)\rangle$ to $|\alpha(H)\rangle \otimes |\gamma\rangle$, and for any eigenstate of H $|\alpha^\perp\rangle$ orthogonal to the ground-state, $O_H|\alpha^\perp\rangle = |\alpha^\perp\rangle \otimes |\beta(\alpha^\perp)\rangle$, where $|\langle\gamma|\beta(\alpha^\perp)\rangle| \leq O(n^{-d})$.*

Proof. We would like to apply the phase estimation algorithm for the unitary matrix e^{iH}, with $\epsilon = \Delta(H)/2$. Suppose we could exactly simulate H; namely, we could apply e^{iH} exactly. In this case, by the above lemma, Kitaev's phase estimation algorithm does the following. For an input state which is a ground-state of H, the output state is exponentially close to a vector of the form $|\lambda'\rangle \otimes |0\rangle$ for $|\lambda'-\lambda_0| < \Delta(H)/2$, where λ_0 is the ground-value. For an input state which is any other eigenvector, the output state is exponentially close to a vector of the form $|\lambda''\rangle \otimes |0\rangle$ for $|\lambda''-\lambda_j| < \Delta(H)/2$. For a superposition of eigenvectors not including the ground-state, the output is a superposition of such vectors. The two output vectors are therefore exponentially close to being orthogonal, as required.

We now need to show how to apply the phase estimation algorithm. To do that, we recall that H is simulatable, and so we can ζ-approximate e^{-iH} in time polynomial in n and $\frac{1}{\zeta}$. We pick ζ to be small enough, but still inverse polynomial in n and $\frac{1}{\epsilon}$, as follows. Let m be the number of times that e^{iH} is applied in the phase estimation algorithm, with the above ϵ. We set ζ to be n^{-d}/m. We then apply the above phase estimation algorithm, where every time we need to apply e^{iH}, we simulate it with this accuracy. The accumulated error due to the inaccuracies in the simulation of H is at most n^{-d}. The output states are therefore within n^{-d} Euclidean distance from the correct ones. This means that the inner product discussed above (between the output state for an input which is a ground-state, and the output state in case the input is orthogonal) is also $O(n^{-d})$. □

We now prove the Hamiltonian-to-projection lemma (Lemma 1.2). Consider a simulatable Hamiltonian H whose ground-state is α. This time we want to simulate the Hamiltonian $\Pi_{\alpha(H)}$, rather than the original Hamiltonian H. The Hamiltonian-to-projection lemma provides a way to do this, provided the spectral gap of H is nonnegligible.

LEMMA 6.3 (Lemma 1.2 repeated). *Assume H is a simulatable Hamiltonian on n qubits, with nonnegligible spectral gap $\Delta(H) \geq 1/n^c$ for some constant $c > 0$ and* *with a known ground-value. Then the Hamiltonian $\Pi_{\alpha(H)}$ is simulatable.*

Proof. As before, let us start with the assumption that we can apply e^{iH} efficiently and perfectly. We do the following:

- Apply Kitaev's phase estimation algorithm for the unitary matrix e^{iH}, using $\epsilon = \Delta(H)/2$. Now, given the output λ', we can write down one bit of information on an extra qubit: whether an input eigenstate of H is the ground-state or orthogonal to it (it is here that we use the fact that we know the ground-value).
- Apply a phase shift of value e^{-it} to this extra qubit, conditioned that it is in the state $|1\rangle$ (if it is $|0\rangle$, do nothing). This conditional phase shift corresponds to applying for time t a Hamiltonian with two eigenspaces, the ground-state and the subspace orthogonal to it, with respective eigenvalues 0 and 1, which is exactly the desired projection $\Pi_{\alpha(H)}$.
- Finally, to erase the extra data written down, we reverse the first step and uncalculate the information written on the qubits again using Kitaev's phase estimation algorithm.

This procedure computes the desired transformation $e^{i\Pi_{\alpha(H)}t}$ on any vector, with $1 - 2^{-O(n)}$ fidelity.

As in the proof of Lemma 1.1, we now need to take into account the fact that we cannot apply e^{iH} exactly, but we can simulate H only approximately. We approximate each of the applications of e^{iH} to within ζ. To determine ζ, we note that the number of times m we apply e^{iH} in the above procedure is $poly(n, \frac{1}{\Delta(H)})$. To get an overall error of size ζ', we simply fix ζ to be $\zeta'/2m$. The overall procedure is then polynomial in $\zeta', n, \frac{1}{\Delta(H)}$. □

Remark 6.1. We remark that in the two previous lemmas there is nothing special about the ground-state of the Hamiltonian. The same techniques work for measuring or projecting onto any eigenstate with a *known* eigenvalue, that is, separated from all other eigenvalues.

6.4. The jagged adiabatic path lemma. Next, we consider the question of which paths in the Hamiltonian space guarantee nonnegligible spectral gaps. Figure 6.1 gives an example of two Hamiltonians H_1, H_2 with nonnegligible spectral gaps that can be connected through a jagged line but not through a direct line.

The jagged adiabatic path lemma (Lemma 1.3) provides a way, in a more specific case, to connect Hamiltonians such that the spectral gaps along the path are always nonnegligible. The additional condition in Lemma 1.3 is that the ground-state of H_j is close to the ground-state of H_{j+1} (this condition is not fulfilled in the example of Figure 6.1). It may seem natural that in this case the way to prove the jagged adiabatic path lemma is to connect each pair of Hamiltonians H_j and H_{j+1} by a straight line. However, again this does not work. To see this consider

$$H_1 = \begin{pmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad H_2 = \begin{pmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix},$$

where λ_1 and λ_2 are the eigenvalues of

$$\begin{pmatrix} 1.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix}$$

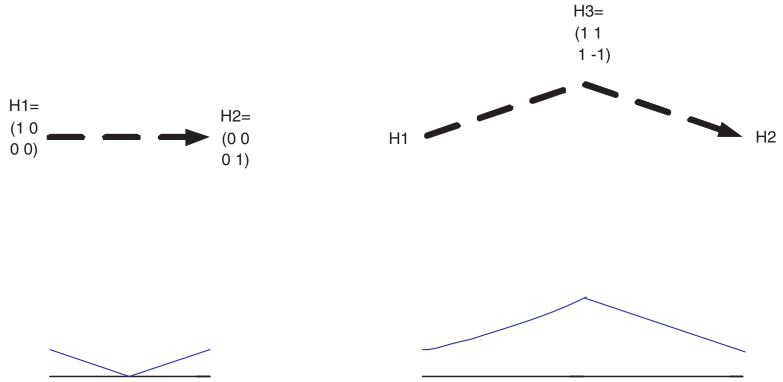


FIG. 6.1. *In the left side of the drawing, we see two Hamiltonians H_1 and H_2 connected by a straight line and the spectral gaps along that line. In the right side of the drawing, we see the same two Hamiltonians H_1 and H_2 connected through a jagged line that goes through a third connecting Hamiltonian H_3 in the middle and the spectral gaps along that jagged path. Note that on the left the spectral gap becomes zero in the middle, while on the right it is always larger than one.*

(they are about 0.3 and 1.7). Let us say the matrices are represented in the orthonormal basis $\{v_1, \dots, v_4\}$. Then H_1 has $|v_4\rangle$ as a unique ground-state with ground-value 0, and H_2 has $\frac{1}{\sqrt{2}}[|v_3\rangle - |v_4\rangle]$ as a unique ground-state with ground-value 0. It is now easy to check that in $\frac{1}{2}[H_1 + H_2]$ all eigenspaces have dimension two, and no eigenvalue is separated from the others.

The problem with connecting H_1 and H_2 by a line stems from the fact that H_i may behave arbitrarily outside the subspace containing their ground-states. This also hints at the solution. As we are interested only in the ground-states, let us project each H_i onto the subspace orthogonal to its ground-state; i.e., if H is a Hamiltonian, let us define Π_H to be the projection onto the space orthogonal to the ground-state of H. We then replace the sequence $\{H_j\}$ with the sequence of Hamiltonians $\{\Pi_{H_j}\}$, and we connect two neighboring *projections* by a line.

LEMMA 6.4 (Lemma 1.3 repeated). *Let $\{H_j\}_{j=1}^{T=poly(n)}$ be a sequence of bounded norm, simulatable Hamiltonians on n qubits, with nonnegligible spectral gaps, $\Delta(H_j) \geq n^{-c}$, and with known ground-values, such that the inner product between the unique ground-states $\alpha(H_j), \alpha(H_{j+1})$ is at least n^{-c} for all j. Then there exists an adiabatic state generator with $\alpha(H_0)$ as its initial state and $\alpha(H_T)$ as its final state. In particular there exists an efficient quantum algorithm that takes $\alpha(H_0)$ to within arbitrarily small distance from $\alpha(H_T)$.*

Proof. We replace the sequence $\{H_j\}$ with the sequence of Hamiltonians $\{\Pi_{H_j}\}$, and we connect two neighboring projections by a line. We claim the following:

- As projections, Π_{H_j} have bounded norms, $\|\Pi_{H_j}\| \leq 1$. As the Hamiltonians on the path connecting these projections are convex combinations of the projections, they also have bounded norm.
- We proved in Lemma 1.2 that the fact that H_j is simulatable and has a known ground-value implies that Π_{H_j} is also simulatable. It follows that the Hamiltonians on the path connecting these projections, which are convex combinations of the projections and are of polynomially bounded norms, are also simulatable (see section 4). This means that all the Hamiltonians on the path are simulatable.

- The projections Π_{H_i} have the same ground-states as the Hamiltonians H_i, and as a result each two neighboring projections have nonnegligible inner product between their ground-states. In Claim 1.2 we showed that this implies the Hamiltonians on the line connecting Π_{H_j} and $\Pi_{H_{j+1}}$ have nonnegligible large spectral gaps. Notice that this step is possible only when working with the projections Π_H but not with the Hamiltonians H_i themselves.

Thus, we can define an adiabatic state generator in which the initial state is $\alpha(H_0)$ and the final state is $\alpha(H_T)$. The interval $[0,1]$ of the rescaled time s is divided equally between the different steps of the jagged path; when we move from one projection to the next, s increases by $1/T$ (recall that T is the number of Hamiltonians we connect). Note that this implies that the maximal norm of the first derivative of the Hamiltonians in the adiabatic state generator, which is denoted by η in the definition of ASG, is $O(T)$. The minimal spectral gap in the adiabatic state generator, which we denoted by Δ, is bounded from below by an inverse polynomial in n, by the arguments above. This gives us an adiabatic state generator which takes time $T_\epsilon = \frac{\eta^2}{\epsilon\Delta^2} = poly(T, n, \frac{1}{\epsilon})$. We can now apply Theorem 1.2 and get an efficient quantum algorithm that takes $\alpha(H_0) = \alpha(\Pi_{H_0})$ to within arbitrarily small distance from $\alpha(H_T) = \alpha(\Pi_{H_T})$. $\square$

6.5. The sparse Hamiltonian lemma. The sparse Hamiltonian lemma (Lemma 1.4) gives fairly general conditions for a Hamiltonian to be simulatable: the Hamiltonian need only be sparse and explicit.

The main idea of the proof is to explicitly write H as a sum of polynomially many bounded norm Hamiltonians H_m, which are all block diagonal (in a combinatorial sense), and such that the size of the blocks in each matrix is at most 2×2. We then show that each Hamiltonian H_m is simulatable and use Trotter's formula to simulate H.

6.5.1. Decomposition of H as a sum of block diagonal matrices with 2×2 blocks.

Definition 6.1 (combinatorial block). *We say that A is combinatorially block diagonal if we can decompose the set of rows of A by $ROWS(A) = \bigcup_{b=1}^{B} R_b$, where we require that $A(i,j) \neq 0$ implies that there exists b such that both $i \in R_b$ and $j \in R_b$.*

We say that A is 2×2 combinatorially block diagonal if, for every b, either $|R_b| = 1$ or $|R_b| = 2$.

Claim 6.3 (decomposition lemma). *Let H be a sparse explicit Hamiltonian over n qubits, with at most D nonzero elements in each row. Then there is a way to decompose H into $H = \sum_{m=1}^{(D+1)^2 n^6} H_m$, where*

- *each H_m is a sparse explicit Hamiltonian over n qubits, and*
- *each H_m is 2×2 combinatorially block diagonal.*

Proof. We color all the entries of H with $(D+1)^2 n^6$ colors as follows. For $(i,j) \in [N] \times [N]$ and $i \leq j$ (i.e., (i,j) is an upper-diagonal entry), we define

$$col_H(i,j) = (k\ ,\ i \bmod k\ ,\ j \bmod k\ ,\ rindex_H(i,j)\ ,\ cindex_H(i,j)),$$

where we have the following:

- If $i = j$, we set $k = 1$; otherwise, we let k be the first integer in the range $[2 \ldots n^2]$ such that $i \neq j \pmod{k}$. We know there must be such a k (for the product of all primes smaller than n^2 is larger than 2^n, and by the Chinese remainder theorem two numbers that have the same modula are equal).

- $rindex_H(i,j) = 0$ if $H_{i,j} = 0$, and otherwise it is the index of $H_{i,j}$ in the list of all nonzero elements in the ith row of H. We define $cindex_H(i,j)$ similarly, using the columns.

For lower-diagonal entries, $i > j$, we define $col_H(i,j) = col_H(j,i)$. Altogether, we use at most $(n^2)^3 \cdot (D+1)^2$ colors.

The Hamiltonian's entries are decomposed by their colors. For a color m, we define $H_m[i,j] = H[i,j] \cdot \delta_{col_H(i,j),m}$; i.e., H_m is H on the entries colored by m and zero everywhere else. Clearly, $H = \sum_m H_m$, and each H_m is Hermitian. Also as H is explicit and sparse, there is a simple $poly(n)$-time classical algorithm computing the coloring $col_H(i,j)$, and so each H_m is also explicit and sparse. It is left to show that it is 2×2 combinatorially block diagonal.

Indeed, fix a color m and consider H_m. For every nonzero element (i_b, j_b) of H_m, we define a block. If $i_b = j_b$, then we set $R_b = \{i_b\}$, while if $i_b \neq j_b$, then we set $R_b = \{i_b, j_b\}$. Say $i_b \neq j_b$ (the $i_b = j_b$ case is similar and simpler). We know that the elements (i_b, j_b) and (j_b, i_b) are colored by the same color m, and suppose $m = (k, i_b \bmod k, j_b \bmod k, rindex_H(i_b, j_b), cindex_H(i_b, j_b))$. To see that $R_b = \{i_b, j_b\}$ is a 2×2 block, we need to show that there are no other elements colored by m on the i_bth and j_bth rows and columns. As the color m contains the row-index and column-index of (i_b, j_b), it must be that (i_b, j_b) is the only nonzero element in H_m from that row or column. Furthermore, all elements (j_b, a) on the j_bth row have $j_b \bmod k \neq i_b \bmod k$ and therefore are not colored by m. A similar argument shows no element on the i_bth row is colored by m, and the claim follows (see Figure 6.2). □

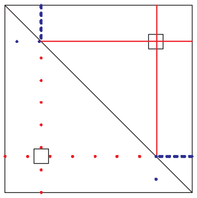


FIG. 6.2. *In the upper diagonal side of the matrix H_m, the row and column of (i_b, j_b) are empty because the color m contains the row-index and column-index of (i,j), and the j_bth row and i_bth column are empty because m contains k, $i \bmod k$, $j \bmod k$, and $i \bmod k \neq j \bmod k$. The lower diagonal side of H_m is just a reflection of the upper diagonal side. It follows that $\{i_b, j_b\}$ is a 2×2 combinatorial block.*

CLAIM 6.4. *For every m, $\|H_m\| \leq \|H\|$.*

Proof. Fix an m. H_m is block diagonal, and therefore its norm $\|H_m\|$ is achieved as the norm of one of its blocks. Now H_m blocks are either

- 1×1, and then the block is $(H_{i,i})$ for some i, and it has norm $|H_{i,i}|$, or
- 2×2, and then the combinatorial block is

$$\begin{pmatrix} 0 & A_{k,\ell} \\ A^*_{k,\ell} & 0 \end{pmatrix}$$

for some k, ℓ, and it has norm $|A_{k,\ell}|$.

It follows that $\max_m \|H_m\| = \max_{k,\ell} |H_{k,\ell}|$. On the other hand, $\|H\| \geq \max_{k,\ell}$ $|H_{k,\ell}|$ (see section 2.2). The proof follows. □

6.5.2. 2×2 combinatorially block diagonal matrices are simulatable.

CLAIM 6.5. *Every 2×2 block diagonal, explicit Hamiltonian A is simulatable to within arbitrary polynomial approximation.*

Proof. The proof is standard, but we include it for completeness.

Let $t > 0$, and let $\alpha > 0$ be an accuracy parameter.

The circuit: A is 2×2 combinatorially block diagonal. It therefore follows that A's action on a given basis state $|k\rangle$ is captured by a 2×2 unitary transformation U_k. Formally, given k, say $|k\rangle$ belongs to a 2×2 combinatorial block $\{k, \ell\}$ in A. We set $b_k = 2$ (for a 2×2 block) and $\min_k = \min(k, \ell)$, $\max_k = \max(k, \ell)$ (for the subspace to which k belongs). We then set A_k to be the part of A relevant to this subspace,

$$A_k = \left(\begin{array}{cc} A_{min_k,min_k} & A_{min_k,max_k} \\ A_{max_k,min_k} & A_{max_k,max_k} \end{array} \right),$$

and $U_k = e^{-itA_k}$. If $|k\rangle$ belongs to a 1×1 block, we similarly define $b_k = 1$, $\min_k = \max_k = k$, $A_k = (A_{k,k})$, and $U_k = (e^{-itA_k})$.

Our approximated circuit simulates this behavior. We need two transformations. We define

$$\widetilde{T_1} : |k, 0\rangle \to \left| b_k, \min_k, \max_k, \widetilde{A_k}, \widetilde{U_k}, k \right\rangle,$$

where $\widetilde{A_k}$ is our approximation to the entries of A_k and $\widetilde{U_k}$ is our approximation to $e^{-it\widetilde{A_k}}$, and where both matrices are expressed by their four (or one) entries. We use $\Theta(\alpha)$ accuracy such that $\|U_k - \widetilde{U_k}\| \le 4\|U_k - \widetilde{U_k}\|_\infty \le \alpha$. Having $\widetilde{U_k}, \min_k, \max_k, k$ written down, we can simulate the action of $\widetilde{U_k}$. We can therefore have an efficient unitary transformation T_2:

$$\widetilde{T_2} : \left| \widetilde{U_k}, \min_k, \max_k \right\rangle |k\rangle = \left| \widetilde{U_k}, \min_k, \max_k \right\rangle \left| \widetilde{U_k} k \right\rangle$$

for $|\widetilde{U_k} k\rangle \in \mathit{Span}\{\min_k, \max_k\}$. We can similarly define T_2, which applies U_k on its input, $T_2 = |A, \min, \max, k\rangle = |A, \min, \max, U_k k\rangle$.

Our algorithm applies $\widetilde{T_1}$, followed by $\widetilde{T_2}$ and then $\widetilde{T_1}^{-1}$ for cleanup.

Correctness: Let us denote $X = e^{-itA} - T_1^{-1} T_2 T_1$. Our goal is to show that $\|X\| \le \alpha$. We notice that X is also 2×2 combinatorially block diagonal, and therefore its norm can be achieved by a vector ψ belonging to one of its dimension one or two subspaces, say, to $\mathit{Span}\{|\min_k\rangle, |\max_k\rangle\}$. On this subspace, we have a 2×2 operator $X = e^{-itA_k} - \widetilde{T_1}^{-1} \widetilde{T_2} \widetilde{T_1}$. Also, $e^{-itA_k} = \widetilde{T_1} T_2 \widetilde{T_1}^{-1}$. It follows that $\|X\| = \|T_2 - \widetilde{T_2}\| = \|U_k - \widetilde{U_k}\| \le \alpha$. □

Remark 6.2. We proved the claim for matrices with 2×2 combinatorial blocks. A similar claim applies for matrices with $m \times m$ combinatorial blocks, with the same proof technique, as long as m is polynomial in n.

6.5.3. Proving the sparse Hamiltonian lemma. We now prove the sparse Hamiltonian lemma.

LEMMA 6.5 (Lemma 1.4 repeated). *If H is an explicit and sparse Hamiltonian on n qubits and $\|H\| \le poly(n)$, then H is simulatable.*

Proof. Let H be row-sparse with $D \le poly(n)$ nonzero elements in each row, and let $\|H\| = \Lambda \le poly(n)$. Let $t > 0$. Our goal is to efficiently simulate e^{-itH} to within α accuracy.

We express $H = \sum_{m=1}^{M} H_m$ as in Claim 6.3, $M \le (D+1)^2 n^6 \le poly(n)$. By Claim 6.5, for every $\delta > 0$ we can simulate $e^{-i\delta H_m}$ to within $\frac{\alpha}{2Mt/\delta}$ accuracy in time

$poly(n, t, M, \frac{1}{\delta}, \frac{1}{\alpha})$. It follows that we can approximate U_δ (see section 4.1.1) to within $\frac{\alpha}{t/\delta}$ accuracy and $U_\delta^{\lfloor \frac{t}{2\delta} \rfloor}$ to within $\frac{\alpha}{2}$ accuracy.

Also, Corollary 4.1 ensures us that $U_\delta^{\lfloor \frac{t}{2\delta} \rfloor}$ is $O(M\widetilde{\Lambda}\delta + M\widetilde{\Lambda}^3 t\delta^2)$ close to e^{-itH}, where $\widetilde{\Lambda} = \max_{k \le \ell} \| \sum_{i=k}^{\ell} H_i \| \le M\|H\|$ (because we saw that for every m, $\|H_m\| \le \|H\|$). Picking δ small enough (inverse polynomial in M, Λ, t), we see that $U_\delta^{\lfloor \frac{t}{2\delta} \rfloor}$ is $\frac{\alpha}{2}$ close to e^{-itH}. Altogether, our approximation is α close to e^{-itH}. It follows that our approximation has circuit size bounded by $poly(n, t, M, \frac{1}{\delta}, \frac{1}{\alpha}) = poly(n, t, \frac{1}{\alpha})$. □

Appendix A. QS instances: Specific problems in SZK. We saw in the introduction an example of a QS problem, the solution of which implies an efficient quantum algorithm for graph isomorphism. Here we give three more examples of problems in SZK which are of particular relevance for quantum algorithms: discrete log, quadratic residuosity, and a gap version of closest vector in a lattice. An efficient QS algorithm for the lattice problem is a major open problem.

We also remind the reader that the existence of the reduction of the problems we consider below to certain instances of QS problems already follows from Theorem 1.1. Here we do a direct reduction and get simple QS instances sufficient for solving the above problems.

A.1. A promise problem equivalent to discrete log.

The problem: Goldreich and Kushilevitz [25] define the promise problem DLP$_c$ as

Input: A prime p, a generator g of Z_p^*, and an input $y \in Z_p^*$.

Promise:

- Yes: $x = \log_g(y)$ is in $[1, cp]$.
- No: $x = \log_g(y)$ is in $[\frac{p}{2} + 1, \frac{p}{2} + cp]$.

Goldreich and Kushilevitz [25] prove that the discrete-log problem is reducible to DLP$_c$ for every $0 < c < 1/2$. They also prove that DLP$_c$ has a perfect zero knowledge proof if $0 < c \le 1/6$. We take $c = 1/6$ and show how to solve DLP$_{1/6}$, given a certain QS algorithm.

The reduction: We assume we can solve the QS problem for the circuit $C_{y,k} = C_{p,g,y,k}$ that computes $C_{y,k}(i) = y \cdot g^i \pmod{p}$ for $0 \le i < 2^k$. The algorithm generates the states $\left|C_{g^{p/2+1}, \lfloor \log(p) \rfloor - 1}\right\rangle, \left|C_{y, \lfloor \log(p) \rfloor - 3}\right\rangle$ and proceeds as in Theorem 1.1.

Correctness:

We have

$$\left|C_{g^{p/2+1}, \lfloor \log(p) \rfloor - 1}\right\rangle = \frac{1}{\sqrt{t}} \sum_{i=0}^{t-1} \left|g^{p/2+i}\right\rangle, \tag{A.1}$$

where t is the largest power of 2 smaller than $p/2$. Also, as $y = g^x$ we have

$$\left|C_{y, \lfloor \log(p) \rfloor - 3}\right\rangle = \frac{1}{\sqrt{t'}} \sum_{i=0}^{t'-1} \left|g^{x+i}\right\rangle, \tag{A.2}$$

where t' is the largest power of 2 smaller than $p/8$. Now, comparing the powers of g in the support of (A.1) and (A.2), we see that

- if $x \in [1, cp]$, then $\left|C_{g^{p/2+1}, \lfloor \log(p) \rfloor - 1}\right\rangle$ and $\left|C_{y, \lfloor \log(p) \rfloor - 3}\right\rangle$ have disjoint supports, and therefore $\langle C_{y, \lfloor \log(p) \rfloor - 3} | C_{g^{p/2+1}, \lfloor \log(p) \rfloor - 1}\rangle = 0$, while

- if $x \in [\frac{p}{2}+1, \frac{p}{2}+cp]$, then the overlap is large and

$$|\langle C_{y,\lfloor \log(p) \rfloor -3} | C_{g^{p/2+1}, \lfloor \log(p) \rfloor -1} \rangle|$$

is a constant.

A.2. Quadratic residuosity.

The problem: The (total) language QR is to decide on input x, n whether or not x is a square modulo n. Without loss of generality we can assume the input x to the problem belongs to Z_n^*. Let us denote xRn if x is a square, i.e., if $x = y^2 \pmod{n}$ for some y. An efficient algorithm is known for the case where n is a prime, and the problem is believed to be hard for $n = pq$, where p, q are chosen at random among large primes p and q. A basic fact that follows directly from the Chinese remainder theorem is the following.

FACT A.1.

- *If the prime factorization of n is $n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$, then for every x*

$$xRn \iff \forall_{1 \le i \le k}\ xRp_i.$$

- *If the prime factorization of n is $n = p_1 p_2 \dots p_k$ with different primes p_i, then every $z \in Z_n^*$ that has a square root has exactly 2^k square roots.*

Using this fact, we show how to reduce the $n = pq$ case to QS adopting the zero knowledge proof of [28].

The reduction: We use the circuit $C_a(r)$ that on input $r \in Z_n^*$ outputs $z = r^2 a \pmod{n}$. Suppose we know how to generate $|C_a\rangle$ for every a. On input integers n, x, $(n, x) = 1$, the algorithm proceeds as in the proof of Theorem 1.1 with the states $|C_1\rangle, |C_x\rangle$.

Correctness: We have

$$|C_x\rangle = \sum_{z \in Z_n^*} \sqrt{p_z}\, |z\rangle\,,$$

where $p_z = \Pr_{r \in Z_n^*}(z = r^2 x)$, and

$$|C_1\rangle = \alpha \sum_{z : zRn} |z\rangle$$

for $\alpha = \frac{4}{(p-1)(q-1)}$ independent of z.

- If xRn, then $z = r^2 x$ is also a square. Furthermore, $p_z = \Pr_{r \in Z_n^*}(z = r^2 x) = \Pr_r(r \text{ is a square root of } \frac{z}{x})$, and as every square in Z_n^* has the same number of square roots, we conclude that $|C_x\rangle = |C_1\rangle$ and $\langle C_x | C_1 \rangle = 1$.
- Suppose x is not a square. For every $r \in Z_n^*$, $z = xr^2$ must be a nonresidue (or else xRn as well). We conclude that C_x has support only on nonresidues, and so $\langle C_x | C_1 \rangle = 0$.

We note that for a general n, different elements might have a different number of solutions (e.g., try $n = 8$), and so the given construction does not work.

A.3. Approximating CVP. Here we describe the reduction to QS from a gap problem of CVP (closest vector in a lattice), which builds upon the SZK proof of Goldreich and Goldwasser [26]. A lattice of dimension n is represented by a basis, denoted B, which is an $n \times n$ nonsingular matrix over $\mathbb{R}$. The lattice $\mathcal{L}(B)$ is the set of points $\mathcal{L}(B) = \{Bc \mid c \in \mathbb{Z}^n\}$, i.e., all integer linear combinations of the columns of B. The distance $d(v_1, v_2)$ between two points is the Euclidean distance ℓ_2. The distance between a point v and a set $\mathcal{A}$ is $d(v, \mathcal{A}) = \min_{a \in A} d(v, a)$. We also denote $\|S\|$ the length of the largest vector of the set S. The closest vector problem, CVP, gets as input an n-dimensional lattice represented by a basis B and a target vector $v \in \mathbb{R}^n$. The output should be the point $b \in \mathcal{L}(B)$ closest to v.

The CVP problem is NP-hard. Here we are interested in the variant of the problem in which the distance to the lattice is approximated to within a factor g. The approximation problem is known to be NP-hard when g is small, and on the other hand, it is known to be easy when g is exponential (see [6] for exact parameters and references). Here we are interested in the intermediate case, when g is about $\sqrt{\frac{n}{\log(n)}}$. In this range, the problem is not known to be in BPP, but on the other hand, it is known to be in SZK by [26] and therefore is not likely to be NP-hard. We use the SZK proof of [26] to give a reduction to the QS problem. We first describe the promise problem.

The problem:

Input: An n-dimensional lattice given by a basis B, a vector $v \in \mathbb{R}^n$, and a designated distance d. We set $g = g(n) = \sqrt{\frac{n}{c \log n}}$ for any fixed $c > 0$.

Promise:

- Yes: Instances where $d(v, \mathcal{L}(B)) \leq d$.
- No: Instances where $d(v, \mathcal{L}(B)) \geq g \cdot d$.

We let H_t denote the sphere of all points in $\mathbb{R}^n$ of distance at most t from the origin.

The reduction: The circuit C_0 gets as input a random string, and outputs the vector $r + \eta$, where r is a uniformly random point in $H_{2^n \|B \cup \{v\}\|} \cap \mathcal{L}(B)$ and η is a uniformly random point $\eta \in H_{\frac{g}{2} \cdot d}$. Reference [26] explains how to sample such points with almost the right distribution; i.e., they give a description of an efficient such C_0.

We remark that the points cannot be randomly chosen from the real (continuous) vector space, due to precision issues, but [26] shows that taking a fine enough discrete approximation results in an exponentially small error. From now on, we work in the continuous world, bearing in mind that in fact everything is implemented by its discrete approximation. Now assume we can Qsample from the circuit C_0. We can then also Qsample from the circuit C_v, which we define to be the same circuit, except that the outputs are shifted by the vector v and become $r + \eta + v$. To solve the gap problem, the algorithm proceeds as in the proof of Theorem 1.1 with the states $|C_0\rangle, |C_v\rangle$.

Correctness: In a No instance, v is far away from the lattice $\mathcal{L}(B)$, namely, $d(v, \mathcal{L}(B)) \geq g \cdot d$. The calculation in [26] shows that the states $|C_0\rangle$ and $|C_v\rangle$ have no overlap, and so $\langle C_0 | C_v \rangle = 0$. On the other hand, suppose v is close to the lattice, $d(v, \mathcal{L}(B)) \leq d$. Notice that the noise η has magnitude about gd, and so the spheres around any lattice point r and around $r + v$ have a large overlap. Indeed, the argument of [26] shows that if we express $|C_0\rangle = \sum_z p_z |z\rangle$ and $|C_v\rangle = \sum_z p'_z |z\rangle$, then $\|p - p'\| \leq 1 - n^{-2c}$. We see that $\langle C_0 | C_v \rangle = F(p, p') \geq n^{-2c}$. Hence, if we could generate these states, we could iterate the above $poly(n)$ times and get a BQP algorithm for the problem.

Acknowledgments. We wish to thank Umesh Vazirani, Wim van Dam, Zeph Landau, Oded Regev, Dave Bacon, Manny Knill, Eddie Farhi, Ashwin Nayak, and

John Watrous for many inspiring discussions. D.A. is particularly grateful to Erik Winfree for an insightful conversation that initiated this work.

REFERENCES

[1] S. Aaronson, *Quantum lower bound for the collision problem*, in Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing, Montreal, Canada, 2002, pp. 635–642.

[2] D. Aharonov, *Adiabatic quantum computation: Universality and tools*, talk at Mathematical Sciences Research Institute, 2002.

[3] D. Aharonov and A. Ta-Shma, *Adiabatic quantum state generation and statistical zero knowledge*, in Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing, San Diego, CA, 2003, pp. 20–29. A longer version is available online from http://arxiv.org/abs/quant-ph/0210077.

[4] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev, *Adiabatic quantum computation is equivalent to standard quantum computation*, in Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, Rome, Italy, 2004, pp. 42–51.

[5] D. Aharonov and O. Regev, *A lattice problem in quantum NP*, in Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, Cambridge, MA, 2003, pp. 210–219.

[6] D. Aharonov and O. Regev, *Lattice problems in $NP \cap coNP$*, in Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, Rome, Italy, 2004, pp. 362–371.

[7] N. Alon, *Eigenvalues and expanders*, Combinatorica, 6 (1986), pp. 83–96.

[8] A. Ambainis and O. Regev, *An elementary proof of the quantum adiabatic theorem*, http://arxiv.org/abs/quant-ph/0411152 (2006).

[9] D. Applegate and R. Kannan, *Sampling and integration of near log-concave functions*, in Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing, New Orleans, LA, 1991, pp. 156–163.

[10] Y. Avron and A. Elgart, *Adiabatic theorem without a gap condition*, Comm. Math. Phys., 203 (1999), pp. 445–463.

[11] R. B. Boppana, J. Hastad, and S. Zachos, *Does coNP have short interactive proofs?*, Inform. Process. Lett., 25 (1987), pp. 127–132.

[12] R. Bubley and M. Dyer, *Faster random generation of linear extensions*, in Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, 1998, pp. 350–354.

[13] A. M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. A. Spielman, *Exponential algorithmic speedup by quantum walk*, in Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing, San Diego, CA, 2003, pp. 59–68. Also available online from http://arxiv.org/abs/quant-ph/0209131.

[14] A. M. Childs, E. Deotto, E. Farhi, J. Goldstone, S. Gutmann, and A. J. Landahl, *Quantum search by measurement*, Phys. Rev. A, 66 (2002), 032314.

[15] A. M. Childs, E. Farhi, J. Goldstone, and S. Gutmann, *Finding cliques by quantum adiabatic evolution*, http://arxiv.org/abs/quant-ph/0012104 (2000).

[16] A. M. Childs, E. Farhi, and S. Gutmann, *An example of the difference between quantum and classical random walks*, http://arxiv.org/abs/quant-ph/0103020 (2001).

[17] W. van Dam and S. Hallgren, *Efficient quantum algorithms for shifted quadratic character problems*, http://arxiv.org/abs/quant-ph/0011067 (2001).

[18] W. van Dam, M. Mosca, and U. V. Vazirani, *How powerful is adiabatic quantum computation?*, in Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science, Las Vegas, NV, 2001, pp. 279–287.

[19] W. van Dam and U. Vazirani, *More on the Power of Adiabatic Computation*, manuscript, 2001.

[20] E. Farhi, J. Goldstone, and S. Gutmann, *A numerical study of the performance of a quantum adiabatic evolution algorithm for satisfiability*, http://arxiv.org/abs/quant-ph/0007071 (2000).

[21] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, *A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem*, Science, 292 (2001), pp. 472–476.

[22] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, *Quantum computation by adiabatic*

evolution, http://arxiv.org/abs/quant-ph/0001106 (2000).

[23] E. Farhi and S. Gutmann, *Quantum computation and decision trees*, http://arxiv.org/abs/quant-ph/9706062 (1998).

[24] L. Fortnow, *The complexity of perfect zero knowledge*, in Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, New York, NY, 1987, pp. 204–209.

[25] O. Goldreich and E. Kushilevitz, *A perfect zero-knowledge proof system for a problem equivalent to the discrete logarithm*, J. Cryptology, 6 (1993), pp. 97–116.

[26] O. Goldreich and S. Goldwasser, *On the limits of nonapproximability of lattice problems*, in Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, Dallas, TX, 1998, pp. 1–9.

[27] O. Goldreich, A. Sahai, and S. Vadhan, *Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge*, in Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, Dallas, TX, 1998, pp. 399–408.

[28] S. Goldwasser, S. Micali, and C. Rackoff, *The knowledge complexity of interactive proof systems*, SIAM J. Comput., 18 (1989), pp. 186–208.

[29] M. Grötschel and L. Lovász, *Combinatorial optimization*, in Handbook of Combinatorics, Vol. 1, 2, Elsevier, Amsterdam, 1995, pp. 1541–1597.

[30] L. Grover and T. Rudolph, *Creating superpositions that correspond to efficiently integrable probability distributions*, http://arxiv.org/abs/quant-ph/0208112 (2002).

[31] S. Hallgren, *Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem*, in Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing, Montreal, Canada, 2002, pp. 653–658.

[32] M. Jerrum, A. Sinclair, and E. Vigoda, *A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries*, in Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing, Heraklion, Greece, 2001, pp. 712–721.

[33] M. Jerrum and A. Sinclair, *The Markov chain Monte Carlo method: An approach to approximate counting and integration*, in Approximation Algorithms for NP-Hard Problems, D. S. Hochbaum, ed., PWS, Boston, MA, 1996.

[34] T. Kato, *On the adiabatic theorem of quantum mechanics*, J. Phys. Soc. Japan, 5 (1951), pp. 435–439.

[35] A. Yu. Kitaev, *Quantum measurements and the Abelian stabilizer problem*, http://arxiv.org/abs/quant-ph/9511026 (1995).

[36] E. Knill, *private communication.*

[37] J. Kobler, U. Schoning, and J. Turan, *The Graph Isomorphism Problem*, Birkhäuser Boston, Boston, MA, 1993.

[38] L. Lovász, *Random walks on graphs: A survey*, in Combinatorics, Paul Erdös is Eighty, Vol. 2, D. Miklos, V. T. Sos, and T. Szonyi, eds., Janos Bolyai Mathematical Society, Budapest, Hungary, 1996, pp. 353–398.

[39] A. Messiah, *Quantum Mechanics*, John Wiley and Sons, New York, 1958.

[40] M. A. Nielsen and I. Chuang, *Quantum Computation and Information*, Cambridge University Press, Cambridge, UK, 2000.

[41] T. Okamoto, *On relationships between statistical zero knowledge proofs*, in Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, Philadelphia, PA, 1996, pp. 649–658.

[42] A. Peres, *Quantum Theory: Concepts and Methods*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1995.

[43] B. Reichardt, *The quantum adiabatic optimization algorithm and local minima*, in Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing, Chicago, IL, 2004, pp. 502–510.

[44] A. Sahai and S. P. Vadhan, *A complete promise problem for statistical zero-knowledge*, in Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science, Miami Beach, FL, 1997, pp. 448–457.

[45] P. W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM J. Comput., 26 (1997), pp. 1484–1509.

[46] W. L. Spitzer and S. Starr, *Improved bounds on the spectral gap above frustration free ground states of quantum spin chains*, http://arxiv.org/abs/math-ph/0212029 (2002).

[47] S. Vadhan, *A Study of Statistical Zero Knowledge Proofs*, Ph.D. thesis, MIT, Cambridge, MA, 1999.

[48] J. Watrous, *Quantum algorithms for solvable groups*, in Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing, Heraklion, Greece, 2001, pp. 60–67.

SIAM J. COMPUT.
Vol. 37, No. 1, pp. 83–111

A PROBABILISTIC STUDY ON COMBINATORIAL EXPANDERS AND HASHING[*]

PHILLIP G. BRADFORD[†] AND MICHAEL N. KATEHAKIS[‡]

Abstract. This paper gives a new way of showing that certain constant degree graphs are graph expanders. This is done by giving new proofs of expansion for three permutations of the Gabber–Galil expander. Our results give an expansion factor of $\frac{3}{16}$ for subgraphs of these three-regular graphs with $(p-1)^2$ inputs for p prime. The proofs are not based on eigenvalue methods or higher algebra. The same methods show the expected number of probes for unsuccessful search in double hashing is bounded by $\frac{1}{1-\alpha}$, where α is the load factor. This assumes a double hashing scheme in which two hash functions are randomly and independently chosen from a specified uniform distribution. The result is valid regardless of the distribution of the inputs. This is analogous to Carter and Wegman's result for hashing with chaining. This paper concludes by elaborating on how any sufficiently sized subset of inputs in any distribution expands in the subgraph of the Gabber–Galil graph expander of focus. This is related to any key distribution having expected $\frac{1}{1-\alpha}$ probes for unsuccessful search for double hashing given the initial random, independent, and uniform choice of two universal hash functions.

Key words. expander graphs, double hashing, Gabber–Galil expander, expansion factor, combinatorial expanders, pairwise independence, hash collisions

AMS subject classifications. 05C90, 68P05, 68P20, 60C05

DOI. 10.1137/S009753970444630X

1. Introduction.

Consider a bipartite graph $G = (\mathcal{I} \cup \mathcal{O}, E)$, where $\mathcal{I}$ and $\mathcal{O}$ form the bipartition of the nodes with $n = |\mathcal{I}| = |\mathcal{O}|$, and let r be G's maximum degree. G is an (n, r, c)-*expander* if the following holds:

$$|\mathcal{N}(\widehat{\mathcal{I}})| \geq |\widehat{\mathcal{I}}| \left(1 + c\left(1 - \frac{|\widehat{\mathcal{I}}|}{n}\right)\right),$$

for every subset $\widehat{\mathcal{I}} \subset \mathcal{I}$ that contains up to $n/2$ elements (inputs), where $\mathcal{N}(\widehat{\mathcal{I}}) = \{o \in \mathcal{O} : (i, o) \in E \text{ for some } i \in \widehat{\mathcal{I}}\}$. The constant c is the expansion factor of the graph. Further, the degree r is bounded by a constant. There are many other essentially equivalent definitions for graph expanders. The "hard part" in designing graph expanders is proving they expand. In fact, the decision problem of determining expansion is co-$\mathcal{NP}$-complete [6].

A series of classic papers firmly established that certain graph families expand. First, Margulis [21] showed that expanders exist, without giving bounds on their expansion. However, he did show how to construct them explicitly. Next, Gabber and Galil [12] gave an explicit expander construction with bounds on their expansion.

[*] Received by the editors October 15, 2004; accepted for publication (in revised form) November 1, 2006; published electronically March 30, 2007. This research was supported by the Rutgers Business School Research Resources Committee.

http://www.siam.org/journals/sicomp/37-1/44630.html

[†] Department of Computer Science, University of Alabama, Box 870290, Tuscaloosa, AL 35487-0290 (pgb@cs.ua.edu).

[‡] Department of Management Sciences and Information Systems, Rutgers Business School—Newark and New Brunswick, Rutgers University, 180 University Ave., Newark, NJ 07102 (mnk@andromeda.rutgers.edu).

Finally, Alon [2] showed that a graph is an expanding graph iff its largest and second-largest eigenvalues are well separated. See also, for example, [5, 4, 26, 10, 18, 29] for varying depths of coverage of eigenvalue methods for graph expansion. The eigenvalue methods have been central in much research on graph expanders.

Eigenvalue methods do not give the best possible expanding graph coefficients [33]. For example, probabilistic methods show the existence of expanders that have better expansion than is possible to show by the separation of the largest and second-largest eigenvalues. Pinsker [27] first showed the existence of expanders using probabilistic methods.

There are some other constructions of expanders. According to Alon [3], the (eigenvalue-based) construction of Jimbo and Maruoka [15] "only uses elementary but rather complicated tools from linear algebra." Ajtai [1] also gives an algorithm using linear algebra for constructing three-regular expanding graphs. This algorithm is complex and takes $O(n^3 \log^3 n)$ time to construct an expander. The expansion factor of these expanders is unknown but positive. Lubotzky, Phillips, and Sarnak [19] and independently Margulis [22] gave the best possible expanders using the eigenvalue methods [2, 18, 19, 29]. Kahale [16] gave the best expansion constant to date for Ramanujan and related graphs. Reingold, Vadhan, and Wigderson [28] give very important combinatorial constructions of constant degree expanders based on their new "zig-zag" graph product. By showing how the zig-zag product maintains the eigenvalue bounds (then breaks them), they show how to construct expanders recursively starting from a small expander. Further, Meshulam and Wigderson [25] give group theoretic techniques whose expansion they show depends on universal hash functions. Capalbo et al. [7] give constant degree d lossless expanders. These expand by $(1-\epsilon)d$, for $\epsilon > 0$, which is just about as much as possible.

We demonstrate expansion of $\frac{3}{16} = 0.1875$ for three of the five permutations that comprise the Gabber–Galil expander [12]. These results hold for three-regular subgraphs of the Gabber–Galil graphs of p^2 input vertices, where p is a prime. This is done without using Eigenvalue based bounds. The actual Gabber–Galil expansion was shown to be $(2-\sqrt{3})/4$ or about 0.067.

Suppose double hashing is based on randomly, independently, and uniformly choosing two hash functions h_1 and h_2 from a universal set [11]. Then this paper shows the expected number of probes for unsuccessful search in double hashing is bounded by $\frac{1}{1-\alpha}$, where α is the load factor. This holds regardless of the distribution of the inputs. This is analogous to Carter and Wegman's result for hashing with chaining.

1.1. Intuitive overview. Given three permutations of the Gabber–Galil expander graph, this paper shows no matter what subset of inputs (up to half of them) an adversary chooses, then there is at least $\frac{3}{16}$ expansion. This is done in two steps while trading off the local and global structure of the graph. If the adversary allows enough local expansion, then we are done. Therefore, assume the adversary focuses on sufficiently restricting local expansion. In this case, the adversary must choose inputs in certain patterns. Now, in the second step of our main result, it is shown that these patterns cannot block much global expansion.

If the elements are in the appropriate local patterns to minimize local expansion, then the adversary has freedom to choose the number of elements in the patterns as well as where these patterns start. Certain global patterns are *collision sequences* (see Definition 3). Collision sequences reduce the global expansion. Constraining ourselves to local input patterns, the expected length of all of these collision sequences is at most 2, no matter how the adversary chooses to position the local patterns or how many elements the adversary chooses to put in them.

It is essential to note that showing the expected collision sequence length is at most 2 uses probability theory applied to the adversary's constrained selections of input node patterns. Our argument shows the adversary has some very restricted choices of input nodes in the three fixed permutations of Gabber–Galil's graph; otherwise, the adversary allows lots of local expansion. At all times, the three permutations comprising the Gabber–Galil graph remain fixed. The results are given by using probabilistic methods on these fixed graphs.

Further, using virtually the same methods, start by randomly, uniformly, and independently selecting two universal hash functions h_1 and h_2 to build a double hashing table T. All elements will be put in T by double hashing using h_1 and h_2. In this case, let T have fixed load factor $\alpha : 1 > \alpha > 0$. Then we show the expected number of probes for an unsuccessful search in T, still using these initially chosen hash functions, is $\frac{1}{1-\alpha}$. As in the case of our expander result, we show this using probabilistic techniques on fixed graphs.

1.2. Structure of this paper. Section 2 gives details of the three permutations comprising the Gabber–Galil expander and sets the foundations for showing both expansion as well as our hashing result. Section 2 has five subsections. Subsection 2.1 gives the actual graph construction. Next, subsection 2.2 defines local and global expansion. Subsection 2.3 explains the relation of double hashing to the expander graph representation. Next, subsection 2.4 focuses on the results of Chor and Goldreich [9] showing randomly choosing such functions and computing their values gives pairwise independent and uniformly distributed values. Finally, subsection 2.5 bounds functions that are necessary for our final result.

Section 3 uses our methods to show that randomly independently and uniformly selecting two double hash functions from a strongly universal set gives a double hashing result analogous to the classical result of Carter and Wegman [8] for hashing with chaining.

Section 4 completes the expander result, showing the subgraphs expand by $\frac{3}{16}$ by enunciating the trade-off of local and global expansion. Finally, in section 5 we give our conclusions and tie together the notion of expansion with the notion of double hashing with universal hash functions.

2. Combinatorial expanders. This section gives the construction and starts the analysis of expanders without using eigenvalue bounds. Without loss, always assume that $n = |\mathcal{I}| = |\mathcal{O}|$ and $n = p^2$, where p is a prime. Let $\widehat{\mathcal{I}}$ denote the elements from $\mathcal{I}$ that an adversary selects from $\mathcal{I}$ in trying to foil any expansion. The adversary foils an expansion by selecting inputs in such a way so there are relatively few outputs. This section shows that no matter what set $\widehat{\mathcal{I}}$ the adversary chooses, there is expansion.

2.1. The construction. This subsection constructs three-regular bipartite graphs $G_3 = (V, E)$ with vertices $V = \mathcal{I} \cup \mathcal{O}$ denoting the *inputs* and *outputs*, respectively. This graph is made up of permutations σ_0, σ_2, and σ_3 used in building Gabber and Galil's expander [12]. The permutations comprising the Gabber–Galil expander are very similar to the permutations that make up Margulis' expander.

Only inputs can have edges to outputs. Let $\mathbb{Z}_p^+ = \{0, 1, \ldots, p-1\}$. Partition the inputs $\mathcal{I}$ and the outputs $\mathcal{O}$ into p blocks I_j and O_j, for all $j \in \mathbb{Z}_p^+$, containing p

nodes each. In particular, for any $j : p > j \geq 0$,

$$\begin{aligned} I_j &= \{\ (j,0),\ (j,1),\ \ldots,\ (j,p-1)\ \}, \\ O_j &= \{\ (j,0)',\ (j,1)',\ \ldots,\ (j,p-1)'\ \}. \end{aligned}$$

For notational convenience let (j,k) denote the kth element of both lists I_j and O_j for all $j,k \in \mathbb{Z}_p^+$.

As an example, consider $p = 3$ in Figure 1.

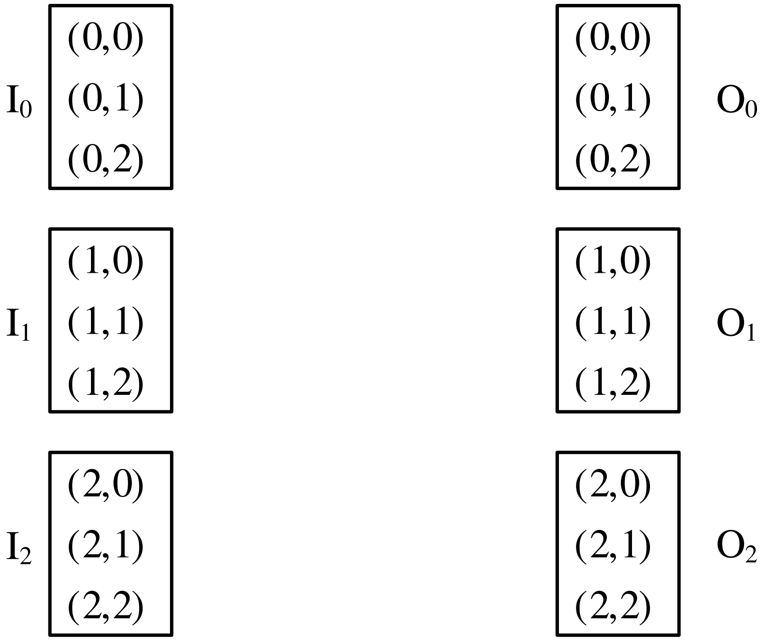


FIG. 1. *The nodes in G_3 where $p = 3$.*

Now take $\mathcal{I}$ as

$$\mathcal{I} = \bigcup_{j=0}^{p-1} I_j.$$

Likewise, for $\mathcal{O}$,

$$\mathcal{O} = \bigcup_{j=0}^{p-1} O_j.$$

For any input node $(j,k) \in I_j$ such that $j \in \mathbb{Z}_p^+$ and $k \in \mathbb{Z}_p^+$, the graph G_3 has the following edges:

1. *Identity*: $\boldsymbol{id}(j,k) \ \longrightarrow\ (j,k)$.
2. *Local shift*: $\boldsymbol{loc}(j,k) \ \longrightarrow\ (j,\ (j+k+1) \bmod p)$.
3. *Global shift*: $\boldsymbol{g}(j,k) \ \longrightarrow\ ((j+k) \bmod p,\ k)$.

These edges are directed from the inputs to the outputs. This does not affect the expansion since it is measured from how the inputs expand to the outputs. Likewise, these directed edges are consonant with the hashing result given in this paper.

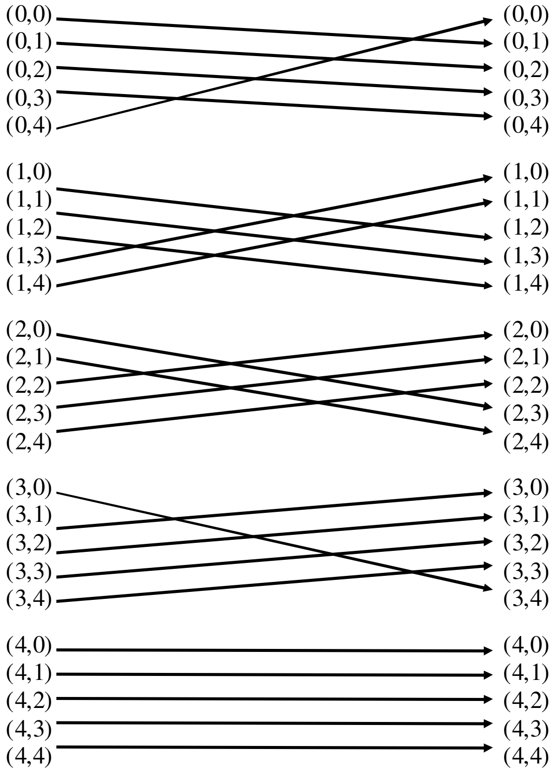


Fig. 2. *Local edges in* G_3 *where* $p = 5$.

Figure 2 gives local edges for G_3 where $p = 5$, and Figure 3 gives global edges for input blocks I_0 and I_1 in G_3 where $p = 5$. The identity edges are not shown in either of these figures. Also see Gabber and Galil [12] or, for example, Motwani and Raghavan [26]. Note in block I_{p-1} the local shift edges degenerate as $\boldsymbol{loc}(p-1,k) = (p-1,k)$ for all $k \in \mathbb{Z}_p^+$. Likewise, in nodes $(j,0)$ the global shift edges degenerate as $\boldsymbol{g}(j,0) = (j,0)$ for all $j \in \mathbb{Z}_p^+$. Therefore, these nodes $(j,0)$ for $\boldsymbol{g}$ and $(p-1,k)$ for $\boldsymbol{loc}$ do not share all of the necessary properties for expansion. Generally, this paper assumes the adversary does not select these degenerate elements. However, after the main theorems, Theorems 8 and 9, an accounting is made assuming the adversary does select degenerate elements.

These maps are well defined on sets. So $\boldsymbol{id}(S) \cup \boldsymbol{loc}(S) \cup \boldsymbol{g}(S) = \mathcal{N}(S) \subseteq \mathcal{O}$ for any set of inputs $S \subseteq \mathcal{I}$. Further, $\boldsymbol{g}_1(j,k), \boldsymbol{loc}_1(j,k)$, and $\boldsymbol{id}_1(j,k)$ denote the first component of the pair, while $\boldsymbol{g}_2(j,k), \boldsymbol{loc}_2(j,k)$, and $\boldsymbol{id}_2(j,k)$ denote the second element. An instance of this subcase of the Gabber–Galil expander is

$$G_3 = (\mathcal{O} \cup \mathcal{I}, \boldsymbol{id}(\mathcal{I}) \cup \boldsymbol{loc}(\mathcal{I}) \cup \boldsymbol{g}(\mathcal{I})).$$

2.2. The analysis. An adversary, who knows G_3's construction, selects sublists $\widehat{I_j}$ from each block I_j. A sublist may be empty. This paper shows that no matter what elements the adversary selects, the graph G_3 expands. This paper assumes up

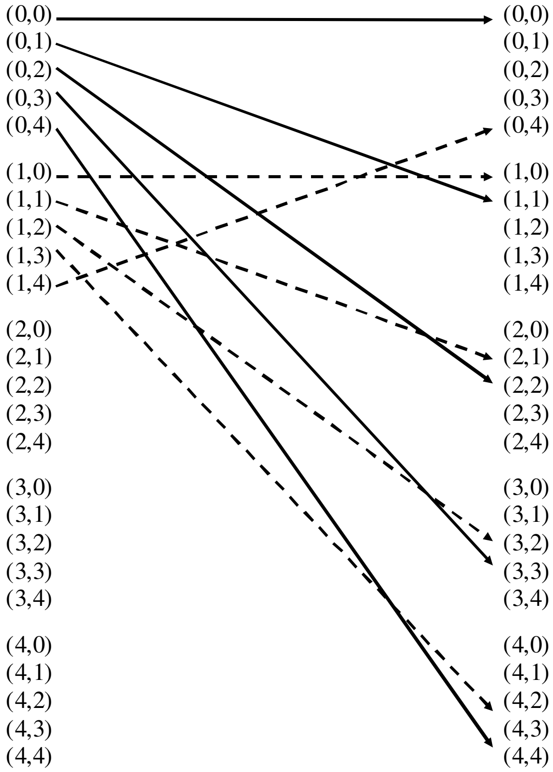


Fig. 3. *Global edges in G_3, from I_0 and I_1, where $p = 5$.*

to half of the inputs to be chosen by the adversary:

$$\sum_{j=0}^{p-1} |\widehat{I}_j| \leq \left\lfloor \frac{n}{2} \right\rfloor .$$

Let

$$\widehat{\mathcal{I}} = \bigcup_{j=0}^{p-1} \widehat{I}_j .$$

Definition 1. *Given block I_j, for $j \in \mathbb{Z}_p^+$, the local $\mathcal{L}$ and global $\mathcal{G}$ expansions of I_j are*

$$\mathcal{L}(\widehat{I}_j) = |\boldsymbol{loc}(\widehat{I}_j) - \boldsymbol{id}(\widehat{I}_j)|,$$
$$\mathcal{G}(\widehat{I}_j) = |\boldsymbol{g}(\widehat{\mathcal{I}}) \cap O_j - \boldsymbol{id}(\widehat{I}_j)|.$$

Definition 1 immediately gives

$$\begin{aligned} \mathcal{L}(\widehat{\mathcal{I}}) &= \sum_{j=0}^{p-1} \mathcal{L}(\widehat{I}_j) \\ &= |\boldsymbol{loc}(\widehat{\mathcal{I}}) - \boldsymbol{id}(\widehat{\mathcal{I}})| \end{aligned}$$

and

$$\begin{aligned}\mathcal{G}(\widehat{\mathcal{I}}) &= \sum_{j=0}^{p-1} \mathcal{G}(\widehat{I_j}) \\ &= |\boldsymbol{g}(\widehat{\mathcal{I}}) - \boldsymbol{id}(\widehat{\mathcal{I}})|.\end{aligned}$$

Local and global expansion can "collide" in that output nodes that give local expansion can also give global expansion. That is, there may be some $\widehat{\mathcal{I}'} \subseteq \widehat{\mathcal{I}}$, where $\boldsymbol{loc}(\widehat{\mathcal{I}'}) = \boldsymbol{g}(\widehat{\mathcal{I}'})$. In this case, to compute the total expansion of $\widehat{\mathcal{I}'}$ just divide $\mathcal{L}(\widehat{\mathcal{I}'}) + \mathcal{G}(\widehat{\mathcal{I}'})$ by 2. Likewise, if local expansion and global expansion share output nodes, then just consider the case that offers more expansion (if they do not offer the same expansion).

For ease of exposition, when possible we generally refer to the elements of the inputs $\mathcal{I}$ from here on. Each input is directly associated with the element that it maps to by the identity mapping.

DEFINITION 2. *In a block $\widehat{I_j}$, for some $j \in \mathbb{Z}_p^+$, the element $(j, k_1) \in \widehat{I_j}$ is* ***loc**-*contiguous *iff* $\boldsymbol{loc}(j, k_1) = \boldsymbol{id}(j, k_2)$ *for* $k_2 = (j + k_1 + 1) \bmod p$ *and* $(j, k_2) \in \widehat{I_j}$. *A* ***loc**-*contiguous set *is a list* $(j, k_1), (j, k_2), \ldots, (j, k_t)$ *all in* $\widehat{I_j}$ *and* $\boldsymbol{loc}(j, k_s) = \boldsymbol{id}(j, k_{s+1})$ *for all* $s : t > s \geq 1$.

If $\mathcal{L}(\widehat{I_j}) \leq 1$, for some $j \in \mathbb{Z}_p^+$, then the elements in $\widehat{I_j}$ are ***loc**-*contiguous.

LEMMA 1. *If there exists a fixed* $d : 1 \geq d > 0$, *where* $d|\widehat{I_j}| \geq |\boldsymbol{id}(\widehat{I_j}) \cap \boldsymbol{loc}(\widehat{I_j})|$, *for all blocks* I_j *such that* $j \in \mathbb{K} \subseteq \mathbb{Z}_p^+$, *where* $\mathbb{K} \neq \emptyset$ *and* $\widehat{I_j} \neq \emptyset$, *then* $\mathcal{L}(\widehat{\mathcal{I}}) \geq (1 - d) \sum_{j \in \mathbb{K}} |\widehat{I_j}|$.

Proof. First, since $|\boldsymbol{id}(\widehat{I_j}) \cap \boldsymbol{loc}(\widehat{I_j})| \leq d|\widehat{I_j}|$ so $|\boldsymbol{loc}(\widehat{I_j}) - \boldsymbol{id}(\widehat{I_j})| \geq (1 - d)|\widehat{I_j}|$, therefore it must be that $\mathcal{L}(\widehat{I_j}) \geq (1 - d)|\widehat{I_j}|$. Since $\boldsymbol{loc}(\widehat{I_j}) \subseteq I_j$, this proof generalizes for the index set $\mathbb{K}$. □

DEFINITION 3. *A* collision sequence *of* length t *is the maximal sequence of elements* $(j_1, k), \ldots, (j_t, k)$, *where* $t \geq 1$, *such that* $(j_i, k) \in \widehat{I_{j_i}}$, *for all* $i \in \{1, \ldots, t\}$ *and* $\boldsymbol{g}(j_0, k) \notin \widehat{I_{j_0}}$ *and* $(j_{t+1}, k) \notin \widehat{I_{j_{t+1}}}$, *where*

$$\begin{aligned}\boldsymbol{g}(j_0, k) &\longrightarrow (j_1, k), \\ \boldsymbol{g}(j_1, k) &\longrightarrow (j_2, k), \\ &\vdots \\ \boldsymbol{g}(j_t, k) &\longrightarrow (j_{t+1}, k).\end{aligned}$$

So $\mathbf{Length}((j_1, k)) = t$.

For example, if $(j_1, k) \in \widehat{I_{j_1}}$, but $\boldsymbol{g}(j_1, k) \notin \widehat{I_t}$, where $t = \boldsymbol{g}_1(j_1, k)$, then (j_1, k) is a length 1 collision sequence starting in input block I_{j_1}. Therefore, a collision sequence of length 1 starts and ends in the same block. Length 1 collision sequences do not diminish expansion but rather increase it.

DEFINITION 4. *Suppose the elements* $(j_s, k) \in \widehat{I_{j_s}}$ *for all* $s : t \geq s \geq 1$ *form a collision sequence. The collision sequence* $(j_1, k) \rightarrow \cdots \rightarrow (j_t, k)$ ends *in block* I_{j_t} *if* $(j_{t+1}, k) \notin \widehat{I_{j_{t+1}}}$ *and* $\boldsymbol{g}(j_t, k) \rightarrow (j_{t+1}, k)$.

Collision sequences prevent global expansion. That is, if we have "many" long collision sequences, then there is "not much" opportunity for global expansion.

Definition 5. *Consider $(j,k_0),(j,k_1)$, and (j,k_2) all from I_j so that $(j,k_1) = \boldsymbol{loc}(j,k_0)$ and $(j,k_2) = \boldsymbol{loc}(j,k_1)$, where*

$$\begin{aligned}&(j,k_0) \notin \widehat{I}_j,\\ &(j,k_1) \in \widehat{I}_j \quad \textit{so } (j,k_1) \textit{ is selected},\\ &(j,k_2) \notin \widehat{I}_j;\end{aligned}$$

then (j,k_1) is a singleton. *A singleton has local expansion of* 1.

Definition 5 is about elements in the same input block I_j. A collision sequence has one or more selected inputs that are all in different input blocks. In fact, a length t collision sequence containing u singletons gives total expansion of at least $u+1$.

The degenerate elements $(j,0)$ for all $j \in \mathbb{Z}_p^+$ do not have global expansion since $\boldsymbol{g}(j,0) = (j,0)$. This means if an adversary can select an element $(j,0)$ to extend a ***loc***-contiguous set in $\widehat{I}_j$, then they should do it since it will not give any global expansion. That is, as long as $(j,0)$ would not be a singleton, then selecting it increases the number of elements selected but does not increase any expansion.

Likewise, the degenerate elements $(p-1,k)$ for all $k \in \mathbb{Z}_p^+$ do not have local expansion since $\boldsymbol{loc}(p-1,k) = (p-1,k)$. Therefore, if selecting a $(p-1,k)$ either extends one collision sequence or joins two collision sequences, then an adversary should select it. Selecting such an $(p-1,k)$ will increase the number of selected elements without increasing expansion. In fact, if $(p-1,k)$ joins two collision sequences, then it reduces the overall expansion.

2.3. Double hashing. Hashing with *open addressing* is a storage and search technique on a table T that assumes the number of elements to be stored in the table is at most the table size: $|T|$. Elements or keys are put directly in the table T. No pointers or data structures are used. There is a special element NIL denoting no element in a position it occupies. Given t elements in the table T, the load factor is $\alpha = \frac{t}{|T|}$ and $\alpha < 1$. Generally, the important questions that have arisen for open address hashing are related to the number of probes necessary to find elements in the table.

Consider an open addressing table T of size m and two hash functions h_1 and h_2. Given a key x, determining the $(i+1)$st hash location using *double hashing* is done by

$$h(i,x) = (h_1(x) + i\,h_2(x)) \bmod m.$$

Double hashing is a classical data structure, and discussions of it can be found in [11, 24, 17], for example.

Inserting the element x into the table T is done by first searching for x in T. If T does not contain x, then x can be inserted into T. Likewise, to delete x from T, then it must be determined if x is in T as well as where x is located in T. Therefore, searching for an element x is the focus of studies of double hashing.

The first probe to T is to position $T[h_1(x) \bmod m]$. If $T[h_1(x) \bmod m] = \mathsf{NIL}$, then x is not in T. Otherwise, if x is in $T[h_1(x) \bmod m]$, then double hashing reports where x is: position $h_1(x) \bmod m$ since $i = 0$. If x is not in $T[h_1(x) \bmod m]$, then the next element probed is $T[(h_1(x)+h_2(x)) \bmod m]$ since $i = 1$. If $T[(h_1(x)+h_2(x)) \bmod m]$ is NIL, then x is not in T. Otherwise, if $x = T[(h_1(x)+h_2(x)) \bmod m]$, then the double hashing algorithm is found where x resides and returns the value $(h_1(x)+h_2(x)) \bmod m$. Otherwise, x may still be in T. Therefore, element $T[(h_1(x) + 2\,h_2(x)) \bmod m]$ is probed, etc. This continues using the function in the $(i+1)$st probe $h(i,x)$ until either x is found or $T[(h_1(x) + i\,h_2(x)) \bmod m]$ is NIL, indicating x is not in T. In summary, the probe sequence is in the following addresses of T:

$$h_1(x),\ (h_1(x) + h_2(x)) \bmod m, \quad (h_1(x) + 2\,h_2(x)) \bmod m,\ \ldots\,.$$

Assume $m > 2$ is prime and h_1 and h_2 are based on ***loc*** and $\boldsymbol{g}$. For a double hash table T, this paper assumes $|T| = m$ as well as the key $x \in \mathbb{Z}_m$.

In the case of this paper's double hashing result, the $\boldsymbol{g}$ edges are the focal point and the ***loc*** edges are not used. In this double hashing scheme, say the pair (j,k) is generated for some key x by h_1 and h_2. That is, start in position k in input block I_j. Hash function h_1 generates the first position j (input block) and hash function h_2 generates the hop-size $k+1$ (how to travel from input block to input block). So the key x is hashed into I_j, starting at local position k. In other words, the first probe is in $T[j]$. If necessary, the second probe is in $T[(j + (k+1)) \bmod m]$. If necessary, the third probe is in $T[(j + 2(k+1)) \bmod m]$, etc.

More precisely, first, a block j_0 and a position k are chosen by h_1 and h_2, respectively. That is, given the key x, compute $j_0 = h_1(x)$ and $k = h_2(x)$. Next, as necessary, the following blocks are computed: $j_1 = \boldsymbol{g}_1(j_0, k)$ and, in general, $j_i = \boldsymbol{g}_1(j_{i-1}, k)$, for $i : m-1 \leq L \geq i \geq 1$, giving the permutation

$$\langle j_0, \boldsymbol{g}_1(j_0,k), \boldsymbol{g}_1(j_1,k), \ldots, \boldsymbol{g}_1(j_L,k)\rangle.$$

Since m is prime, $\boldsymbol{g}$ sends this permutation exactly once through each of the input blocks $\widehat{I}_0, \widehat{I}_1, \ldots, \widehat{I}_L$, where $L \leq m-1$.

The graph G_3, since m a prime, with the functions $\boldsymbol{g}$ represents all permutations used by open addressed double hashing on a table $T[0,\ldots,m-1]$. Of course, $T[j]$ corresponds to I_j.

Double hashing approximates uniform open address hashing [26, 11, 24]. More precisely, Guibas and Szemerédi [14] showed unsuccessful searches using double hashing take asymptotically the same number of probes as idealized uniform hashing does for any fixed load factor α less than about 0.319. For any fixed $\alpha < 1$, see Lueker and Molodowitch [20]. However, as pointed out in Schmidt and Siegel [30], these last results assume ideal randomized functions, whereas [30] utilizes more realistic k-wise independent and uniform functions (where $k = c \log n$ for a suitable constant c).

THEOREM 1 (see [20] and [30]). *Suppose T has load factor of any fixed $\alpha < 1$. The expected number of probes for an unsuccessful search in an open addressing double hashing table is $\frac{1}{1-\alpha} + \epsilon$, where ϵ is a lower-order term.*

Lueker and Molodowitch [20] give the most straightforward method of showing this based on assumed randomized inputs. Schmidt and Siegel [30] give the tightest bound (sharpest bound on ϵ) and the weakest notion of randomness to date. That is, [30] shows the result of Theorem 1 by supplying randomized hash functions, in particular randomized hash functions of degree $c \log n$ for some constant c, giving $c \log n$-wise independent functions.

2.3.1. Strong universal hash functions. Given a graph G_3, where $\alpha = |\widehat{\mathcal{I}}|/|\mathcal{I}|$ for fixed $\alpha : 1 > \alpha > 0$, let $|\widehat{I}_j|/p = \alpha_j$, such that $j \in \mathbb{Z}_p^+$, and each fixed $\alpha_j : 1 > \alpha_j > 0$. This means

$$\begin{aligned}\alpha &= |\widehat{\mathcal{I}}|/|\mathcal{I}| \\ &= \frac{\alpha_0 + \cdots + \alpha_{p-1}}{p}.\end{aligned}$$

Consider any block $\widehat{I}_j$ such that $\mathcal{L}(\widehat{I}_j) \leq 1$. In such blocks the adversary chooses the starting point b_j for the elements of I_j, as well as the total number of elements to select from I_j, expressed here as α_j. More precisely, since $\mathcal{L}(\widehat{I}_j) \leq 1$, the adversary must have chosen the inputs so that $|\boldsymbol{id}(\widehat{I}_j) \cap \boldsymbol{loc}(\widehat{I}_j)| \leq 1$, leaving only the number of elements selected and their starting point to question.

Definition 6 (Carter and Wegman [8]). *The set of function H is strongly universal iff randomly, uniformly, and independently choosing $h \in H$; then for any two different keys x_1 and x_2 and any two values $y_1, y_2 \in \mathbb{Z}_p^+$, it must be that*

$$\mathbf{Pr}[h(x_1) = y_1] = \frac{1}{p} \qquad \textit{and}$$
$$\mathbf{Pr}[h(x_1) = y_1 \wedge h(x_2) = y_2] = \frac{1}{p^2}.$$

Theorem 2 (Carter and Wegman [8]). *The functions*

$$h_{j,b}(x) = jx + b \bmod p \textit{ for all } (j, b) \in \mathbb{Z}_p^+ \times \mathbb{Z}_p^+$$

give the strongly universal set

$$H = \{h_{j,b} \textit{ for all } (j, b) \in \mathbb{Z}_p^+ \times \mathbb{Z}_p^+\}.$$

For all $h \in H$, the range is $\mathbb{Z}_p^+$. Generally, hash functions are expressed as $h_{j,b}(i) \bmod m$, where m is the table size, but here $m = p$, allowing the focus to be entirely on $h_{j,b}(i)$.

2.4. Counting frequencies of selected elements. The basic progression from this subsection to section 3 works as follows. We start with the case where an adversary selects the same number of input elements in each position in all I_j, for $j : j \in \mathbb{Z}_p^+$, while maintaining $\boldsymbol{loc}$-contiguity of the elements in each $\widehat{I}_j$. Basic bounds on the expansion are developed in this subsection. Subsequent subsections in section 2 incrementally allow an adversary to select any elements they choose as long as they maintain $\boldsymbol{loc}$-contiguity.

This subsection applies to both universal hashing as well as expansion.

Theorem 3 (Chor and Goldreich [9]). *Take $h_{j,b} \in H$ uniformly at random; then the associated values $h_{j,b}(i), \ldots, h_{j,b}(0)$, for $p > L \geq i \geq 1$ and $L \geq 2$, are pairwise independent and for all $i \geq 0$ the elements $h_{j,b}(i), \ldots, h_{j,b}(0)$ are uniform in $\mathbb{Z}_p^+$.*

Chor and Goldreich present this result for the sequences of random variables $h_{j,b}(i), \ldots, h_{j,b}(1)$, and it is straightforward that $h_{j,b}(0)$ can be included since $h_{j,b}(0) = b$, which is uniformly and randomly chosen.

Theorem 3 will be applied to $\boldsymbol{g}$ functions between different blocks. Relations *in* the blocks are discussed next. Recall, for each block I_j, the adversary chooses each b_j as well as α_j, and so Theorem 3 does not apply to each block. That is, Theorem 3 assumes the pair $(j, b) \in \mathbb{Z}_p^+ \times \mathbb{Z}_p^+$ is randomly and uniformly chosen.

In contrast to the strongly universal set H of Theorem 2, take $U \subseteq \mathbb{Z}_p^+ \times \mathbb{Z}_p^+$ such that, for all $j \in \mathbb{Z}_p^+$, there is some pair $(j, b) \in U$ and further, if $(j, b_1) \in U$ and $(j, b_2) \in U$, then $b_1 = b_2$. So

$$H' = \{h_{j,b_j}, \text{ for all } (j, b_j) \in U\}, \text{ where } b_j \text{ depends on } j.$$

Note that $|H| = p(p-1)$ and $|H'| = p-1$.

So, in our situation, selecting pairs from H' uniformly at random does not satisfy the hypothesis of this theorem because the adversary chooses each b_j in each pair $(j, b_j) \in H'$.

Definition 7. *Now, for $k \in \mathbb{Z}_p^+$, denote the frequency*

$$n_k = \sum_{j=0}^{p-1} \delta((j,k) \in \widehat{I}_j),$$

where δ is the indicator function, and so $\delta(\mathbf{true}) = 1$ and $\delta(\mathbf{false}) = 0$.

That is, n_k is the frequency of k being selected in all blocks given the adversary's choices of the b_j's and the α_j's.

Note that if $n_k = 0$, then k does not contribute to expansion or lack of expansion. Further, if $n_k = 1$, then k must contribute to global expansion by one.

Aggregating the frequencies gives

$$\alpha = \frac{1}{p^2} \sum_{k=0}^{p-1} n_k.$$

Lemma 2. *Suppose $n_1 = n_2 = \cdots = n_{p-1}$ and assume $\mathcal{L}(\widehat{I}_j) \leq 1$ for all $j \in \mathbb{Z}_p^+$. Take any randomly and uniformly chosen $(J_1, K) \in \mathbb{Z}_p^+ \times \mathbb{Z}_p^+$, where $(J_2, K) = \boldsymbol{g}(J_1, K)$; then*

$$\mathbf{Pr}[(J_1,K) \in \widehat{I}_{J_1} \wedge (J_2,K) \in \widehat{I}_{J_2}] \leq \frac{n_k^2}{p^2}.$$

Proof. Assume $(J_1, K) \in \mathbb{Z}_p^+ \times \mathbb{Z}_p^+$ is randomly and uniformly chosen. So we are considering the collision sequence,

$$\mathcal{C}_{J_1,K} = (J_1,K) \to (J_2,K),$$

where $(J_2, K) = \boldsymbol{g}(J_1, K)$.

This gives

$$\mathbf{Pr}[(J_1,K) \in \widehat{I}_{J_1} \wedge (J_2,K) \in \widehat{I}_{J_2}]$$

$$= \frac{1}{p^2} \sum_{j=0}^{p-1} \sum_{k=0}^{p-1} \mathbf{Pr}[(j,k) \in \widehat{I}_j \wedge \boldsymbol{g}(j,k) \in \widehat{I}_{\boldsymbol{g}_1(j,k)}]$$

$$= \frac{1}{p^3} \sum_{k=0}^{p-1} \sum_{j=0}^{p-1} \delta((j,k) \in \widehat{I}_j)\, \mathbf{Pr}[\boldsymbol{g}(j,k) \in \widehat{I}_{\boldsymbol{g}_1(j,k)} \mid (j,k) \in \widehat{I}_j]$$

$$= \frac{1}{p^3} \sum_{k=0}^{p-1} n_k \sum_{j=0}^{p-1} \mathbf{Pr}[\boldsymbol{g}(j,k) \in \widehat{I}_{\boldsymbol{g}_1(j,k)}] \text{ by independence and uniformity of Theorem 3}$$

$$= \sum_{k=0}^{p-1} \frac{n_k^2}{p^3}$$

$$\leq \frac{n_k^2}{p^2}.$$

This completes the proof. □

If all $n_k \leq \alpha p$, then

$$\sum_{k=0}^{p-1} \frac{n_k^2}{p^3} \leq \alpha^2.$$

In other words, let

$$\mathbf{Average}[n_k^2] = \sum_{k=0}^{p-1} \frac{n_k^2}{p}.$$

If $n_k \leq \lceil \alpha p \rceil$, for all $k : p > k \geq 0$, then

$$\mathbf{Average}[n_k^2] \leq \alpha^2 \, p^2$$

so that for uniformly and randomly chosen $(J_1, K) \in \mathbb{Z}_p^+ \times \mathbb{Z}_p^+$ and $(J_2, K) = \boldsymbol{g}(J_1, K)$, then

$$\begin{aligned}\mathbf{Pr}[(J_1, K) \in \widehat{I}_{J_1} \wedge (J_2, K) \in \widehat{I}_{J_2}] &\leq \frac{\mathbf{Average}[n_k^2]}{p^2} \\ &\leq \alpha^2.\end{aligned}$$

Furthermore, if there is a set

$$\mathbb{K} = \{k_0, \ldots, k_r\}$$

and $|\mathbb{K}| \leq p^{\delta}$, where $\delta : 1 > \delta \geq 0$, such that

$$n_k > \lceil \alpha(p-1) \rceil \text{ for all } k \in \mathbb{K},$$

then, letting $\overline{\mathbb{K}} = \mathbb{Z}_p^+ - \mathbb{K}$, this gives $n_{k'} \leq \lceil \alpha \, p \rceil$ for all $k' \in \overline{\mathbb{K}}$. This means

$$\begin{aligned}\mathbf{Average}[n_k^2] &= \sum_{k \in \mathbb{K}} \frac{n_k^2}{p} + \sum_{k' \in \overline{\mathbb{K}}} \frac{n_{k'}^2}{p} \\ &\leq \frac{p^{\delta} p^2}{p} + \frac{\alpha^2 \, p^2 (p - p^{\delta})}{p} \\ &\leq p^{\delta} p + \alpha^2 \, p^2.\end{aligned}$$

Therefore, if $|\mathbb{K}| \leq p^{\delta}$ for any $\delta : 1 > \delta \geq 0$, and for uniformly and randomly chosen $(J_1, K) \in \mathbb{Z}_p^+ \times \mathbb{Z}_p^+$ and $(J_2, K) = \boldsymbol{g}(J_1, K)$, then

$$\mathbf{Pr}[(J_1, K) \in \widehat{I}_{J_1} \wedge (J_2, K) \in \widehat{I}_{J_2}] \leq \alpha^2 + O(p^{\delta - 1}) \text{ for } \delta : 1 > \delta \geq 0.$$

2.5. Bounding Average[$\boldsymbol{n_k^2}$] when $\boldsymbol{n_k > \lceil \alpha \, p \rceil}$ for $\boldsymbol{k \in \mathbb{Z}_p^+}$. Consider the maximal subset $\mathbb{K} \subseteq \mathbb{Z}_p^+$, where all $k' \in \mathbb{K}$ are such that the frequencies $n_{k'} > \lceil \alpha \, p \rceil$ when $\mathcal{L}(\widehat{\mathcal{I}}) < \frac{3}{16} |\widehat{\mathcal{I}}|$.

Start with the case $\mathcal{L}(\widehat{I}_j) \leq 1$ for all $j \in \mathbb{Z}_p^+$. This subsection shows for all $k' \in \mathbb{K} \subseteq \mathbb{Z}_p^+$ such that $n_{k'} > \lceil \alpha \, p \rceil$ there can be a total of at most $p \lceil 1/\alpha \rceil$ total selected elements in all collision sequences of length more than $\lceil \alpha \, p \rceil$ for all $k \in \mathbb{Z}_p^+$.

Definition 8. *Let $\alpha = |\widehat{\mathcal{I}}|/|\mathcal{I}|$. If there is some $k \in \mathbb{Z}_p^+$ so that k's frequency n_k is such that*

$$n_k > \lceil \alpha \, p \rceil,$$

then there are $n_k - \lceil \alpha \, p \rceil$ excess *elements selected in the kth position of $\mathcal{I}$.*

2.5.1. The case with up to 1 excess element selected. Here k_0 denotes a single excess element.

DEFINITION 9. *Let $n_{k,L}$ denote any n_k when all $j_r \in \{j_0, \ldots, j_{t-1}\}$ are all such that $|\widehat{I}_{j_r}| = L$.*

Further, by Definition 9 it must be that $\alpha' = L/p$ and all n_k's throughout the rest of this section are associated only with the blocks indexed by $\{j_0, \ldots, j_{t-1}\}$.

For the next lemma recall h_{j_i} is a hash function representing the ***loc*** functions corresponding to $\widehat{I}_{j_i}$.

LEMMA 3. *Let $\mathcal{L}(\widehat{I}_j) \leq 1$ for all $j \in \mathbb{Z}_p^+$. Assume k_0 is selected in all of $I_{j_0}, \ldots, I_{j_{t-1}}$ and $|\widehat{I}_{j_r}| = L \leq p$, for $j_r \in \{j_0, \ldots, j_{t-1}\}$, $t \geq L \geq 2$, with $h_{j_0}(v) = \cdots = h_{j_{t-1}}(v) = k_0$ for some $v \in \mathbb{Z}_p^+$; then $n_k \leq \lceil (\alpha' - \frac{1}{p})\, p \rceil$ for all $k \neq k_0$ and $\alpha' = L/p$.*

Proof. The proof is by induction on the size of $|\widehat{I}_{j_r}| = L$. For the moment, assume $v = 0$ for the induction; this will be generalized after the induction is complete.

Basis. Consider the case where $|\widehat{I}_{j_r}| = 2 = L$ for all $j_r \in \{j_0, \ldots, j_{t-1}\}$, where $t \geq L$, and $h_{j_0}(0) = \cdots = h_{j_{t-1}}(0) = k_0$. Then all elements $u_r = (j_r + 1 + k_0) \bmod p$, for all $r : t > r \geq 0$, are distinct since $\{j_0, \ldots, j_{t-1}\}$ are distinct and $t \geq 2$. That is, if $u_{r_x} = u_{r_y}$, then

$$(j_x + 1 + k_0) \bmod p = (j_y + 1 + k_0) \bmod p,$$

and so $j_x = j_y \bmod p$, and it must be that $j_x < p$ and $j_y < p$, and thus $x = y$, a contradiction. So $n_k = 1$, for all $k \neq k_0$, and $\alpha' = \frac{2}{p}$. Further, all $k \neq k_0$ are such that $n_k \leq \lceil (\alpha' - \frac{1}{p})\, t \rceil$, completing the basis.

Inductive hypothesis. Suppose $|\widehat{I}_{j_r}| = L \geq 2$, where $j_r \in \{j_0, \ldots, j_{t-1}\}$ and $t \geq L$. Then $n_{k,L} \leq \lceil (\alpha' - \frac{1}{p})\, t \rceil$ for all $k \neq k_0$ and $\alpha' = \frac{L}{p}$.

Inductive step. Suppose $|\widehat{I}_{j_r}| = L+1$, where $j_r \in \{j_0, \ldots, j_{t-1}\}$ and $t \geq L+1 \geq 3$, where α' is associated with $n_{k,L+1}$. Then by the inductive hypothesis the first L elements in each block share the property that all but one of the $n_{k,L}$'s are such that $n_{k,L} \leq \lceil (\alpha' - \frac{1}{p})\, t \rceil$. Adding one element to each block and each in a unique position gives $n_{k,L+1} \leq \lceil (\alpha' + \frac{1}{p} - \frac{1}{p})\, t \rceil$. Each element is put in a unique position since $t \geq L$, and no two elements among $u_r = (L+1)(j_r+1) + k_0 \bmod p$, for $r \in \{t-1, \ldots, 0\}$, are the same: If $j_x \neq j_y$, where

$$((L+1)(j_x+1) + k_0) = ((L+1)(j_y+1) + k_0) \bmod p,$$

then since each element of a field (mod-p) has a multiplicative and additive inverse, we must have $j_x = j_y \bmod p$, a contradiction since $j_x < p$ and $j_y < p$. Therefore, applying the inductive hypothesis completes the induction.

Now we show this lemma holds for any $v \in \mathbb{Z}_p^+$, where $h_{j_0}(v) = \cdots = h_{j_{t-1}}(v) = k_0$. Consider $|\widehat{I}_{j_r}| = L$ for all $j_r \in \{j_0, \ldots, j_{t-1}\}$ and $t \geq L$, and given some $v \neq 0$, then break the problem into two cases: The first case consists of all elements $h_{j_0}(i), \ldots, h_{j_{t-1}}(i)$ for $i : L > i \geq v$. The second case consists of all elements $h_{j_0}(i'), \ldots, h_{j_{t-1}}(i')$ for $i' : v \geq i' \geq 0$. (Note that these cases overlap since they both have k_0 in common.)

Now, treating these cases separately, apply the induction above with $\alpha_1 = \frac{L-v}{p}$ to the $L - v$ elements of $h_{j_0}(i), \ldots, h_{j_{t-1}}(i)$ for $i : L > i \geq v$.

Likewise, for each $h_{j_0}(i'), \ldots, h_{j_{t-1}}(i')$, where $i' : v \geq i' \geq 0$, apply the induction above to the v elements. Here $\alpha_2 = \frac{v+1}{p}$.

Now $\alpha' = \alpha_1 + \alpha_2 - \frac{1}{p}$ since k_0 was counted twice. With this in mind, take the following inequalities, where $k \neq k_0$:

$$\begin{aligned} n_k &\leq \left\lceil \left(\alpha_1 - \frac{1}{p}\right) t \right\rceil + \left\lceil \left(\alpha_2 - \frac{1}{p}\right) t \right\rceil \\ &\leq \left\lceil \left(\alpha_1 + \alpha_2 - \frac{2}{p}\right) p \right\rceil \\ &\leq \left\lceil \left(\alpha' + \frac{1}{p} - \frac{2}{p}\right) p \right\rceil \\ &\leq \left\lceil \left(\alpha' - \frac{1}{p}\right) p \right\rceil, \end{aligned}$$

completing the proof. □

With a little work, Lemma 3 generalizes to Theorem 4. Assume $\mathcal{L}(\widehat{I}_j) \leq 1$, for all $j \in \mathbb{Z}_p^+$ and k_0, is selected in each block $I_0, \ldots, I_{p-2}$. Let L_u be the number of elements selected going "above" and including k_0. Similarly, let L_d be the number of elements selected going "down" from k_0 but not including k_0. (If $k_0 = h(i)$, then $h(i+c)$ is "above" for any integer c, where $i + c < p$, and $h(i-c)$ is "down," where $i - c \geq 0$.) The induction is about the same; the only difference is the proof of Theorem 4 assumes the relation

$$n_k \leq \lceil (\alpha_u + \alpha_d)\, t \rceil$$

holds before the inductive step. More precisely, suppose $t = \lceil \frac{p}{c} \rceil$ for some integer c, and by definition $\alpha_u + \alpha_d = \frac{L_u + L_d}{p}$, meaning

$$\left(\frac{L_u + L_d}{p}\right) t \leq \frac{L_u + L_d}{c}.$$

So there are a total of $L_u + L_d$ elements selected per input block, and $\frac{1}{c}$ bounds the percent of blocks under consideration. The inductive hypothesis says $\lceil \frac{L_u + L_d}{c} \rceil$ is an upper bound on the number of elements for each $k \neq k_0$.

Now consider adding one element to each block going "up" and one element to each block going "down." That is, increase α_u to $\alpha_u + \frac{1}{p}$ and increase α_d to $\alpha_d + \frac{1}{p}$, but at the same time none of the new "up" elements collide with each other and none of the new elements going "down" collide with each other. That is, for all $r \in \{0, 1, \ldots, t-1\}$,

$$u_r = (L_u + 1)(j_r + 1) \bmod p$$

are all different by the uniqueness of multiplicative inverses in $\mathbb{Z}_p^+ - \{0\}$. Likewise, for all $r \in \{0, 1, \ldots, t-1\}$,

$$d_r = (L_d + 1)(j_r + 1) \bmod p$$

are all different by the uniqueness of multiplicative inverses in $\mathbb{Z}_p^+ - \{0\}$. Furthermore, by the uniqueness of multiplicative inverses in $\mathbb{Z}_p^+ - \{0\}$, for each d_i there can be at most one u_j so that $d_i = u_j$, where $i, j \in \{0, 1, \ldots, t-1\}$. Consider increasing L_u to $L_u + 1$ and increasing L_d to $L_d + 1$, and assume $t = \lfloor \frac{p}{c} \rfloor$, for some integer c, giving

$$\left(\frac{L_u + L_d + 2}{p}\right) t \leq \frac{L_u + L_d + 2}{c},$$

which is the number of selected elements per input block multiplied by the percent of the input blocks. Therefore, $n_k \le \lceil (\alpha_u + \alpha_d)\, t + \frac{2\,t}{p} \rceil$, giving, for all $k \neq k_0$,

$$n_k \le \left\lceil \left(\alpha_u + \alpha_d + \frac{2}{p} \right) t \right\rceil , \tag{1}$$

which clearly holds for $t > p/2$ and $(\alpha_u + \alpha_d)\, t$ bounded by an integer. In the case where $t < p/2$, then since $L_u + L_d + 2$ elements were selected per input block and their ***loc***-continuity gives for each $k \neq k_0$, by the inductive hypothesis $c\, n_k \le L_u + L_d$ and by the uniqueness of multiplicative inverses, n_k can increase no more than 2 when both L_u and L_d are increased by 1 each. That is, now $c\, n_k \le L_u + L_d + 2$ holds, completing the inductive step. This gives the next theorem.

THEOREM 4. *Let $\mathcal{L}(\widehat{I}_j) \le 1$ for all $j \in \mathbb{Z}_p^+$. Assume k_0 is selected in all of $I_{j_0}, \ldots, I_{j_{t-1}}$ and $|\widehat{I}_{j_r}| = L \le p$, for $j_r \in \{j_0, \ldots, j_{t-1}\}$, $t \ge L \ge 2$, with $h_{j_0}(v) = \cdots = h_{j_{t-1}}(v) = k_0$ for some $v \in \mathbb{Z}_p^+$; then $n_k \le \lceil \alpha'\ t \rceil$ for all $k \neq k_0$ and $\alpha' = L/p$.*

The next lemma allows any of t blocks to have any number of elements L selected as long as $t \ge L$. That is, $|\widehat{I}_{j_r}| \le L$ for all $j_r \in \{j_0, \ldots, j_{t-1}\}$ and $t \ge L$.

LEMMA 4. *Let $\mathcal{L}(\widehat{I}_{j_r}) \le 1$ for all $j_r \in \{j_0, \ldots, j_{t-1}\}$. Assume k_0 is selected in all of $I_{j_0}, \ldots, I_{j_{t-1}}$ and $|\widehat{I}_{j_r}| \le L \le p$ for $j_r \in \{j_0, \ldots, j_{t-1}\}$ and $t \ge L \ge 2$ while $h_{j_0}(v) = \cdots = h_{j_{t-1}}(v) = k_0$ for some $v \in \mathbb{Z}_p^+$; then $n_k \le \lceil \alpha'\ t \rceil$ for all $k \neq k_0$ and $\alpha' = (|\widehat{I}_{j_0}| + \cdots + |\widehat{I}_{j_{t-1}}|)/(t\, p)$.*

Proof. Consider two sets T_1 and T_2 with $s : t \ge s \ge 0$, so that

$$T_1 = \{\ \widehat{I}_{j_0}, \ldots, \widehat{I}_{j_{s-1}}\ \}, \quad \text{where} \ \ \alpha_1\, p = |\widehat{I}_{j_k}| \text{ for all } k : s-1 \ge k \ge 0,$$

and

$$T_2 = \{\ \widehat{I}_{j_s}, \ldots, \widehat{I}_{j_{t-1}}\ \}, \quad \text{where} \ \ \alpha_2\, p = |\widehat{I}_{j_k}| \text{ for all } k : t-1 \ge k \ge s.$$

Therefore, there is a total of $\alpha'\, p\, t$ selected elements in all of the blocks

$$\{\ \widehat{I}_{j_0}, \ldots, \widehat{I}_{j_{t-1}}\ \},$$

and so

$$\alpha'\, p\, t = \alpha_1\, p\, s + \alpha_2\, p\, (t - s).$$

That is,

$$\alpha'\, t = \alpha_1\, s + \alpha_2\, (t - s).$$

Without loss, assume $\alpha_1 < \alpha_2$ and T_1 represents the first s input blocks. Now, applying Theorem 4 to all t input blocks considering only $\alpha_{\min} = \min\{\ \alpha_1, \alpha_2\ \}$, it must be that

$$n_k^0 \le \lceil \alpha_{\min}\ \ t \rceil$$

for all $k \neq k_0$, and each n_k^0 is computed restricting k to the $\alpha_{\min}\, t$ elements of each of all t input blocks. Note that Theorem 4 applies to the first $\alpha_{\min}\, t$ ***loc***-contiguous elements from each block since $t \ge L$. Without loss, assume $\alpha_{\min}\, t$ is an integer.

Now, letting $\alpha_{\max} = \max\{\ \alpha_1, \alpha_2\ \}$, then since $t \ge L$ and assuming $t = \lceil \frac{p}{c} \rceil$ for some integer c, then applying Theorem 4

$$n_k^1 \le \lceil (\alpha_{\max} - \alpha_{\min})\ t \rceil$$

for all $k \neq k_0$. But, not considering the $\lceil \alpha_{\min} t \rceil$ elements, it must be that

$$n_k^1 \leq \lceil (\alpha_{\max} - \alpha_{\min})\ (t-s) \rceil\,.$$

Without loss, assume that $(\alpha_{\max} - \alpha_{\min})\ (t-s)$ and $\alpha_{\min}\ t$ are integers. This means, for $k \neq k_0$, and since the elements represented by $\alpha_{\max}$ and $\alpha_{\min}$ are ***loc***-contiguous,

$$\begin{aligned} n_k^0 + n_k^1 &\leq \lceil \alpha_{\min} t + (\alpha_{\max} - \alpha_{\min})\ (t-s) \rceil \\ &\leq \lceil \alpha_{\max} (t-s) - \alpha_{\min} (t-s) + \alpha_{\min} t \rceil \\ &\leq \lceil \alpha_{\max} (t-s) + \alpha_{\min} s \rceil \\ &\leq \lceil \alpha_1 s + \alpha_2 (t-s) \rceil \\ &\leq \lceil \alpha' t \rceil\,, \end{aligned}$$

since $\alpha_{\max} = \alpha_2$ and $\alpha_{\min} = \alpha_1$ and $\alpha' t = \alpha_1 s + \alpha_2 (t-s)$, while at the same time $n_k^0 + n_k^1 > \lceil (\alpha_1 + \alpha_2)(t-s) \rceil$ only for $k = k_0$; the proof is completed by induction on the number of sets of blocks, each set containing blocks with the same number of elements selected. □

Now consider combining ***loc***-contiguous collision sequences as described by Lemma 4. In particular, now look at bounding the length of all collision sequences in G_3 by combining different ***loc***-contiguous subblocks.

The next definition generalizes Definition 2.

DEFINITION 10. *Take $j \in \mathbb{Z}_p^+$. The set $U_{j,s}$ is a* maximal ***loc***-contiguous subblock *of $\widehat{I}_j$ if $U_{j,s} \subset \widehat{I}_j$. And if $|U_{j,s}| \geq 2$, while $\mathcal{L}(U_{j,s}) \leq 1$, and for any $U' \subset \widehat{I}_j : U_{j,s} \subset U'$ and $U_{j,s} \neq U'$, then $\mathcal{L}(U') > 1$. Further, if $U_{j,0} = I_j$ (so $|U_{j,0}| = p$), then $U_{j,0}$ is the only* maximal ***loc***-contiguous subblock *of $\widehat{I}_j$.*

Note that a maximal ***loc***-contiguous subblock $U_{j,s}$ must contain at least two elements; otherwise, it is not ***loc***-contiguous. Further, a block $\widehat{I}_j$ may have many maximal ***loc***-contiguous subblocks. Allow maximal ***loc***-contiguous subblocks to be empty. This means maximal ***loc***-contiguous subblocks cannot consist of a single ***loc***-contiguous element.

DEFINITION 11. *Consider the collision sequence $\mathcal{C}_1$ made up of selected elements from the max **loc**-contiguous subblocks $U_{0,0}, \ldots, U_{p-1,0}$. Another collision sequence $\mathcal{C}_2$* overlaps *with $\mathcal{C}_1$ iff $\mathcal{C}_2$ consists of elements from at least one of the same max **loc**-contiguous subblocks $U_{0,0}, \ldots, U_{p-1,0}$.*

So now remove from consideration all subblocks associated with any collision sequence in c_0, i.e., $U_{0,0}, \ldots, U_{p-1,0}$. Now with the remaining elements, put the next largest collision sequences in a set c_1. Any collision sequence $\mathcal{C}_r \in c_1$ is associated with the sets of max ***loc***-contiguous sequences $U_{0,r}, \ldots, U_{p-1,r}$. By Lemma 4 and Theorem 4 all elements of any other collision sequence can share no more than $\lceil \alpha_1 p \rceil$ elements with $U_{0,r}, \ldots, U_{p-1,r}$, where $\alpha_1 = (|U_{0,r}| + \cdots + |U_{p-1,r}|)/p^2$.

This argument extends to all collision sequences larger than $\lceil \alpha p \rceil$.

LEMMA 5. *Let $\mathbb{K}$ be the set of indices of all s collision sequences larger than $\lceil \alpha p \rceil$. Suppose the adversary selects no singletons and $\alpha_i = (|U_{0,i}| + \cdots + |U_{p-1,i}|)/p^2$, where n_k^i is n_k* restricted *to the jth set of max **loc**-contiguous subblocks $U_{j,i}$, for i, where $i : s \geq i \geq 0$ and all $j : p-1 \geq j \geq 0$. So for any $k \in \mathbb{Z}_p^+ - \mathbb{K}$, then*

$$n_k^0 + \cdots + n_k^s \leq \left\lceil \left(\alpha_0 - \frac{1}{p}\right) p \right\rceil + \left\lceil \left(\alpha_1 - \frac{1}{p}\right) p \right\rceil + \cdots + \left\lceil \left(\alpha_s - \frac{1}{p}\right) p \right\rceil.$$

Proof. Without loss, let $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_s$ be collision sequences of lengths larger than $\lceil \alpha\, p \rceil$. Assume these are listed largest ($\mathcal{C}_0$) to smallest ($\mathcal{C}_s$). No two collision sequences $\mathcal{C}_i$ and $\mathcal{C}_j$, for $i \neq j$, are made up of elements from the same max ***loc***-contiguous subblocks $U_{j_0,i'}, \ldots, U_{j_{k-1},i'}$ for some i', by Lemma 4 and Theorem 4. In the case where two collision sequences share a max ***loc***-contiguous subblock, then this max ***loc***-contiguous subblock can be cut into two ***loc***-contiguous subblocks. Using this fact, starting with the largest collision sequences first, each collision sequence is uniquely associated with a set of $p-1$ ***loc***-contiguous subblocks, one for each input block I_j, for $j \in \mathbb{Z}_p^+$. (Some of these ***loc***-contiguous subblocks may be empty.)

This means each collision sequence is associated with one ***loc***-contiguous subblock from each input block,

$$\langle U_{0,0}, \ldots, U_{p-1,0} \rangle\,, \quad \langle U_{0,1}, \ldots, U_{p-1,1} \rangle\,, \ldots, \langle U_{0,s}, \ldots, U_{p-1,s} \rangle\,,$$

and in some cases $U_{j,i} = \{\emptyset\}$.

Let $\mathbb{K} = \{k_0, \ldots, k_s\} \subset \mathbb{Z}_p^+$ be such that $k \in \mathbb{K}$ means $n_k \geq \lceil \alpha\, p \rceil$, and now take it as $n_k = p-1$. If $k \in \mathbb{Z}_p^+ - \mathbb{K}$ and n_k^i is restricted to $U_{0,i}, \ldots, U_{p-1,i}$, where $\alpha_i = (|U_{0,i}| + \cdots + |U_{p-1,i}|)/p(p-1)$, then

$$n_k^i \leq \left\lceil \left(\alpha_i - \frac{1}{p} \right) p \right\rceil,$$

which gives the lemma. □

The next lemma deals with the case when the number of elements in each block is larger than the total number of such blocks, or $L > t$.

Lemma 6. *Let $\mathcal{L}(\widehat{I}_j) \leq 1$ for all $j \in \mathbb{Z}_p^+$. Given t blocks $I_{j_0}, \ldots, I_{j_{t-1}}$, where $|\widehat{I}_{j_r}| \leq L \leq p$, for $j_r \in \{j_0, \ldots, j_{t-1}\}$ so that $\alpha' = (|\widehat{I}_{j_0}| + \cdots + |\widehat{I}_{j_{t-1}}|)/(t\ p)$, and letting $S = |\boldsymbol{id}_2(\widehat{I}_{j_0}) \cup \cdots \cup \boldsymbol{id}_2(\widehat{I}_{j_{t-1}})|$, now let*

$$T = \left\lceil \frac{S}{t} \right\rceil;$$

then at most T of the n_k's are such that $n_k > \lceil \alpha'\, t \rceil$.

Proof. Take the L elements in each block and consider them in $T = \lceil \frac{S}{t} \rceil$ ***loc***-contiguous sets of inputs of size t each, in every block. If $T = 1$, then we are done. Next, consider each set of t selected ***loc***-contiguous input elements among the t blocks; then assume there is an input set $\{i_0, \ldots, i_{t-1}\}$, so that $h_{j_0}(i_0) = \cdots = h_{j_{t-1}}(i_{t-1})$. If not, then consider the largest such input set for each of the t selected ***loc***-contiguous sets of elements. By Theorem 4 and since there are up to t elements in each set of ***loc***-contiguous elements, then each set of ***loc***-contiguous blocks alone has at most one $n_{k'}$ such that $n_{k'} > \lceil \alpha_s\ t \rceil$, for $\alpha_s = \alpha'/T$, for some k'.

This means, among all T size t ***loc***-contiguous sets among the t input blocks, there will be at most T elements $n_{k'_i} > \lceil \alpha_s\, t \rceil$ for $\alpha_s = \frac{t}{L}\alpha'$ and $i : T-1 \geq i \geq 0$. Let

$$\mathbb{K} = \{\ k_0, \ldots, k_{T-1}\ \}$$

be the set of n_{k_i} elements so that $n_{k_i} > \lceil \alpha_s\, t \rceil$.

Let n_k^ℓ denote n_k restricted to the ℓth set of ***loc***-contiguous blocks. In other words, n_k^ℓ is the number of times k occurs in the ℓth set of ***loc***-contiguous blocks.

Next, without loss, suppose that $\alpha_s t = \frac{L\,t}{p}$ is an integer, and considering all ***loc***-contiguous sets at the same time gives, for any $k_i \notin \mathbb{K}$,

$$\begin{aligned} n_{k_i}^{0} + \cdots + n_{k_i}^{T-1} &\leq \lceil (\alpha_s + \cdots + \alpha_s)\, t \rceil, \text{ where there are } T \text{ total } \alpha_s \\ &\leq \lceil \alpha'\, t \rceil, \end{aligned}$$

by Lemma 5.

This completes the proof. □

Suppose there is a collision sequence of length t made of one selected element from each of $\widehat{I}_{j_0}, \ldots, \widehat{I}_{j_{t-1}}$. If $t \geq \lceil \alpha\, p \rceil$, then Lemma 6 indicates that

$$\begin{aligned} T &\leq \left\lceil \frac{S}{\alpha\, p} \right\rceil \\ &\leq \left\lceil \frac{1}{\alpha} \right\rceil, \end{aligned}$$

since $p \geq S$, and where $S = |\boldsymbol{id}_2(\widehat{I}_{j_0}) \cup \cdots \cup \boldsymbol{id}_2(\widehat{I}_{j_{t-1}})|$. That is, suppose this occurs in a set of t input blocks where $S > t$ and the largest collision sequences are of length at least $\lceil \alpha\, p \rceil$. The only selected elements in excess of $\lceil \alpha\, p \rceil$ in $T = \lceil \frac{S}{\alpha} \rceil$ "large" collision sequences can be larger than $\lceil \alpha\, p \rceil$. Since $T \leq \lceil 1/\alpha \rceil$, if all of these T "large" collision sequences consist of p elements each, there is a total of at most

$$\frac{p - \lceil \alpha\, p \rceil}{\alpha} \leq \frac{p}{\alpha}$$

excess elements out of a total of p^2 possible elements and $\alpha\, p^2$ selected elements. That is, the uniform random probability of selecting an element that extends a collision sequence to one of these excess elements is at most $\frac{1}{\alpha\, p}$.

It is also possible that the adversary chooses $t > \lceil \alpha\, p \rceil$ blocks that have more elements selected in each block than αp^2, where $\alpha = |\widehat{\mathcal{I}}|/|\mathcal{I}|$. Take the case where $\alpha' > \alpha$, where $\alpha = |\widehat{\mathcal{I}}|/|\mathcal{I}|$, and take $\alpha = \frac{L}{p}$ for appropriate L, but say $\alpha' \leq \frac{L+d}{p}$, for some integer d, for all $\widehat{I}_{j_0}, \ldots, \widehat{I}_{j_{t-1}}$. This means

$$T = \left\lceil \frac{L+d}{\lceil \alpha\, p \rceil} \right\rceil,$$

but since $\alpha = \frac{L}{p}$, it must be that $\alpha\, p = L$, and therefore

$$T = 1 + \left\lceil \frac{d}{\lceil \alpha\, p \rceil} \right\rceil.$$

Assuming $d > \lceil \alpha\, p \rceil$, then $T \leq 1 + \lceil \frac{1}{\alpha} \rceil$, since $d < p$, and thus say $d = \frac{p}{c}$ for some number $c \geq 1$; then

$$\begin{aligned} \left\lceil \frac{d}{\lceil \alpha\, p \rceil} \right\rceil &= \left\lceil \frac{p}{c\, \lceil \alpha\, p \rceil} \right\rceil \\ &\leq \frac{1}{c\, \alpha} \\ &\leq \frac{1}{\alpha} \end{aligned}$$

since $1 > \alpha$ and $c \geq 1$. So now we discard any case where $L > t$ by this discussion and Lemma 6.

2.5.2. When more than one excess element is selected in $\widehat{\mathcal{I}}$. The next theorem assumes no more than one ***loc***-contiguous subblock is selected per block I_j for $j \in \mathbb{Z}_p^+$. Therefore, $\mathcal{L}(\widehat{I}_j) \leq 1$ for all $j \in \mathbb{Z}_p^+$, which means at most p total elements of all of the n_k's are such that $n_k > \lceil (\alpha - \frac{1}{p})\, p \rceil$.

Definition 12. *Given the frequency n_k, then $B[n_k] \subseteq \mathbb{Z}_p^+$ is the set of block indices that have element k selected in each of them. That is, $u \in B[n_k]$ iff $\delta((u,k) \in \widehat{I}_u)$.*

Theorem 5. *Suppose $\mathcal{L}(\widehat{I}_j) \leq 1$ for all blocks $j \in \mathbb{Z}_p^+$. Then the total number of excess elements in G_3 is at most $p\lceil 1/\alpha \rceil$.*

Proof. Consider the frequencies, $n_{k_0} \geq n_{k_1} \geq \cdots \geq n_{k_s} > \lceil \alpha\, p \rceil$. By Lemma 5, this proof must consider only the elements $\{k_0, \ldots, k_s\} \in \mathbb{Z}_p^+$. Further, by Lemma 6, at most $T = \lceil 1/\alpha \rceil$ frequencies are such that $|B[n_{k_i}] \cap B[n_{k_j}]| \geq \lceil \alpha p \rceil$ for $k_i \neq k_j$.

Now, if all $k, k' \in \{k_0, \ldots, k_s\}$ are such that

$$B[n_k] \cap B[n_{k'}] \neq \emptyset,$$

then clearly

$$\sum_{i=0}^{s} n_{k_i} \leq p,$$

which would complete the proof.

Furthermore, if any subset $\{u_0, \ldots, u_t\} \subseteq \{k_0, \ldots, k_s\}$ is such that for all $k_i \in \{k_0, \ldots, k_s\}$ and for all $u_i \in \{u_0, \ldots, u_t\}$,

$$B[n_{u_i}] \cap B[n_{k_i}] = \emptyset,$$

then we need only consider the set

$$\mathbb{K} = \{k_0, \ldots, k_s\} - \{u_0, \ldots, u_t\}.$$

Given two distinct collision sequences, then by Lemma 4 these collision sequences can share no more than $\lceil \alpha\, p \rceil$ elements as long as $\mathcal{L}(\widehat{I}_j) \leq 1$.

This means

$$|B[n_{k_0}] \cap (B[n_{k_1}] \cup \cdots \cup B[n_{k_s}])| \leq \lceil \alpha p \rceil,$$

and thus removing the block indices $B[n_{k_0}]$ and by applying Lemma 4 again gives

$$|B[n_{k_1}] \cap (B[n_{k_2}] \cup \cdots \cup B[n_{k_s}])| \leq \lceil \alpha p \rceil.$$

The proof is completed by induction on the remaining blocks $B[n_{k_2}], \ldots, B[n_{k_s}]$. □

That is, if $\mathcal{L}(\widehat{I}_j) \leq 1$, for all $j \in \mathbb{Z}_p^+$, then the total number of excess selected elements is $\lceil 1/\alpha \rceil p$. This means the probability of uniformly and randomly selecting one of these up to p *excess* elements is at most $\lceil \frac{p}{\alpha\, p^2} \rceil = \lceil \frac{1}{\alpha\, p} \rceil$.

Next, this is generalized to the case where $\mathcal{L}(\widehat{\mathcal{I}}) \leq \frac{3}{16}|\widehat{\mathcal{I}}|$. In this case, this subsection concludes by showing for all $k \in \mathbb{Z}_p^+$ that $\mathbf{Average}[n_k^2]$ is bounded so that randomly, independently, and uniformly choosing (J_1, K) from $\mathbb{Z}_p^+ \times \mathbb{Z}_p^+$, where $(J_2, K) = \boldsymbol{g}(J_1, K)$, gives

$$\mathbf{Pr}[(J_1, K) \in \widehat{I}_{J_1} \wedge (J_2, K) \in \widehat{I}_{J_2}] \leq \alpha^2 + \epsilon,$$

where $\epsilon = O(\frac{1}{p})$.

As before, start assuming $\mathcal{L}(\widehat{I}_j) \leq 1$ for all blocks $j \in \mathbb{Z}_p^+$. Take the relations,

$$n_{k_0} \geq n_{k_1} \geq \cdots \geq n_{k_s} > \lceil \alpha\, p \rceil,$$

so that $n_k \leq \lceil \alpha\, p \rceil$ for all $k \in \mathbb{Z}_p^+ - \mathbb{K}$, where $\mathbb{K} = \{\, k_0, \ldots, k_s \,\}$.

Theorem 6. *Suppose $\mathcal{L}(\widehat{I}_j) \leq 1$ for all $j \in \mathbb{Z}_p^+$ and $\alpha = |\widehat{\mathcal{I}}|/|\mathcal{I}|$. Take any randomly and uniformly chosen $(J_1, K) \in \mathbb{Z}_p^+ \times \mathbb{Z}_p^+$, where $(J_2, K) = \boldsymbol{g}(J_1, K)$; then*

$$\mathbf{Pr}[(J_1, K) \in \widehat{I}_{J_1} \wedge (J_2, K) \in \widehat{I}_{J_2}] \leq \alpha^2 + \epsilon,$$

where ϵ is of lower-order terms.

Proof. If there is at most one frequency $n_{k_0} > \lceil \alpha\; p \rceil$, then for all of the $p^2 - p$ or more *non*excess elements in G_3, it must be that

$$\mathbf{Pr}[(J_1, K) \in \widehat{I}_{J_1} \wedge (J_2, K) \in \widehat{I}_{J_2}] \leq \alpha^2$$

holds by Lemma 2.

Consider the frequencies $n_{k_0} \geq n_{k_1} \geq \cdots \geq n_{k_s} > \lceil \alpha\; p \rceil$, where $s \geq 1$. The case of the up to p elements in $n_{k_0}, \ldots, n_{k_s}$, the probability of them extending a one-element collision sequence with excess elements by Theorem 5, is

$$\left\lceil \frac{p}{\alpha p^2} \right\rceil = \left\lceil \frac{1}{\alpha p} \right\rceil.$$

Thus, for all the elements in G_3 it must be that

$$\mathbf{Pr}[(J_1, K) \in \widehat{I}_{J_1} \wedge (J_2, K) \in \widehat{I}_{J_2}] \leq \alpha^2 + O\left(\frac{1}{p}\right)$$

holds.

This completes the proof. □

If $\mathcal{L}(\widehat{I}_j) > 1$ for some $j \in \mathbb{Z}_p^+$, then there may be many collision sequences that must be considered.

Start with G_3 where the adversary has selected $|\widehat{\mathcal{I}}|$ input elements. Then consider any set of at most $\frac{3}{16}|\widehat{\mathcal{I}}|$ sets of associated ***loc***-contiguous elements, where each associated set contains a common collision sequence. By applying Theorem 6 to each collision sequence created by increasing local expansion to decrease global expansion (by extending or joining collision sequences) gives the next theorem. Note that each extended collision sequence has some associated $\alpha_i < \alpha = |\widehat{\mathcal{I}}|/|\mathcal{I}|$ and $\alpha_1 + \cdots + \alpha_u = \alpha$, and so $\alpha_1^2 + \cdots + \alpha_u^2 \leq \alpha^2$. This is because

$$\begin{aligned} \alpha_1^2 + \cdots + \alpha_u^2 &\leq (\alpha_1 + \cdots + \alpha_u)^2 \\ &\leq \alpha^2. \end{aligned}$$

Theorem 7. *Let $\alpha = |\widehat{\mathcal{I}}|/|\mathcal{I}|$. Suppose $\mathcal{L}(\widehat{\mathcal{I}}) < \frac{3}{16}|\widehat{\mathcal{I}}|$ and all selected elements are in a **loc**-contiguous subblocks and there are no singletons. Take any randomly and uniformly chosen $(J_1, K) \in \mathbb{Z}_p^+ \times \mathbb{Z}_p^+$, where $(J_2, K) = \boldsymbol{g}(J_1, K)$; then*

$$\mathbf{Pr}[(J_1, K) \in \widehat{I}_{J_1} \wedge (J_2, K) \in \widehat{I}_{J_2}] \leq \alpha^2 + \epsilon,$$

where ϵ is of lower-order terms.

3. Variations on hashing. Given an open addressing hash table T with a fixed load factor of $\alpha : 1 > \alpha > 0$, assume T is filled using double hashing to load factor α. As discussed in subsection 2.3, double hashing uses two hash functions h_1 and h_2. The goal of this section is to show that if both hash functions h_1 and h_2 are randomly, uniformly, and independently chosen from the strongly universal hash functions H (Definition 6), then the expected cost of an unsuccessful search using double hashing is $\frac{1}{1-\alpha}$ table accesses. This question was suggested by Carter and Wegman [8].

Another important form of hashing is *hashing with chaining*; see, for example, [11, 13, 24]. Carter and Wegman [8] showed that given any strongly universal set of hash functions H, then randomly and uniformly selecting a hash function $h \in H$ gives an expected chain length of at most $1 + \alpha'$ for fixed load factor $\alpha' > 0$. For instance, taking the set of hash functions H with domain and range $\mathbb{Z}_p^+$ as in Definition 6, Carter and Wegman's result is important since the strongly universal set H (of size $O(p^2)$) behaves as if randomly selecting a function from the set of all functions from $\mathbb{Z}_p^+$ to $\mathbb{Z}_p^+$ (of size $O(p^p)$). See Mehlhorn [24, 23] for lower bounds on the sizes of universal hash sets.

As future research they suggest extending such an analysis to double or open hashing. Schmidt and Siegel [30] and Siegel [31] answer this, giving $c \log n$-independent functions that are computable in constant time for a standard word model random access machine. Their results are quite general; see also [32]. Next, we focus on another answer to Carter and Wegman's question using the standard set H of strongly universal hash functions, see Definition 6, as they are represented in the G_3 graph. Although this paper uses a different model, the $\boldsymbol{g}$-edges in G_3 make selecting entire blocks simulate twowise independent functions; see Theorem 3.

The G_3 graph can represent a double hashing configuration if all elements in each input block are either all selected or all unselected. That is, say each block $|\widehat{I}_j|/p = \alpha_j$ is such that either $\alpha_j = 1$ or $\alpha_j = 0$. This gives a fixed load factor $\alpha = |\widehat{\mathcal{I}}|/|\mathcal{I}|$, where $1 > \alpha > 0$. Each entire input block corresponds to a cell in the hash table T, where T is of size p. So, if $\alpha_j = 1$, then $T[j]$ is full; and if $\alpha_j = 0$, then $T[j]$ is empty.

If one wants to build a double hashing table, do this by making two independent and uniform random choices $h_1, h_2 \in H$, where H is the strongly universal set described in subsection 2.4. So, given a key x, the value $j_0 = h_1(x)$ is the first table element $T[j_0]$ to probe. Now, if $T[j_0]$ is full and $T[j_0] \neq x$, then probe the values $T[(j_0 + i\, h_2(x)) \bmod p]$ for $i = 1, \ldots, p-1$, until encountering the key x or an empty table element (NIL).

Next, this paper shows that building a hash table by double hashing with the initial uniform and independent random choices $h_1, h_2 \in H$ gives an open addressing table of load factor α with expected number of probes $\frac{1}{1-\alpha}$ for an unsuccessful search, regardless of the input distribution.

When searching through a hash table for x, a collision sequence equates to a probe sequence $(j_0 + i\, h_2(x)) \bmod p$, given $j_0 \leftarrow h_1(x)$ and h_1, h_2 both randomly, uniformly, and independently chosen from a strongly universal set H.

In this context, consider the collision sequence $\mathcal{C}_{J_1,K}$ starting in position (J_1, K). So let J_1 and K be randomly, uniformly, and independently chosen. Since J_1 is independent of J_2 and $(J_2, K) = \boldsymbol{g}(J_1, K)$, then

$$\begin{aligned}&\mathbf{Pr}[\mathbf{Length}(\mathcal{C}_{J_1,K}) \geq 2]\\ &= \mathbf{Pr}[\mathbf{Length}(\mathcal{C}_{J_1,K}) = 1 \wedge \mathbf{Length}(\mathcal{C}_{J_2,K}) \geq 1 \wedge (J_2, K) = \boldsymbol{g}(J_1, K)].\end{aligned}$$

In the next proof, lower-order terms that would appear if the adversary selected degenerate elements are ignored.

Proposition 1 (hashing collision sequence). *Say T is an open address hash table with any configuration of elements built by initially randomly, uniformly, and independently choosing h_1, h_2 from H and then performing double hashing to insert elements into T. Let $\alpha = |\widehat{\mathcal{I}}|/|\mathcal{I}|$. Assume each block I_j is such that $|\widehat{I_j}|/p = \alpha_j$ and either $\alpha_j = 1$ or $\alpha_j = 0$. Then the expected collision sequence length in T is* $\mathbf{E}[\mathbf{Length}(\mathcal{C})] \leq \frac{\alpha}{1-\alpha}$.

Proof. For any fixed $\mathcal{S} = \{\alpha_{i_0}, \ldots, \alpha_{i_s}\} \subset \{\alpha_0, \ldots, \alpha_{p-2}\}$ let $s < p-2$ and $\alpha_{i_r} = 1$ for all $i_r \in \{i_0, \ldots, i_s\}$ and $\alpha_j = 0$ for all $j \notin \{i_0, \ldots, i_s\}$. This means $\alpha = \frac{s+1}{p}$. In this case, since each full block ($\alpha_{i_r} = 1$) has exactly one edge going to all other blocks, then the fraction $1 - \frac{s+1}{p}$ of all collision sequences starting in any full blocks are of length exactly 1.

Now the claim that $\mathbf{E}[\mathbf{Length}(\mathcal{C})] \leq \frac{\alpha}{1-\alpha}$ is shown by induction.

Let $\mathcal{C}_{J_1,K}$ be a potential collision sequence that passes through at any randomly and uniformly chosen $(J_1, K) \in \mathbb{Z}_p^+ \times \mathbb{Z}_p^+$; then $\mathbf{E}[\mathbf{Pr}[\mathbf{Length}(\mathcal{C}_{J_1,K}) \geq 1]] = \alpha$ since α is the probability of randomly and uniformly selecting an input element.

Basis. Since $\alpha = \frac{s+1}{p}$, then randomly and uniformly choosing $(J_1, K) \in \mathbb{Z}_p^+ \times \mathbb{Z}_p^+$, where $(J_2, K) = \boldsymbol{g}(J_1, K)$, gives

$$\begin{aligned}\mathbf{Pr}[(J_1, K) \in \widehat{I}_{J_1} \wedge (J_2, K) \in \widehat{I}_{J_2}] &= \left(\frac{s+1}{p}\right)\left(\frac{s}{p}\right) \\ &\leq \alpha^2\end{aligned}$$

by Lemma 2 noting that since $\alpha_{i_r} = 1$, for all $\alpha_{i_r} \in \mathcal{S}$, then $n_0 = \cdots = n_{p-1} = \alpha p$. Further, if $T[J_1]$ is full, then $\mathbf{Pr}[(J_2, K) \in \widehat{I}_{J_2}] = \alpha - \frac{1}{p} = \frac{s}{p}$ and $\frac{s}{p} < \alpha$.

Inductive hypothesis. For some $c \geq 2$, and for all $i < c$, assume for any collision sequences $\mathcal{C}_{J_1,K}$ that $\mathbf{Pr}[\mathbf{Length}(\mathcal{C}_{J_1,K}) \geq i] \leq \alpha^i$.

Inductive step. Take $c \geq 2$ and consider $t \leq c$; then we claim for all potential collision sequences $\mathcal{C}_{J_1,K}$, where $(J_2, K) = \boldsymbol{g}(J_1, K)$,

$$\mathbf{Pr}[\mathbf{Length}(\mathcal{C}_{J_1,K}) \geq t+1] = \mathbf{Pr}[\mathbf{Length}(\mathcal{C}_{J_1,K}) = 1]\; \mathbf{Pr}[\mathbf{Length}(\mathcal{C}_{J_2,K}) \geq t].$$

To substantiate this claim, take a *potential* length $t+1$ collision sequence $\mathcal{C}_{J_1,K}$ and suppose the first probe starts in block I_{J_1}, and so

$$\mathcal{C}_{J_1,K} = (J_1, K) \rightarrow (J_2, K) \rightarrow \cdots \rightarrow (J_t, K)$$

and $J_i = \boldsymbol{g}_1(J_{i-1}, K)$, where $c \geq t \geq i > 1$.

It must be that

$$\begin{aligned}\mathbf{Pr}[\mathbf{Length}(\mathcal{C}_{J_1,K}) \geq t+1] &= \mathbf{Pr}[\mathbf{Length}(\mathcal{C}_{J_1,K}) = 1]\; \mathbf{Pr}[\mathbf{Length}(\mathcal{C}_{J_2,K}) \geq t] \\ &\leq \alpha\; \mathbf{Pr}[\mathbf{Length}(\mathcal{C}_{J_2,K}) \geq t],\end{aligned}$$

which holds by pairwise independence from Theorem 3 and, further, by the inductive hypothesis $\mathbf{Pr}[\mathbf{Length}(\mathcal{C}_{J_2,K}) \geq t] \leq \alpha^t + \epsilon$, completing the induction.

Now, for any $t \geq 1$, since the random variable $\mathbf{Length}(\mathcal{C}_{J_1,K})$ is nonnegative, this means

$$\begin{aligned}\mathbf{E}[\mathbf{Length}(\mathcal{C})] &= \sum_{t \geq 1} \mathbf{Pr}[\mathbf{Length}(\mathcal{C}) \geq t] \\ &\leq \sum_{t \geq 1} \alpha^t \\ &\leq \frac{\alpha}{1-\alpha}.\end{aligned}$$

This completes the proof. □

Suppose $t < p - 1$ elements are in an open addressing hashing table T, where $|T| = p$, that is filled with $t = \alpha p$ elements. Such a configuration represents the elements of T that are filled with load factor $\alpha = \frac{t}{p}$. In addition, by Proposition 1, the expected collision sequence length is $\frac{\alpha}{1-\alpha}$, no matter how the table T is filled. In the next theorem, α is the load factor of the table T.

THEOREM 8 (main double hashing theorem). *Say T is an open address hash table with any configuration of elements built by initially randomly, uniformly, and independently choosing h_1, h_2 from H and then performing double hashing to insert elements into T. Now an unsuccessful search for a key x in T using double hashing with h_1 and h_2 has expected cost of at most $1 + \mathbf{E}[\mathbf{Length}(\mathcal{C})] = \frac{1}{1-\alpha}$ hash probes.*

Proof. Suppose $t = \alpha p$ different keys $x_1, \dots, x_t$ have been inserted into the table T using the randomly, independently, and uniformly chosen $h_1, h_2 \in H$. Then there are $t = \alpha p$ blocks $I_{j_1}, \dots, I_{j_t}$ that have $\alpha_{j_r} = 1$ for all $j_r \in \{j_1, \dots, j_t\}$. That is, the table T has load factor α. Note that $\alpha_{j'_r} = 0$ for all $j'_r \in \mathbb{Z}_p^+ - \{j_1, \dots, j_t\}$.

Now suppose we are searching for a key $x \notin \{x_1, \dots, x_t\}$ in T given the hash functions h_1, h_2. Since H is strongly universal, it must be that for any $x_i \in \{x_1, \dots, x_t\}$, then

$$\mathbf{Pr}[h_1(x) = h_1(x_i)] = \frac{1}{p},$$
$$\mathbf{Pr}[h_2(x) = h_2(x_i)] = \frac{1}{p}.$$

This means, for $x_i \in \{x_1, \dots, x_t\}$ and $x \notin \{x_1, \dots, x_t\}$, that

$$\begin{aligned}\mathbf{Pr}[h_1(x) = h_1(x_i) \wedge h_2(x) = h_2(x_i)] &= \mathbf{Pr}[h_1(x) = h_1(x_i)]\, \mathbf{Pr}[h_2(x) = h_2(x_i)] \\ &= \frac{1}{p(p-1)}.\end{aligned}$$

Therefore, the probing sequence for x has equal probability $(\frac{1}{p(p-1)})$ of starting at any $(J_1, K) \in \mathbb{Z}_p^+ \times (\mathbb{Z}_{p-1}^+ - \{0\})$. Thus, applying Proposition 1, the expected collision sequence length is $\frac{\alpha}{1-\alpha} + 1 = \frac{1}{1-\alpha}$ for any $x \notin \{x_1, \dots, x_t\}$ and x not in T.

This completes the proof. □

This last theorem assumes any (j, k) is not of the form $(j, 0)$ for any $j \in \mathbb{Z}_p^+$. Allowing such elements to be selected from $\mathbb{Z}_p^+ \times \mathbb{Z}_{p-1}^+$ would add a lower-order term of $O(\frac{1}{p})$ to $\frac{1}{1-\alpha}$.

4. Showing the graph G_3 expands. Suppose $\mathcal{L}(\widehat{I}_j) \leq 1$ for all $j \in \mathbb{Z}_p^+$. By Theorem 6 for randomly and uniformly chosen $(J_1, K) \in \mathbb{Z}_p^+ \times \mathbb{Z}_p^+$, then

$$\mathbf{Pr}[(J_1, K) \in \widehat{I}_{J_1} \wedge (J_2, K) \in \widehat{I}_{J_2}] \leq \alpha^2 + \epsilon,$$

where $(J_2, K) = \boldsymbol{g}(J_1, K)$ and ϵ is a lower-order term.

Let L_1 be the set of collision sequences of length at least 1. Suppose $(J_1, K) \in \mathbb{Z}_p^+ \times \mathbb{Z}_p^+$ is randomly and uniformly chosen and it so happens that $(J_1, K) \in L_1$; then $\mathbf{Pr}[\mathbf{Length}(\mathcal{C}_{J_1,K}) \geq 1 \mid (J_1, K) \in L_1] = 1$, and $\mathbf{Pr}[\mathbf{Length}(\mathcal{C}_{J_1,K}) \geq 2 \mid (J_1, K) \in L_1] = \alpha$. This last equality holds because $(J_1, K) = \boldsymbol{g}(J_0, K)$ and

$$\begin{aligned}\mathbf{Pr}[(J_1, K) \in \widehat{I}_{J_1} \wedge (J_2, K) \in \widehat{I}_{J_2} \mid (J_1, K) \in \widehat{I}_{J_1}] &= \mathbf{Pr}[(J_2, K) \in \widehat{I}_{J_2} \mid (J_1, K) \in \widehat{I}_{J_1}] \\ &= \mathbf{Pr}[(J_2, K) \in \widehat{I}_{J_2}],\end{aligned}$$

and the last equality is by pairwise independence of Theorem 3. Furthermore, $\mathbf{Pr}[(J_2, K) \in \widehat{I}_{J_2}] = \alpha$ since (J_2, K) is independent of (J_1, K), making (J_2, K) randomly and uniformly chosen from $\mathbb{Z}_p^+ \times \mathbb{Z}_p^+$.

In the next proof, lower-order terms that would appear if the adversary selected degenerate elements are ignored.

Proposition 2 (general collision sequence length). *Take G_3 so that $\alpha = |\widehat{\mathcal{I}}|/|\mathcal{I}|$, where α is fixed and $1 > \alpha > 0$, and $\mathcal{L}(\widehat{I}_j) \leq 1$ for all $j \in \mathbb{Z}_p^+$; then the expected collision sequence length is* $\mathbf{E}[\mathbf{Length}(\mathcal{C})] \leq 1 + \frac{2\,\alpha}{1-\alpha}$.

Proof. The fact that a collision sequence traveling through $(J_1, K) \in \widehat{I}_{J_1}$ has expected remaining length via $\boldsymbol{g}$ of $\mathbf{E}[\mathbf{Length}(\mathcal{C}^r_{J_1,K})] \leq \frac{\alpha}{1-\alpha}$ is proved by induction. After the induction, the proof accounts for the expected prior collision sequence length $\mathbf{E}[\mathbf{Length}(\mathcal{C}^p_{J_1,K})]$ going to (J_1, K). (Note that $\mathcal{C}^p$ is *prior* collision sequence and $\mathcal{C}^r$ is the *remaining* collision sequence.)

Basis. Assuming $(J_1, K) \in \mathbb{Z}_p^+ \times \mathbb{Z}_p^+$ is randomly and uniformly chosen and $(J_1, K) \in \widehat{I}_{J_1}$ and $(J_2, K) = \boldsymbol{g}(J_1, K)$, then

$$\begin{aligned}
&\mathbf{Pr}[\mathbf{Length}(\mathcal{C}^r_{J_1,K}) \geq 2 \mid (J_1, K) \in \widehat{I}_{J_1}] \\
&= \mathbf{Pr}[\mathbf{Length}(\mathcal{C}^r_{J_1,K}) \geq 1 \wedge \mathbf{Length}(\mathcal{C}^r_{J_2,K}) \geq 1 \mid (J_1, K) \in \widehat{I}_{J_1}] \\
&= \mathbf{Pr}[(J_2, K) \in \widehat{I}_{J_2}] \\
&\leq \alpha,
\end{aligned}$$

by the pairwise independence of J_1 and J_2 and the uniformity of (J_2, K), since $(J_2, K) = \boldsymbol{g}(J_1, K)$ by Theorem 3 and by the choice of (J_1, K).

Inductive hypothesis. For some $c \geq 2$, and for all $i < c$, assume for any potential collision sequence $\mathcal{C}^r_{J_1,K}$ starting in block I_{J_1}, so that $(J_1, K) \in \widehat{I}_{J_1}$, and traveling via $\boldsymbol{g}$ that $\mathbf{Pr}[\mathbf{Length}(\mathcal{C}^r_{J_1,K}) \geq i \mid (J_1, K) \in \widehat{I}_{J_1}] \leq \alpha^{i-1} + \epsilon$ for $i > 1$ and for $i = 1$; then $\epsilon = 0$.

Inductive step. Take $c \geq 2$ and consider $t \leq c$; then take a potential length $t + 1$ collision sequence starting with $(J_1, K) \in \widehat{I}_{J_1}$ and traveling via $\boldsymbol{g}$ so for $i : t \geq i \geq 0$,

$$\mathcal{C}^r_{J_1,K} = (J_1, K) \to (J_2, K) \to \cdots \to (J_t, K)$$

and $J_i = \boldsymbol{g}_1(J_{i-1}, K)$, where $c \geq t \geq i > 1$ and $(J_1, K) \in \mathbb{Z}_p^+ \times \mathbb{Z}_p^+$. It must be that

$$\begin{aligned}
&\mathbf{Pr}[\mathbf{Length}(\mathcal{C}^r_{J_1,K}) \geq t + 1 \mid (J_1, K) \in \widehat{I}_{J_1}] \\
&= \mathbf{Pr}[\mathbf{Length}(\mathcal{C}^r_{J_1,K}) \geq 1]\, \mathbf{Pr}[\mathbf{Length}(\mathcal{C}^r_{J_2,K}) \geq t] \\
&= \mathbf{Pr}[\mathbf{Length}(\mathcal{C}^r_{J_2,K}) \geq t],
\end{aligned}$$

which holds by pairwise independence by Theorem 3 and by Theorem 6 and, further, by the inductive hypothesis $\mathbf{Pr}[\mathbf{Length}(\mathcal{C}^r_{J_2,K}) \geq t] \leq \alpha^{t-1} + \epsilon$, completing the induction.

Now, without conditioning on the first element of a collision sequence being selected, say for any $t \geq 1$, since the random variable $\mathbf{Length}(\mathcal{C}^r_{J_1,K})$ is nonnegative, this means

$$\begin{aligned}\mathbf{E}[\mathbf{Length}(\mathcal{C}^r)] &= \sum_{t\geq 1}\mathbf{Pr}[\mathbf{Length}(\mathcal{C}^r)\geq t]\\ &\leq \sum_{t\geq 1}\alpha^t\\ &\leq \frac{\alpha}{1-\alpha}.\end{aligned}$$

Therefore, conditioning on the first element of a collision sequence being selected gives a bound of $1+\frac{\alpha}{1-\alpha}$.

The induction above is based on starting at a random uniformly chosen (J_1, K) and going via $\boldsymbol{g}$. The issue of the prior expected collision sequence length $\mathbf{E}[\mathbf{Length}(\mathcal{C}^p_{J_1,K})] = \frac{\alpha}{1-\alpha}$ is dealt with by a symmetric argument. This means the expected collision sequence length is

$$\begin{aligned}\mathbf{E}[\mathbf{Length}(\mathcal{C}_{J_1,K})] &= \mathbf{E}[\mathbf{Length}(\mathcal{C}^p_{J_1,K})] + \mathbf{E}[\mathbf{Length}(\mathcal{C}^r_{J_1,K})]\\ &= \frac{2\,\alpha}{1-\alpha}.\end{aligned}$$

Finally, conditioning on the first element of a collision sequence being selected gives $1+\frac{2\,\alpha}{1-\alpha}$.

This completes the proof. □

Proposition 2 says for any G_3 where $\mathcal{L}(\widehat{I}_j)\leq 1$, for all $j\in\mathbb{Z}_p^+$, then the expected length of collision sequences is $1+\frac{2\alpha}{1-\alpha}$. Thus randomly and uniformly selecting a collision sequence $\mathcal{C}$ from such a G_3, then this collision sequence will be of expected length $1+\frac{2\alpha}{1-\alpha}$. This means all collision sequences' lengths added together divided by the number of collision sequences is bounded above by $1+\frac{2\alpha}{1-\alpha}$.

Lemma 7. *Suppose $\mathcal{L}(\widehat{I}_j)\leq 1$, for all $j\in\mathbb{Z}_p^+$, and $\alpha = |\widehat{\mathcal{I}}|/|\mathcal{I}|$. Then*

$$\mathcal{G}(\widehat{\mathcal{I}}) \geq \frac{1-\alpha}{1+\alpha}|\widehat{\mathcal{I}}|.$$

Proof. Let c be the total number of collision sequences of length greater than 0. Also, the ith collision sequence is Coll_Sequence(i) for $i : c\geq i\geq 1$. A randomly and uniformly selected collision sequence from G_3 when $\mathcal{L}(\widehat{I}_j)\leq 1$ for $j\in\mathbb{Z}_p^+$ has expected length $1+\frac{2\,\alpha}{1-\alpha}$ by Proposition 2. This means the average of all the collision sequence lengths is $1+\frac{2\,\alpha}{1-\alpha}$. Thus, if $\mathcal{C}$ represents a randomly and uniformly chosen collision sequence, then

$$\mathbf{E}[\mathbf{Length}(\mathcal{C})] = \frac{\sum_{i=1}^{c}\mathbf{Length}(\text{Coll\_Sequence}(i))}{c}.$$

Furthermore, since $\mathbf{E}[\mathbf{Length}(\mathcal{C})]\leq 1+\frac{2\,\alpha}{1-\alpha}$ by Proposition 2 and since $\mathcal{L}(\widehat{I}_j)\leq 1$, for all $j\in\mathbb{Z}_p^+$, and since

$$|\widehat{\mathcal{I}}| = \sum_{i=1}^{c}\mathbf{Length}(\text{Coll\_Sequence}(i)),$$

it must be that

$$\frac{|\widehat{\mathcal{I}}|}{c} \le 1 + \frac{2\alpha}{1-\alpha}.$$

Finally, each of the collision sequences has global expansion of 1, and so $\mathcal{G}(\widehat{\mathcal{I}}) = c$. In other words,

$$\frac{1-\alpha}{1+\alpha}|\widehat{\mathcal{I}}| \le \mathcal{G}(\widehat{\mathcal{I}}).$$

This completes the proof. □

The function $\frac{1-\alpha}{1+\alpha}$ minimizes at $\alpha = \frac{1}{2}$, for $\alpha : \frac{1}{2} \ge \alpha > 0$, since

$$\frac{d}{d\alpha}\left(\frac{1-\alpha}{1+\alpha}\right) = \frac{-1}{1+\alpha} - \frac{1-\alpha}{(1+\alpha)^2},$$

which is negative for $\alpha : \frac{1}{2} \ge \alpha > 0$.

Definition 13. *Two collision sequences*

$$\begin{aligned}\mathcal{C}_1 &= (j_1,k) \to (j_2,k) \to \cdots \to (j_t,k),\\ \mathcal{C}_2 &= (j_{t+2},k) \to (j_{t+3},k) \to \cdots \to (j_u,k)\end{aligned}$$

can be joined *into one by selecting the element* (j_{t+1},k)*, where*

$$\begin{aligned} g(j_t,k) &\to (j_{t+1},k) \qquad \textit{and}\\ g(j_{t+1},k) &\to (j_{t+2},k).\end{aligned}$$

In this case, $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *are* separated *by one selection.*

Consider a total of $c < p/2$ collision sequences,

$$\mathcal{C}_1,\ \mathcal{C}_2, \ldots, \mathcal{C}_c,$$

where $\mathcal{C}_i$ and $\mathcal{C}_{i+1}$, for $i : c > i \ge 1$, are separated by one selection. Then selecting $c-1$ elements joins all of these collision sequences into one single collision sequence. This can be stated as the following lemma.

Lemma 8. *Given* $c < p/2$ *collision sequences* $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_c$*, where* $\mathcal{C}_i$ *and* $\mathcal{C}_{i+1}$*, for* $i : c > i \ge 1$*, are each separated by one selection, then* $c-1$ *singletons can be used to join these collision sequences into one single collision sequence.*

Theorem 9 (main expander theorem). *Consider any input set* $\widehat{\mathcal{I}} \subset \mathcal{I}$ *from* G_3*, where* $|\widehat{\mathcal{I}}| \le \frac{p^2}{2}$ *so* $\alpha \le \frac{1}{2}$*. The subgraph* G_3 *expands by at least* $\frac{3}{16}$*.*

Proof. Suppose the adversary chooses the elements $\widehat{\mathcal{I}}$ from $\mathcal{I}$, where $|\widehat{\mathcal{I}}| \le \frac{1}{2}|\mathcal{I}|$.

If $\mathcal{L}(\widehat{\mathcal{I}}) \ge \frac{3}{16}|\widehat{\mathcal{I}}|$, then we are done since there is a total of at least $\frac{3}{16}$ expansion. Therefore, consider the case where $\mathcal{L}(\widehat{\mathcal{I}}) < \frac{3}{16}|\widehat{\mathcal{I}}|$.

Start with the situation where $\mathcal{L}(\widehat{I}_j) \le 1$ for all $j \in \mathbb{Z}_p^+$; then by Lemma 7 there is global expansion of at least $\frac{1}{3}$ since $\alpha = \frac{1}{2}$ minimizes $\frac{1-\alpha}{1+\alpha}$ for $\alpha : \frac{1}{2} \ge \alpha > 0$. If $\alpha = \frac{1}{2}$, then $\frac{1-\alpha}{1+\alpha} = \frac{1}{3}$, and so

$$\mathcal{G}(\widehat{\mathcal{I}}) \ge \frac{1}{3}|\widehat{\mathcal{I}}|.$$

Now take the more general situation where $\widehat{\mathcal{I}}$ is such that $\mathcal{L}(\widehat{\mathcal{I}}) < \frac{3}{16}|\widehat{\mathcal{I}}|$. There are several cases to consider.

- *Case* 1. Not counting singletons.
  For now, assume more than $\frac{13}{16}|\widehat{\mathcal{I}}|$ selected elements are in ***loc***-contiguous subblocks. This case assumes no singletons. Thus, ignore the up to $\frac{3}{16}|\widehat{\mathcal{I}}|$ potential singletons.
  Therefore, applying Lemma 7 with $\alpha' \leq \frac{13}{16}\alpha \leq \frac{13}{32}$, since $\alpha \leq \frac{1}{2}$, gives

$$\begin{aligned}\mathcal{G}(\widehat{\mathcal{I}}) &\geq \frac{1-\frac{13}{32}}{1+\frac{13}{32}}|\widehat{\mathcal{I}}| \\ &= \frac{19}{45}|\widehat{\mathcal{I}}|.\end{aligned}$$

  Furthermore, the function $\frac{1-\alpha'}{1+\alpha'}$ is minimal for $\alpha' = \frac{13}{32}$, where $\alpha' : \frac{13}{32} \geq \alpha' > 0$. This case is complete since $\frac{19}{45} > \frac{3}{16}$.
- *Case* 2. Extending collision sequences.
  First, if the up to $\frac{3}{16}|\widehat{\mathcal{I}}|$ of the elements are used to extend but not join any two collision sequences, then the global expansion remains the same. Here the adversary has simply made the collision sequences longer, thereby changing in which block each of them terminates. But the same number of collision sequence ends remain, giving the same global expansion. This case is complete.
- *Case* 3. Joining collision sequences.
  Suppose each of the up to $\frac{3}{16}|\widehat{\mathcal{I}}|$ singleton elements can be used to join collision sequences together.
  Of course, it is given there are at least

$$\frac{19}{45}|\widehat{\mathcal{I}}| > \frac{6}{16}|\widehat{\mathcal{I}}|$$

  collision sequences by Case 1 (by applying Lemma 7). If each of the $\frac{3}{16}|\widehat{\mathcal{I}}|$ singleton elements joins exactly two collision sequences and all such pairs of joined collision sequences are disjoint, then the global expansion is reduced by at most $\frac{3}{16}$. This is because joining two collision sequences reduces the global expansion by one. Now if two singletons join three collision sequences, then the global expansion is also reduced by two. Lemma 8 indicates that each singleton reduces the global expansion by exactly one.
  Taking this argument to its logical end, let $\frac{3}{16}|\widehat{\mathcal{I}}|$ singleton elements be used to join at most $\frac{3}{16}|\widehat{\mathcal{I}}| + 1$ collision sequences. Applying Lemma 8, the $\frac{3}{16}|\widehat{\mathcal{I}}|$ singletons can be used to join as many as $\frac{3}{16}|\widehat{\mathcal{I}}| + 1$ collision sequences into one or a few collision sequences. Suppose these collision sequences do not have any expansion (i.e., they are of length p). Going further, say adding these singletons forms new ***loc***-contiguous subblocks and does not add any expansion themselves.
  This leaves more than

$$\frac{6}{16}|\widehat{\mathcal{I}}| - \frac{3}{16}|\widehat{\mathcal{I}}| = \frac{3}{16}|\widehat{\mathcal{I}}|$$

  collision sequences, giving global expansion of at least $\frac{3}{16}$, completing this case.

This completes the proof. □

Next, an accounting is made for an adversary selecting degenerate elements $(j, 0)$ for all $j \in \mathbb{Z}_p^+$ and $(p-1, k)$ for all $k \in \mathbb{Z}_p^+$. Suppose the elements $(j, 0)$ for all $j \in \mathbb{Z}_p^+$ are selected. Recall these degenerate elements give no global expansion since $\boldsymbol{g}(j, 0) = \boldsymbol{id}(j, 0)$. Further, assume they extend ***loc***-contiguous sequences in each $\widehat{I}_j$, thus giving no additional local expansion. Thus, in this case the selection of these elements takes the expansion from $\frac{3}{16}$ to $\frac{3}{16}(\frac{1}{1+\frac{1}{p}})$.

In addition, suppose the elements $(p-1, k)$ for all $k \in \mathbb{Z}_p^+$ are also selected. These elements give no local expansion since $\boldsymbol{loc}(p-1, k) = (p-1, k)$. However, it is possible that $p-1$ of these elements can join two collision sequences together. Note that node $(p-1, 0)$ is degenerate both locally and globally. Joining pairs of collision sequences together without adding local expansion changes the expansion of $\frac{3}{16}(\frac{1}{1+\frac{1}{p}})$ to

$$\frac{3}{16}\left(\frac{1}{1+\frac{1}{p}}\right)\left(1-\frac{1}{p-1}\right) = \frac{3}{16}\left(\frac{p}{p+1}\right)\left(\frac{p-2}{p-1}\right).$$

5. Conclusions. This paper gives a new way of showing expansion of three permutations comprising the Gabber–Galil expander. This is done without using eigenvalue methods or higher algebra. We use methods based on Chor and Goldreich's Theorem 3 on pairwise independence. It is important to notice that we are applying the probabilistic method to a fixed graph. For $\alpha = \frac{1}{2}$, Theorems 8 and 9 tie closely the expected collision sequence length of $2 = \frac{2\alpha}{1-\alpha} = \frac{1}{1-\alpha}$, giving insight into double hashing and graph expansion. This is interesting since our expander result says no matter what distribution of inputs are chosen while restricting local expansion, then we still have expected collision sequence length bounded by 2. If local expansion is not restricted sufficiently, then the graph expands (locally) as well. Likewise, for double hashing, no matter what input distribution is assumed for the keys, randomly, independently, and uniformly choosing two universal hash functions gives expected collision sequence length bounded by 2. This gives insight into both graph expanders as well as double hashing.

Fundamentally, universal hash functions are small sets that "randomize" well. Likewise, expander graphs have relatively few edges, yet they seem to have many properties amenable to randomness.

Acknowledgment. We are grateful to a referee for pointing out a problem in a previous version.

REFERENCES

[1] M. Ajtai, *Recursive construction for* 3*-regular expanders*, Combinatorica, 14 (1994), pp. 379–416.

[2] N. Alon, *Eigenvalues and expanders*, Combinatorica, 6 (1986), pp. 83–96.

[3] N. Alon, *Tools from higher algebra*, in Handbook of Combinatorics, Vol. 1, 2, R. L. Graham, M. Grötschel, and L. Lovász, eds., Elsevier, Amsterdam, 1995, pp. 1749–1783.

[4] N. Alon, Z. Galil, and V. D. Milman, *Better expanders and superconcentrators*, J. Algorithms, 8 (1987), pp. 337–347.

[5] N. Alon and J. Spencer, *The Probabilistic Method*, Wiley-Interscience, John Wiley and Sons, New York, 1992.

[6] M. Blum, R. M. Karp, O. Vornberger, C. H. Papadimitriou, and M. Yannakakis, *The complexity of testing whether a graph is a superconcentrator*, Inform. Process. Lett., 13 (1981) pp. 164–167.

[7] M. Capalbo, O. Reingold, S. Vadhan, and A. Wigderson, *Randomness conductors and constant-degree lossless expanders*, in Proceedings of the Thirty-Fourth Annual Symposium on Theory of Computing, ACM, New York, 2002, pp. 659–668.

[8] J. L. Carter and M. N. Wegman, *Universal classes of hash functions*, J. Comput. System Sci., 18 (1979), pp. 143–154.

[9] B. Chor and O. Goldreich, *On the power of two-points based sampling*, J. Complexity, 5 (1989), pp. 96–106.

[10] F. R. K. Chung, *Spectral Graph Theory*, CBMS Regional Conf. Ser. in Math. 92, AMS, Providence, RI, 1997.

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed., MIT Press, Cambridge, MA, 2001.

[12] O. Gabber and Z. Galil, *Explicit construction of linear-sized superconcentrators*, J. Comput. System Sci., 22 (1981), pp. 407–420.

[13] G. H. Gonnet and R. A. Baeza-Yates, *Handbook of Algorithms and Data Structures*, 2nd ed., Addison–Wesley, Reading, MA, 1990.

[14] L. Guibas and E. Szemerédi, *The analysis of double hashing*, J. Comput. System Sci., 16 (1978), pp. 226–274.

[15] S. Jimbo and A. Maruoka, *Expanders obtained from affine transformations*, Combinatorica, 7 (1987), pp. 343–355.

[16] N. Kahale, *Eigenvalues and expansion of regular graphs*, J. Assoc. Comput. Mach., 42 (1995), pp. 1091–1106.

[17] D. E. Knuth, *The Art of Computer Programming*, *Sorting and Searching*, Vol. 3, Addison–Wesley, Reading, MA, 1973.

[18] A. Lubotzky, *Discrete Groups, Expanding Graphs, and Invariant Measures*, Progr. Math. 125, Birkhäuser, Basel, 1994.

[19] A. Lubotzky, R. Phillips, and P. Sarnak, *Ramanujan graphs*, Combinatorica, 8 (1988), pp. 261–277.

[20] G. S. Lueker and M. Molodowitch, *More analysis of double hashing*, Combinatorica, 13 (1993), pp. 83–96.

[21] G. A. Margulis, *Explicit constructions of concentrators*, Problemy Peredachi Informacii, 9 (1973), pp. 71–80 (in Russian); Problems Inform. Transmission, 9 (1975), pp. 325–332 (in English).

[22] G. A. Margulis, *Explicit group-theoretical constructions of combinatorial schemes and their application to the design of expanders and superconcentrators*, Problemy Peredachi Informatsii, 24 (1988), pp. 51–60 (in Russian); Problems Inform. Transmission, 24 (1988), pp. 39–46 (in English).

[23] K. Mehlhorn, *On the program size of perfect and universal hash functions*, in Proceedings of the 23rd Annual Symposium on Foundations of Computer Science, IEEE, New York, 1982, pp. 170–175.

[24] K. Mehlhorn, *Data Structures and Efficient Algorithms*, Springer-Verlag, Berlin, 1984.

[25] R. Meshulam and A. Wigderson, *Expanders from symmetric codes*, in Proceedings of the Thirty-Fourth Annual Symposium on Theory of Computing, ACM, New York, 2002, pp. 669–677.

[26] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, Cambridge, UK, 1995.

[27] M. Pinsker, *On the complexity of a concentrator*, in Proceedings of the 7th International Teletraffic Conference, Stockholm, Sweden, 1973.

[28] O. Reingold, S. Vadhan, and A. Wigderson, *Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors*, Ann. of Math. (2), 155 (2002), pp. 157–187.

[29] P. Sarnak, *Some Applications of Modular Forms*, Cambridge Tracts in Math. 99, Cambridge University Press, Cambridge, UK, 1990.

[30] J. P. Schmidt and A. Siegel, *Double Hashing Is Computable and Randomizable with Universal Hash Functions*, NYU Technical report TR1995-686, New York University, New York, NY, 1995.

[31] A. Siegel, *On universal classes of fast hash functions, their time-space tradeoff, and their applications*, in Proceedings of 30th Annual Symposium on Foundations of Computer Science, IEEE, New York, 1989, pp. 20–25.

[32] A. Siegel, *On universal classes of extremely random constant-time hash functions*, SIAM J. Comput., 33 (2004), pp. 505–543.

[33] A. Wigderson and D. Zuckerman, *Expanders that beat the eigenvalue bound: Explicit construction and application*, Combinatorica, 19 (1999), pp. 125–138.

SIAM J. COMPUT.
Vol. 37, No. 1, pp. 112–131

HARDNESS OF THE UNDIRECTED CONGESTION MINIMIZATION PROBLEM*

MATTHEW ANDREWS[†] AND LISA ZHANG[†]

Abstract. We show that there is no $\gamma \log\log M/\log\log\log M$-approximation for the undirected congestion minimization problem unless $NP \subseteq ZPTIME(n^{\text{polylog } n})$, where M is the size of the graph and γ is some positive constant.

Key words. hardness of approximation, undirected graphs, congestion minimization

AMS subject classification. 68Q25

DOI. 10.1137/050636899

1. Introduction.

Consider a graph G with M edges and a set $\{(s_i, t_i)\}$ of source-sink pairs. The congestion minimization problem aims to connect all these pairs while minimizing the *edge congestion*, i.e., the maximum number of demands that go through the same edge in G. The problem is known to be NP-hard. The famous result of Raghavan and Thompson [12] states that by applying randomized rounding to a linear relaxation of the problem we obtain an $O(\log M/\log\log M)$-approximation for both directed and undirected graphs. Until a few years ago, the best hardness result was a factor $2-\varepsilon$ assuming $P \neq NP$ for both directed and undirected graphs. This follows from the fact that it is NP-hard to determine if all the terminals can be routed on edge-disjoint paths [11]. Closing the gap between the upper and lower bounds has been a long standing open problem.

For directed graphs, there have been a number of improvements to the lower bound. Chuzhoy and Naor [9] were the first to show a super constant hardness of $\Omega(\log\log M)$ assuming $NP \not\subseteq DTIME(n^{\log\log\log n})$.[1] Recently, we [6] built upon the techniques in this paper and proved an almost tight lower bound of $(\log M)^{1-\varepsilon}$ for any $\varepsilon > 0$ assuming $NP \not\subseteq ZPTIME(n^{\text{polylog}(n)})$.[2] Chuzhoy and Khanna [8] finally closed the small remaining gap and proved a hardness of $\log M/\log\log M$ for directed congestion minimization under the same complexity assumption as in [6].

In this paper we present the first super constant hardness result for undirected graphs. In particular we show that for some constant $\gamma > 0$, undirected congestion minimization is $\gamma \log\log M/\log\log\log M$-hard to approximate unless $NP \subseteq ZPTIME(n^{\text{polylog}(n)})$. Our proof also shows that the integrality gap for the natural linear programming relaxation of the problem is $\Omega(\log\log M/\log\log\log M)$.

The main technique for our hardness proof in this paper is the "canonical path" technique applied to high-girth graphs. We construct an instance of the congestion minimization problem on an undirected graph which has a special girth property. A canonical path is a direct path that connects the source-sink pair of a demand. There

*Received by the editors July 26, 2005; accepted for publication (in revised form) December 4, 2006; published electronically April 10, 2007. A preliminary version of this paper appeared in the Proceedings of the 37th ACM Symposium on Theory of Computing, 2005.

http://www.siam.org/journals/sicomp/37-1/63689.html

[†]Bell Laboratories, 600-700 Mountain Avenue, Murray Hill, NJ 07974 (andrews@research.bell-labs.com, ylz@research.bell-labs.com).

[1] $DTIME(n^{\log\log\log n})$ is the set of languages that have deterministic algorithms that have $n^{\log\log\log n}$ running time.

[2] $ZPTIME(n^{\text{polylog}(n)})$ is the set of languages that have randomized algorithms that always give the correct answer and have an expected running time $n^{\text{polylog}(n)}$.

are two high-level components in the proof. First, if all demands are routes along canonical paths, only then it is relatively straightforward to show some link has to have a high congestion. Unfortunately, demands can deviate from their canonical paths. Second, we show that, due to the girth property of the graph, with high probability most demands will have to follow their canonical paths. This technique of routing along canonical paths in high-girth graphs was first introduced in [1] to prove the logarithmic hardness result for the buy-at-bulk network design problem. Since then, we have further developed this technique to prove hardness for a number of additional problems, including the edge-disjoint paths problem [4, 2] and the wavelength assignment problem in optical networking [3, 5].

2. Construction. For the rest of this paper we refer to this congestion minimization problem on undirected graphs as MinCongestion. To show the hardness of MinCongestion we construct a reduction using the Raz verifier for Max3SAT(5). We begin with a brief overview of the Raz verifier.

2.1. Raz verifier. A 3SAT(5) formula has n variables and $5n/3$ clauses where each variable appears in exactly 5 distinct clauses and each clause contains exactly 3 literals. The Max3SAT(5) problem aims to find an assignment that maximizes the number of satisfied clauses. A 3SAT(5) formula is called a *yes-instance* if it is satisfiable; it is called a *no-instance* if no assignment satisfies more than a $1-\varepsilon$ fraction of the clauses for some constant $\varepsilon > 0$. It follows from the PCP theorem [7] that it is NP-hard to distinguish between yes-instances and no-instances.

A Raz verifier with ℓ repetitions is defined as follows [13, 9]. A verifier interacts with two provers: a clause prover (c-prover) and a variable prover (v-prover). Given a 3SAT(5) formula ϕ, the verifier sends the c-prover a clause query (c-query) that consists of ℓ clauses $c_1, \ldots, c_\ell$ chosen uniformly at random. It also sends the v-prover a variable query (v-query) that consists of one variable $v_1, \ldots, v_\ell$ chosen uniformly at random from each of the ℓ clauses. The c-prover sends back the assignment of every variable in clauses $c_1, \ldots, c_\ell$, and the v-prover sends back the assignment of the variables $v_1, \ldots, v_\ell$. The verifier *accepts* ϕ if all the ℓ clauses are satisfied and the two provers give a consistent assignment to the ℓ variables. The verifier *rejects* ϕ otherwise.

Suppose ϕ has n variables; then ϕ has $5n/3$ clauses. Clearly, a Raz verifier with ℓ repetitions has $Q_c := (5n/3)^\ell$ distinct c-queries, each of which has $A_c := 7^\ell$ answers. It also has $Q_v := n^\ell$ distinct v-queries, each of which has $A_v := 2^\ell$ answers. Since the verifier only queries variables that appear in a c-query, the number of distinct clause-variable query pairs is $R := (5n)^\ell$. Each clause appears in $R/Q_c = 3^\ell$ c-v query pairs, and each variable appears in $R/Q_v = 5^\ell$ c-v query pairs. These parameters are summarized in section 2.3.

Theorem 1 (Raz [13]). *There is a universal constant $\alpha > 1$ such that if ϕ is a yes-instance and there is a proof system in which the verifier always accepts and if ϕ is a no-instance, the verifier never accepts with probability higher than $\alpha^{-\ell}$.*

Given a 3SAT(5) formula ϕ we first construct the two-prover interactive proof system and represent it in a graph that we call the *proof system graph*, P. We then transform this proof system into a MinCongestion instance on a graph that we call the *transformed graph*, T. We show that if ϕ is a yes-instance, then all demands can be routed with congestion 1. If ϕ is a no-instance, then with high probability for any routing of the demands some edge has high congestion.

In defining these two graphs, we do not use the convention of specifying their node sets and edge sets. Instead we first specify a set of paths that exist in the graph.

We then add edges to the graph to ensure that these paths are realizable. Our graph construction involves many parameters which we define in section 2.3. We give an overview of the analysis in section 2.4. Throughout the paper we use subscripts c and v when discussing quantities pertaining to clauses and variables, and we omit the subscript when we do not distinguish them.

2.2. Graph construction. In the proof system graph P, for each possible answer a we have an answer edge which we also denote a. For each query q we group together all the possible answers to q and refer to this group as a query blob which we also denote q. A clause query blob (c-blob) contains A_c edges that we call *answer* edges, and a variable query blob (v-blob) contains A_v answer edges. For each c-v query pair r we have Y demands $d_{r,y}$, $1 \leq y \leq Y$, each of which has a source node $s_{r,y}$ and a destination node $t_{r,y}$ and routes one unit of flow. For each accepting interaction (r, a_c, a_v), demand $d_{r,y}$ has a special path p that we refer to as a *canonical path.*

This path starts at node $s_{r,y}$, passes through edges a_c, a_v, and ends at node $t_{r,y}$. In order for this to be possible, we place a *center* edge between a_c and a_v, a *demand* edge between $s_{r,y}$ and a_v, and a demand edge between a_c and $t_{r,y}$. Note that each demand has multiple canonical paths. Note also that we allow parallel demand edges and center edges so that no two canonical paths share a common edge other than answer edges.

To facilitate the analysis we label the canonical paths as follows. For each accepting interaction (r, a_c, a_v), there is a random Y-element permutation σ. The canonical path for demand $d_{r,y}$ that corresponds to the accepting interaction (r, a_c, a_v) is labeled $p_{r,a_c,a_v,\sigma(y)}$. That is, σ induces a *random* matching between the demands $d_{r,y}$ and the canonical paths corresponding to the accepting interaction (r, a_c, a_v).

Let us use an example to illustrate the above construction of P, shown in Figure 1. Let the repetition parameter $\ell = 1$. Let $\phi = (x \vee y \vee \bar{z}) \wedge (\bar{x} \vee y \vee w)$. For a query pair r that queries clause $x \vee y \vee \bar{z}$ and variable x, there are seven accepting interactions, namely, the seven satisfying assignments for the clause and the assignment of x consistent with the clause. Suppose $Y = 1$. We define one demand $d_{r,1}$ which has seven canonical paths defined by the seven accepting interactions. These seven canonical paths are shown by solid lines by Figure 1. The demand for query pair r' that queries clause $\bar{x} \vee y \vee w$ and variable x also has seven canonical paths, and they share the answer edges in the v-blob x with demands $d_{r,1}$. These canonical paths are shown by dotted lines in Figure 1. We did not draw four other demands and their canonical paths.

If $Y > 1$, then for each query pair r we define Y demands, each of which has its own source node and destination node and its own seven canonical paths. The canonical paths for these Y demands share the answer edges in c-blob $x \vee y \vee \bar{z}$ and v-blob x but have distinct center edges and demand edges.

We construct a MinCongestion instance on the transformed graph T. Like P we have Y demands $d_{r,y}$ for each c-v query pair r. Each demand has a source node $s_{r,y}$ and a destination node $t_{r,y}$. The graph T consists of Z levels. Each level of T consists of Q_c c-blobs and Q_v v-blobs, and they correspond one-to-one to those in P. Each c-blob in T consists of I_c *image edges*, and each v-blob consists of I_v image edges. We use $b_{q,z}$ to denote the blob at level z of T that corresponds to query blob q of P; we use $f_{q,z,j}$ to denote the jth image edge in $b_{q,z}$.

In the following we describe how to map canonical paths from P to T. For any c-blob q_c in P, $D_c := YR/Q_c$ demands have canonical paths that go through q_c.

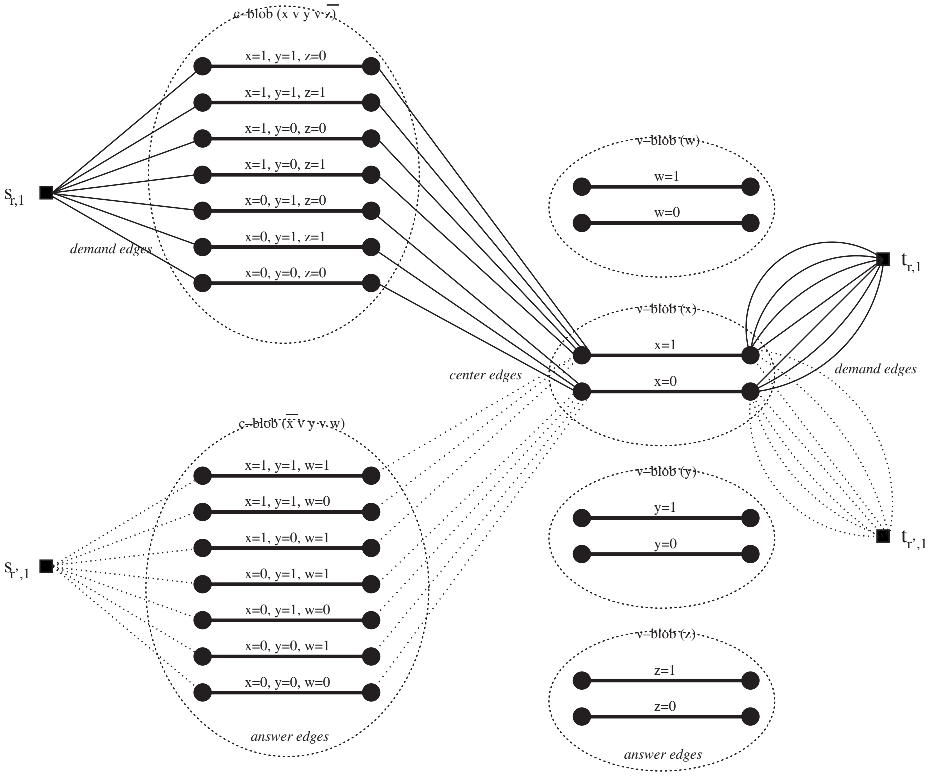


FIG. 1. *Graph P for a proof system: $\phi = (x \vee y \vee \bar{z}) \wedge (\bar{x} \vee y \vee w)$, $\ell = 1$, and $Y = 1$. The figure shows two out of six demands.*

Since two canonical paths of the same demand cannot share a common answer edge in q_c, $I_c := D_c$ canonical paths go through any answer edge a_c in q_c. Consider all the canonical paths that go through a_c in P. We map each of these paths to a random image edge in $b_{q_c,z}$ *subject to the constraint that no two of these paths are mapped to the same image edge.* That is, there is a random matching between the canonical paths going through a_c and the image edges they are mapped to in $b_{q_c,z}$. This is always possible due to the choice of I_c.

We now perform a similar construction for each v-query q_v. Note that for a fixed clause and a fixed variable, the number of satisfying assignments of the clause for which the variable is set to 1 is either 3 or 4, and the number of satisfying assignments for which the variable is set to 0 is also either 3 or 4. This implies that for any v-blob in P, at most 4^ℓ canonical paths of the same demand can share a common answer edge. Therefore, at most $I_v := 4^\ell \cdot YR/Q_v$ canonical paths can go through any answer edge a_v in q_v. These canonical paths are randomly mapped to *distinct* image edges in $b_{q_v,z}$ at every level z. (See Figure 2.) We emphasize that the random mapping for blob $b_{q,z}$ is independent of the random mapping for blob $b_{q,z'}$ for any $1 \leq z < z' \leq Z$.

Consider a canonical path p for demand $d_{r,y}$ that goes through query blobs q_c and q_v in P. For notational convenience, if p passes through the image edge $f_{q_c,z,j}$ (resp., $f_{q_v,z,j}$), we sometimes write $f_{c,p,z} = f_{q_c,z,j}$ (resp., $f_{v,p,z} = f_{q_v,z,j}$). If p is for demand $d_{r,y}$ we sometimes write $d_p = d_{r,y}$, $s_p = s_{r,y}$, and $t_p = t_{r,y}$. The above discussion describes which image edges path p passes through in T. Under the new notation path p has the following form which gives the ordering of the image edges

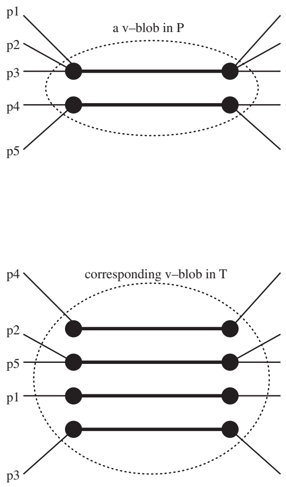


Fig. 2. *A possible mapping of five canonical paths from a v-blob in P to a v-blob in T. Paths p_1, p_2, and p_3, are mapped to distinct image edges, and paths p_4 and p_5 are mapped to distinct image edges. (The figure shows fewer than the actual number of canonical paths in P and fewer than the actual number of edges in T.)*

on the path:

$$s_p \to \cdots \to f_{c,p,1} \to \cdots \to f_{v,p,1} \to \cdots \to f_{c,p,z} \to \cdots \to f_{v,p,z} \\ \to \cdots \to f_{c,p,Z} \to \cdots \to f_{v,p,Z} \to \cdots \to t_p.$$

We add edges to T to make sure that these paths are realizable. In particular we add edges between s_p and $f_{c,p,1}$, between $f_{c,p,z}$ and $f_{v,p,z}$ for all z, between $f_{v,p,z}$ and $f_{c,p,z+1}$ for all z, and between $f_{v,p,Z}$ and t_p. We have now shown how to map a canonical path in P onto a canonical path in T. As before, we may add parallel edges so that no two canonical paths share a common edge other than inside a blob. This completes the description of the graph T and a MinCongestion instance on T. (See Figure 3.)

2.3. Parameters. Given a 3SAT(5) formula ϕ with n variables, we choose the Raz verifier repetition parameter ℓ to be

(1) $\ell = \beta \log\log\log n$ for a sufficiently large constant β.

We use the following parameters in constructing graphs P and T. Throughout the paper, we omit the ceilings and floors for ease of presentation.

(2)	$Q_c = (5n/3)^\ell$	no. of c-blobs in P and per level in T; no. of c-queries,
(3)	$Q_v = n^\ell$	no. of v-blobs in P and per level in T; no. of v-queries,
(4)	$A_c = 7^\ell$	no. of answer edges per c-blob in P; no. of answers to a c-query,
(5)	$A_v = 2^\ell$	no. of answer edges per v-blob in P; no. of answers to a v-query,
(6)	$I_c = Y \cdot 3^\ell$	no. of image edges per c-blob in T,
(7)	$I_v = Y \cdot 20^\ell$	no. of image edges per v-blob in T,
(8)	$R = (5n)^\ell$	no. of c-v query pairs,
	YR	no. of demands
(9)	$D_c = Y \cdot 3^\ell$	no. of demands per c-blob in P and T,
(10)	$D_v = Y \cdot 5^\ell$	no. of demands per v-blob in P and T.

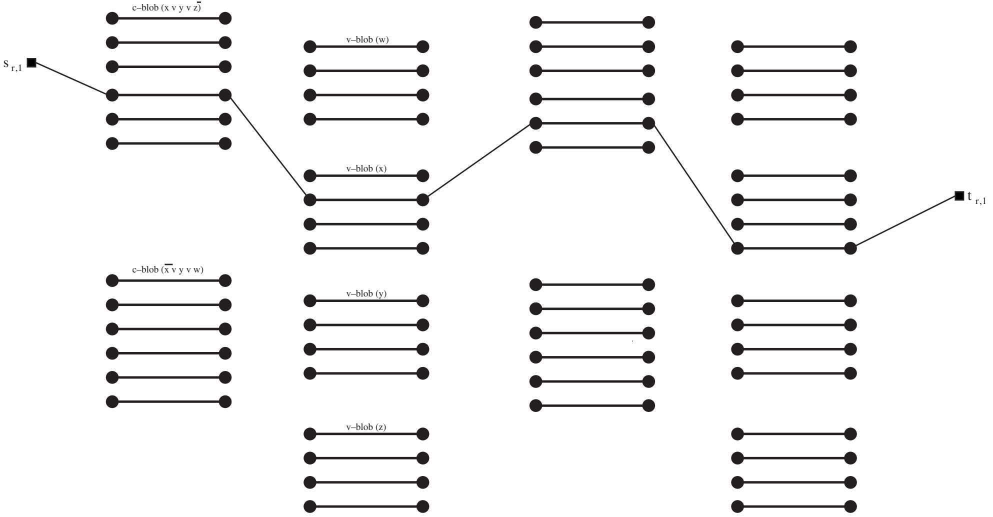


FIG. 3. *The transformed graph T from graph P shown in Figure 1, $Z = 2$. The figure shows one out of seven canonical paths for demand $d_{r,1}$. Note also that the figure shows fewer than the actual number of edges per v-blob.*

In order to define Y, the number of demands per c-v query, and Z, the number of levels in T, we introduce two new parameters, k and h. The parameter k is related to the length of noncanonical paths in T, and h is used to define the concept of *heavy* blobs in P.

(11) $h = \ell^{-1} \cdot \log\log n$ definition of heavy blob in P,

(12) $Z = (40 \cdot 8^{\ell})^{h+1}$ no. of levels in T,

(13) $k = Z\sqrt{Z}$ noncanonical path length in T,

(14) $Y = (6RQ_cA_cQ_vA_vZ)^{2k+1}$ no. of demands per c-v query pair.

Let M be the number of edges in T. We have YRZ image edges from c-blobs and $YRZ \cdot 4^{\ell}$ image edges from v-blobs. Canonical paths do not share common edges outside blobs in T, and there are $7^{\ell}(2Z+1)YR$ such edges. Therefore,

$$M = YRZ + YRZ \cdot 4^{\ell} + 7^{\ell}(2Z+1)YR. \tag{15}$$

To get a sense of how big Z, Y, and M are, we note that $\ell h = \log\log n$ and therefore $Z = \text{polylog}(n)$, $Y = e^{poly\ Z}$ which is $e^{\text{polylog}(n)}$. Finally, since every term in M is $e^{\text{polylog}(n)}$, M is $e^{\text{polylog}(n)}$ as well. We discuss how the values of these parameters are chosen in section 3.3, after we present the proof.

2.4. Overview of proof. At a high level our proof proceeds as follows. We use a solution to the MINCONGESTION instance on T to determine whether the original 3SAT(5) instance ϕ (that defined the proof system) is a yes-instance or a no-instance. The most important feature of our construction is that for all the canonical paths that pass through the same answer edge in query blob q in P, the corresponding canonical paths in T all pass through different image edges in $b_{q,z}$ in T. Therefore, if all demands are routed on canonical paths, then for any solution with high congestion in P, the corresponding solution in T typically has low congestion. Conversely, for any solution with low congestion in P, the corresponding solution in T typically has high congestion.

If ϕ is a yes-instance it is straightforward to argue that demands can be routed in T so that the congestion is 1. The two provers can always answer any query pair using one satisfiable assignment of ϕ, and the verifier always accepts. Therefore, these accepting interactions define one canonical path per demand, and only one answer edge in each query blob of P is used. By construction, when these canonical paths are mapped from P to T, these paths are mapped to distinct image edges in every blob in T. Therefore, the corresponding solution in T has congestion 1.

We concentrate on the case in which ϕ is a no-instance. If all demands are routed on canonical paths only in T (resp., in P), we say this solution is canonical in T (resp., in P). For a canonical solution to T, we show in section 3.1 that the corresponding canonical solution to P must have demands routed through a large number of answer edges in *some* blob. Otherwise, a small number of answer edges would be used in *every* blob, and we could use them to define a pair of provers that would violate the error probability of the proof system. (In the extreme example of a satisfiable ϕ, only one answer edge is used in every query blob in P, and they define the two provers.) We then show that for the blob in P in which many answer edges are used, one of the Z corresponding blobs in T necessarily has high congestion of $\Omega(h)$ with high probability.

In section 3.2 we consider the case in which noncanonical paths may be used. We say that a canonical path p for demand d_p is *regular* at level z in T if d_p is routed through both $f_{c,p,z}$ and $f_{v,p,z}$. We say a demand is *regular* at level z if it has a canonical path regular at level z; a demand is regular at ζ levels if it has a canonical path that is regular at ζ levels. We show that for most demands they are either routed on a long path in T or else they are regular at many levels. The proof of this fact uses girth properties of T and is similar to an analysis of the buy-at-bulk network design problem presented in [1].

If many demands are routed on long paths it is easy to show that the congestion must be high on some edge. If many demands are regular at many levels we generalize the argument for canonical solutions in section 3.1 (i.e., demands regular at all levels) to show that the solution in T must have high congestion of $\Omega(h)$. Therefore, we have shown a gap of $\Omega(h)$ in congestion depending on whether or not ϕ is a yes-instance or a no-instance.

3. No-instances.

For the case that ϕ is a no-instance we now carry out the detailed analysis outlined previously.

3.1. Special case: Canonical paths only.

Let us begin with some notation. Let $\mathcal{M}$ be the particular random mapping between the canonical paths in P and in T. If $\mathcal{T}$ is a canonical solution to the MinCongestion instance on T, then $\mathcal{T}$ and $\mathcal{M}$ define a canonical solution to P which we denote $\mathcal{P}(\mathcal{T}, \mathcal{M})$. Conversely, if $\mathcal{P}$ is a canonical solution to P, $\mathcal{P}$ and $\mathcal{M}$ define a canonical solution to T which we denote $\mathcal{T}(\mathcal{P}, \mathcal{M})$.

Under a canonical solution $\mathcal{P}$, we say that an answer edge in a blob in P is *heavy* if at least $D/(10A)$ demands go through the edge. (As before whenever there is no need to specify whether a blob is a c-blob or a v-blob, we omit the subscripts c and v. For example, A denotes the number of edges in the blob of interest, and D denotes the number of demands that go through the blob.) In addition, we say a query blob is heavy if it has at least h heavy answer edges; we say a solution $\mathcal{P}$ is heavy if it has at least one heavy query blob. We refer to anything as *light* if it is not heavy. For a canonical solution $\mathcal{T}$ to T, we first show that $\mathcal{P}(\mathcal{T}, \mathcal{M})$ has to be a heavy solution under any mapping $\mathcal{M}$. Otherwise, we choose *one* edge from the heavy answer edges in each blob, and these chosen answer edges define two provers. We then show that since every blob is light, many demands are routed along the chosen answer edges only. Furthermore, since canonical paths correspond to accepting interactions of the proof system, it is a contradiction for a no-instance ϕ if there are more such paths than the error probability of the proof system would allow. We then show that with high probability, every heavy canonical solution corresponds to a canonical solution in T that has high congestion. This probability is taken over all the random mappings $\mathcal{M}$.

LEMMA 2. *If $\mathcal{T}$ is a canonical solution, P has a heavy blob under $\mathcal{P}(\mathcal{T}, \mathcal{M})$.*

Proof. For the purpose of contradiction let us assume $\mathcal{P}(\mathcal{T}, \mathcal{M})$ is a light solution. Then every blob in P has fewer than h heavy answer edges. For a particular query blob q, fewer than $A \cdot D/(10A)$ demands can go through the light edges in q. Summing over all the v-blobs and c-blobs we have at most $YR/5$ demands whose routes contain light answer edges. Hence at least $4/5$ of all demands route through heavy answer edges only under solution $\mathcal{P}(\mathcal{T}, \mathcal{M})$.

We now choose one heavy edge uniformly at random in each query blob in P. At least $(4YR/5) \cdot h^{-2}$ demands are expected to go through these randomly chosen heavy edges only. Hence there exists a choice of heavy edges such that at least $(4YR/5) \cdot h^{-2}$ demands go through these heavy edges. We refer to these demands as *accepting demands* and the rest as *unaccepting.* Recall that we have defined Y demands per c-v query pair. Under solution $\mathcal{P}(\mathcal{T}, \mathcal{M})$, some of the Y demands that correspond to the same random string may be accepting, whereas the rest may be unaccepting. If this is the case for any random string, we reroute the unaccepting demands along the path of the accepting ones. Therefore, under this new solution all Y demands of a random string are either accepting or unaccepting. Moreover, a fraction of at least $4/(5h^2)$ of all demands are accepting.

Recall that canonical paths are defined by accepting interactions. Since accepting demands are routed along canonical paths, we have shown that at least a $4/(5h^2)$ fraction of random strings generates queries that have accepting answers to the verifier. By the choice of ℓ in (1) and h in (11), $4/(5h^2)$ is higher than the error probability $\alpha^{-\ell}$ defined in Theorem 1. This is a contradiction for a no-instance ϕ. Note that the choice of ℓ is the smallest possible to ensure the contradiction here. ◻

We study the congestion in T that occurs when $\mathcal{P}(\mathcal{T}, \mathcal{M})$ has a heavy blob. The mapping of canonical paths between P and T corresponds to the following balls-and-bins game. We are given I bins, and we throw n_i balls into n_i distinct bins during the ith round. For $D = n_1 + \cdots + n_A$ and $I \geq D$, what is the maximum number of balls in a bin at the end of A rounds?

To see the connection to our problem, let us consider how demands are routed in one query blob q in P under a solution $\mathcal{P}(\mathcal{T}, \mathcal{M})$. The n_i's represent the number of demands that are routed along the ith answer edge in q; D represents the total number of demands routed through q; A represents the number of edges in q; and I represents the number of edges in a blob in T that correspond to q. By construction these n_i paths are mapped to distinct edges in every corresponding blob in T. This is the same as throwing n_i balls into I bins with each ball landing in a distinct bin. Therefore, the maximum number of balls per bin is the maximum edge congestion in one corresponding blob in T.

Without loss of generality, let us assume $n_1 \geq n_2 \cdots \geq n_A$. Let x be such that $n_x \geq D/(10A) > n_{x+1}$. If $n_A > D/(10A)$, then $x = A$. Such an x always exists.

Lemma 3. *The probability that every bin has fewer than x balls is at most $e^{-(20Aa)^{-x-1}D}$, where $a = I/D$.*

Proof. We compute the probability that some bin has x balls after the first x rounds and ignore the balls thrown after x rounds. It is easy to see that for a particular bin to have a ball in round i equals n_i/I since n_i distinct bins out of I are chosen at random. Therefore, this probability is at least $1/(10Aa)$ for bins $i \leq x$ since $n_i \geq D/(10A)$ and $a = I/D$. Since each round is independent, the probability that one bin has x balls or more is at least $(10Aa)^{-x}$. Equivalently, the probability that one bin has fewer than x balls is at most $1 - (10Aa)^{-x}$.

In order to bound the probability that every bin has fewer than x balls, we deal with the dependence among the bins in the following way. Consider bin j, where $j \leq D/(20A)$.

$$\begin{aligned}
&\Pr\left[\text{Bin } j \text{ has at least } x \text{ balls} \mid \text{Any configuration of bin } 1, \dots, j-1\right]\\
&\geq \Pr\left[\text{Bin } j \text{ has at least } x \text{ balls} \mid \text{Bin } 1, \dots, j-1 \text{ each has } x \text{ balls}\right]\\
&= \prod_{1\leq i\leq x} \frac{n_i - j}{I - j}\\
&\geq \prod_{1\leq i\leq x} \frac{n_i - D/(20A)}{I}\\
&\geq (20Aa)^{-x}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
&\Pr\left[\text{Bin } j \text{ has fewer than } x \text{ balls} \mid \text{Any configuration of bin } 1, \dots, j-1\right]\\
&\qquad \leq 1 - (20Aa)^{-x}.
\end{aligned}$$

Using conditional probability, we have

$$\begin{aligned}
&\Pr\left[\text{Every bin has fewer than } x \text{ balls}\right]\\
&\qquad \leq \Pr\left[\text{Bins } 1, \dots, D/(20A) \text{ each has fewer than } x \text{ balls}\right]\\
&\qquad \leq (1 - (20Aa)^{-x})^{D/(20A)} \leq e^{-(20Aa)^{-x-1}D}. \qquad \square
\end{aligned}$$

We now define a set of bad events. Let q be a query blob in P, let $H = \{a_1, \dots, a_h\}$ be a set of h answer edges, and let E_{a_i} for $1 \leq i \leq h$ be a set of $D/10A$ canonical paths in P that pass through the answer edge a_i. Let $B(q, E_{a_1}, \dots, E_{a_h})$ be the bad event in which under $\mathcal{M}$, the images of the paths in $E_{a_1}, \dots, E_{a_h}$ in T create congestion less than h.

Recall in the construction of P, at most 4^ℓ canonical paths of the same demand can share the same answer edge in a v-blob in P, and no two canonical paths of the same demand can share the same answer edge in a c-blob in P. Therefore, the value of a in Lemma 3 is 4^ℓ for a v-blob and $a = 1$ for a c-blob. Recall also that for every blob in P we create Z consecutive blobs in T, and the mappings of the canonical paths from P to these Z blobs are independent from one another. Finally, imagine that for each E_{a_i} a demand is routed along each path in E_{a_i}. In this case the value of x (defined above) satisfies $x \geq h$. Therefore, Lemma 3 and the choice of Z immediately imply the following.

Corollary 4. *For* fixed $q, E_{a_1}, \dots, E_{a_h}$, *the probability that $B(q, E_{a_1}, \dots, E_{a_h})$ occurs is at most*

$$\begin{aligned}
e^{-(20A_c)^{-h-1}D_cZ} &\leq e^{-D_c} \quad \textit{if } q \textit{ is a c-blob},\\
e^{-(20A_v4^\ell)^{-h-1}D_vZ} &\leq e^{-D_v} \quad \textit{if } q \textit{ is a v-blob}.
\end{aligned}$$

This probability is with respect to the random mapping $\mathcal{M}$.

We now count the number of bad events. If q is a v-blob, then at most $D_v 4^\ell$ canonical paths pass through any answer edge in q. Therefore, an upper bound on the total number of events of the form $B(q, E_{a_1}, \ldots, E_{a_h})$ for a v-blob q is

$$\begin{aligned} Q_v \binom{A_v}{h} \binom{D_v 4^\ell}{D_v/(10A_v)}^h &\le Q_v \cdot (A_v)^h \cdot (10 A_v 4^\ell e)^{hD_v/(10A_v)} \\ &= e^{\log Q_v} \cdot e^{h \log A_v} \cdot e^{(1+\log 10 + \log A_v + \log 4^\ell) h D_v/(10A_v)} \\ &= e^{o(D_v)}. \end{aligned}$$

The last equality holds since the exponent in each of the three terms is $o(D_v)$. If q is a c-blob, at most D_c canonical paths pass through any answer edge in q. A similar (but simpler) calculation shows that an upper bound on the total number of events of the form $B(q, E_{a_1}, \ldots, E_{a_h})$ for a c-blob q is

$$Q_c \binom{A_c}{h} \binom{D_c}{D_c/(10A_c)}^h = e^{o(D_c)}.$$

By a union bound, the probability that *some* event $B(q, E_{a_1}, \ldots, E_{a_h})$ happens is at most

$$\begin{aligned} &e^{-D_c} \cdot Q_c \binom{A_c}{h} \binom{D_c}{D_c/(10A_c)}^h + e^{-D_v} \cdot Q_v \binom{A_v}{h} \binom{D_v 4^\ell}{D_v/(10A_v)}^h \\ &\quad \le e^{-D_v} \cdot e^{o(D_v)} + e^{-D_c} \cdot e^{o(D_c)}, \end{aligned}$$

which is at most $1/\text{poly}(n)$.

Now suppose that under mapping $\mathcal{M}$ no such bad event occurs. For any heavy solution $\mathcal{P}$ in P, by definition we can find a query blob q with h answer edges such that for each such edge $D/10A$ demands are routed on a canonical path that passes through the edge. Since no bad event occurs, the images of these canonical paths in T create congestion at least h. In other words, the canonical solution $\mathcal{T}(\mathcal{P}, \mathcal{M})$ in T has a congestion of at least h. We have therefore shown the following.

LEMMA 5. *With probability* $1 - 1/\text{poly}(n)$ *every heavy solution* $\mathcal{P}$ *corresponds to a solution* $\mathcal{T}(\mathcal{P}, \mathcal{M})$ *in which the congestion is at least* h *in* T.

Lemma 2 states that if $\mathcal{T}$ is a canonical solution, then $\mathcal{P}(\mathcal{T}, \mathcal{M})$ must be a heavy solution in P. Therefore, by Lemma 5, with high probability over the choice of $\mathcal{M}$, $\mathcal{T}$ must have congestion at least h since $\mathcal{T} = \mathcal{T}(\mathcal{P}(\mathcal{T}, \mathcal{M}))$. In summary,

THEOREM 6. *With probability* $1 - 1/\text{poly}(n)$*, every canonical solution* $\mathcal{T}$ *has a congestion of at least* h.

3.2. General case. We now consider the general problem in which a demand does not follow a canonical path p but takes a detour. Whenever this happens the canonical path p and the detour form a cycle in T. We would like to bound the number and the lengths of these cycles in T and thereby quantify the detours. However, a direct analysis on T seems hard mainly due to the cycles formed by two complete canonical paths of the same demand. To facilitate the analysis, we create an *incidence graph* G for which we are able to show the number of small cycles is likely to be small. This allows us to show that for most demands, either they are routed along long paths in T or else they are regular at many levels. We show the congestion in T is high in both cases.

Recall a canonical path p is regular at level z if the demand is routed through both $f_{c,p,z}$ and $f_{v,p,z}$, i.e., it does not take a detour from the two image edges along p at level z; a demand is regular at ζ levels if it has a canonical path regular at ζ levels.

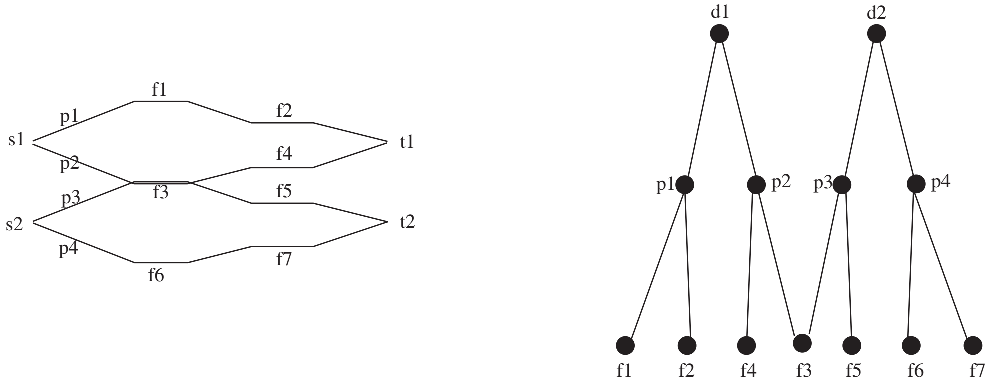


FIG. 4. *(Left) A simple illustrative example of T with two demands d_1 and d_2 and four paths p_1, p_2, p_3, and p_4. (Right) The corresponding incidence graph G.*

3.2.1. Incidence graph. We begin with a description of the incidence graph. The incidence graph G has a node for each demand d, each canonical path p, and each image edge $f_{q,z,j}$ (see Figure 4). If path p is a canonical path for demand d, (i.e., $d_p = d$), then there is an edge between d and p in G. If path p passes through $f_{q,z,j}$, (i.e., $f_{c,p,z} = f_{q,z,j}$ or $f_{v,p,z} = f_{q,z,j}$), then there is an edge between p and $f_{q,z,j}$. (Recall we have two ways of denoting an image edge in T. The notation $f_{c,p,z}$ refers to the image edge in the c-blob at level z along path p; $f_{q,z,j}$ refers to the jth image edge in the zth blob that corresponds to blob q of P.)

We now show how each edge in T maps into a path in G. Each image edge in T maps to a single node in G which we call an *image node*. Each edge that connects two image edges (or an image edge and a source/destination node) along a canonical path p in T, maps to the two-edge path that connects the two image nodes (or the image node and the demand node d_p) via the path node p. We refer to these paths in G as *route components*. More formally, the route components are defined as follows.

- The image edge $f_{i,p,y}$ in T maps into the single image node $f_{i,p,y}$ in G.
- The edge between $f_{c,p,z}$ and $f_{v,p,z}$ maps into the path $f_{c,p,z} \to p \to f_{v,p,z}$.[3]
- The edge between $f_{v,p,z}$ and $f_{c,p,z+1}$ maps into the path $f_{v,p,z} \to p \to f_{c,p,z+1}$.
- The edge between s_p and $f_{c,p,1}$ maps into the path $d \to p \to f_{c,p,1}$.
- The edge between $f_{v,p,Z}$ and t_p maps into the path $f_{v,p,Z} \to p \to d$.

The above definition of route components implies the following relationship between the path lengths in T and in G.

LEMMA 7. *For any route in T of length x, the corresponding route in G has length at most $2x$.*

We can assume without loss of generality that the route in T is *simple* (i.e., it never visits the same node twice). This implies that, although the route in G is not necessarily simple, it can traverse each of the above route components at most once.

Before presenting the details, we first discuss how T and G are related at a high level. Consider a demand d, and let π be the path that d is routed along in T. The first edge in π, i.e., the one that leaves the source node s_d, belongs to a canonical path, say p. Suppose that p and π are not identical. This implies that the union of p

[3] If the route in T goes from $f_{c,p,z}$ to $f_{v,p,z}$, then the path in G goes from node $f_{c,p,z}$ to node p to node $f_{v,p,z}$. If the route in T goes from $f_{v,p,z}$ to $f_{c,p,z}$, then the path in G goes in the opposite direction. A similar statement regarding the orientation of the path in G holds for each of the other mappings.

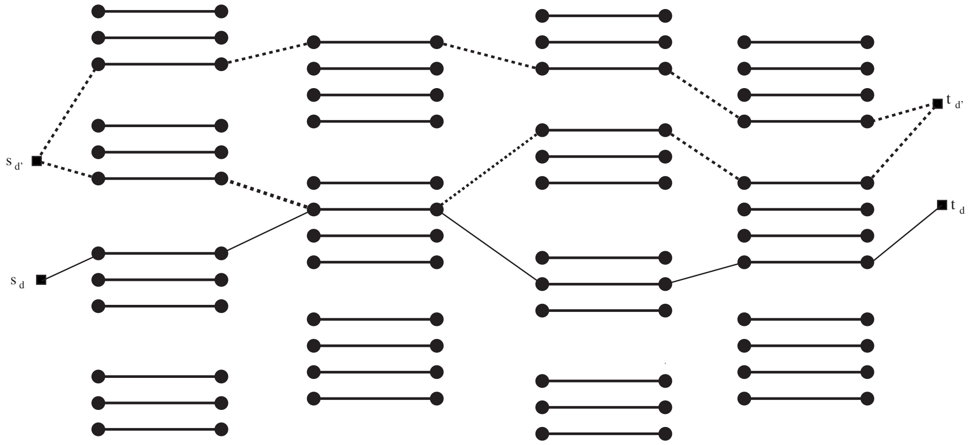


Fig. 5. *An image edge $f_{v,p,1}$ along canonical path p for demand d is detoured along a cycle formed by two complete canonical paths of demand d'. The cycle is shown in dotted lines. Image edges on the dotted cycle can be detoured along other such cycles.*

and π form a cycle in T. There are two scenarios to consider, depending on whether or not $\pi(G)$ contains a cycle in G, where $\pi(G)$ is the mapping of path π in graph G.

Case 1 ($\pi(G)$ *contains a cycle in* G). In this case we can show that either $\pi(G)$ contains a small cycle and the demand node d is close to this small cycle; $\pi(G)$ contains a small cycle but d is far away from any such small cycle; or $\pi(G)$ contains large cycles only. If the first situation holds, then we say the demand is of type (1); if the latter two situations hold the demand is of type (2). We show in Lemma 11 that G is likely to have only a small number of small cycles due to the randomness in our construction. Consequently, the number of type-(1) demands is probably small. We also have that long paths in G correspond to long paths in T; hence, if d is a type-(2) demand, then π is a long path in T. Using this fact we show in Lemma 12 that if there are many type-(2) demands, the congestion in T has to be high.

Case 2 ($\pi(G)$ *does not contain a cycle in* G). We show in Lemma 8 that the only way this scenario can happen is if an image edge in p is detoured along a cycle formed by two complete canonical paths of a different demand d'. (See Figure 5.) (We remark that we allow "nested" detours in that some of the image edges on the detour may themselves be detoured.) In other words, if a nonimage edge in p is detoured or if an image edge in p is detoured in a different manner, then $\pi(G)$ has a cycle in G.

If π detours from p at more than $\sqrt{Z}$ levels, we say d is of type (3); otherwise, d is of type (4). Since each detour of an image edge in this scenario has a length longer than Z, type-(3) demands are routed along long paths of length longer than $Z\sqrt{Z}$. If there are many type-(3) demands the congestion in T has to be high, as shown in Lemma 12. A type-(4) demand is routed along a path π that resembles the canonical path p in most places. In particular, the path π passes through the same image edges as p at more than $Z - \sqrt{Z}$ levels; i.e., the path p is regular at more than $Z - \sqrt{Z}$ levels. In section 3.2.5 we focus on the $Z - \sqrt{Z}$ regular levels and carry out an analysis similar to the special case of canonical paths only.

3.2.2. Demands of four types. We now formally show that demands have four types. The proof is similar to an analysis presented in [1].

Lemma 8. *Every demand $d_{r,y}$ has at least one of the four following types, where k is the parameter defined in* (13)*:*

1. *$\pi(G)$ contains a cycle of length at most k in G, and the demand node $d_{r,y}$ is at most distance k from this cycle.*
2. *$\pi(G)$ contains a cycle in G, but demand $d_{r,y}$ is not of type-(1). This implies that the length of $\pi(G)$ is more than k in G.*
3. *Demand $d_{r,y}$ is regular at at most $Z-\sqrt{Z}$ levels in T. This implies the length of π is more than $Z\sqrt{Z}$ in T.*
4. *Demand $d_{r,y}$ is regular at more than $Z-\sqrt{Z}$ levels in T.*

Proof. If the demand $d_{r,y}$ is type-(2), then either $\pi(G)$ contains a cycle longer than k, or $\pi(G)$ contains cycles of lengths at most k but $d_{r,y}$ is more than distance k from any of these cycles. In both cases the length of $\pi(G)$ is more than k. If the demand $d_{r,y}$ is not of type (1) or (2), then $\pi(G)$ does not have a cycle in G. Under this situation, we use the following fact repeatedly to show that if an edge along π deviates from p, then it must be an image edge and it detours along a cycle formed by two complete canonical paths of a different demand. Such detours can be nested. (See Figure 5.) This implies the demand must be type of (3) or (4).

FACT 9. *Consider a route in a graph such that the subgraph induced by the route does not contain a cycle. Then, if the route leaves node v along edge e, it cannot return to node v except along edge e. More generally, if the route traverses edge e* at any time *after visiting node v, it cannot return to node v without traversing edge e in the opposite direction.*

Consider now the route for demand $d_{r,y}$ in T. The route in T must begin at the source node for $d_{r,y}$ and then cross one of the edges connecting the source node to an answer edge at level 1. Recall that the route in G must follow the route components and it can traverse each route component at most once. Hence the route in G must begin $d_{r,y} \rightarrow p \rightarrow f_{c,p,1}$ for some path p that is owned by $d_{r,y}$. The route in G must return to node $d_{r,y}$ since the route in T must eventually reach the destination node for $d_{r,y}$. By Fact 9 this return to $d_{r,y}$ must happen via node p. Fact 9 also implies that on this return to node p the previous node must be $f_{c,p,1}$. Since each route component can be used at most once, the route in G must have the form

$$d_{r,y} \rightarrow p \rightarrow f_{c,p,1} \rightarrow \Gamma_{c,1} \rightarrow f_{c,p,1} \rightarrow p \rightarrow f_{v,p,1} \cdots$$

for some subroute $\Gamma_{c,1}$ that does not pass through p. Once again, Fact 9 implies that the route must return to node p in order to get back to node $d_{r,y}$. It must do this through node $f_{v,p,1}$. Therefore, because we can use each of our route components at most once, the route in G must have the form

$$d_{r,y} \rightarrow p \rightarrow f_{c,p,1} \rightarrow \Gamma_{c,1} \rightarrow f_{c,p,1} \rightarrow p \rightarrow f_{v,p,1} \rightarrow \Gamma_{v,1} \rightarrow f_{v,p,1} \rightarrow p \rightarrow f_{c,p,2} \cdots$$

for some subroutes $\Gamma_{c,1}$ and $\Gamma_{v,1}$ that do not pass through p. We can repeat this argument inductively to show that the route in G must have the form

$$\begin{gathered}
d_{r,y} \rightarrow p \rightarrow f_{c,p,1} \rightarrow \Gamma_{c,1} \rightarrow f_{c,p,1} \rightarrow p \rightarrow f_{v,p,1} \rightarrow \Gamma_{v,1} \rightarrow f_{v,p,1} \rightarrow p \rightarrow f_{c,p,2} \cdots \\
\vdots \\
f_{c,p,z} \rightarrow \Gamma_{c,z} \rightarrow f_{c,p,z} \rightarrow p \rightarrow f_{v,p,z} \rightarrow \Gamma_{v,z} \rightarrow f_{v,p,z} \rightarrow p \rightarrow f_{v,p,z+1} \cdots \\
\vdots \\
f_{c,p,Z} \rightarrow \Gamma_{c,Z} \rightarrow f_{c,p,Z} \rightarrow p \rightarrow f_{v,p,Z} \rightarrow \Gamma_{v,Z} \rightarrow f_{v,p,Z} \rightarrow p \rightarrow d_{r,y}
\end{gathered}$$

for some subroutes $\Gamma_{c,1},\ldots,\Gamma_{c,z},\Gamma_{v,z},\ldots,\Gamma_{v,Z}$ that do not pass through p.

We now consider the subroutes $\Gamma_{c,z}$ and $\Gamma_{v,z}$. There are two cases.

A. Neither $\Gamma_{c,z}$ nor $\Gamma_{v,z}$ pass through path node p' for any $p' \neq p$. This can happen only if the route for demand $d_{r,y}$ passes through both $f_{c,p,z}$ and $f_{v,p,z}$. Hence path p is regular at level z.

B. Either $\Gamma_{c,z}$ or $\Gamma_{v,z}$ pass through some path node $p' \neq p$. Suppose that this is true for $\Gamma_{c,z}$. (The other case is similar.) Then, $\Gamma_{c,z}$ has either the form

$$\dots f_{c,p',z} \to p' \to f_{v,p',z-1} \cdots$$

or else the form

$$\dots f_{c,p',z} \to p' \to f_{v,p',z} \dots .$$

The two cases can be treated similarly. We focus on the former. By Fact 9 since the subroute $\Gamma_{c,z}$ lies between two visits to p, this subroute must return to p' via $f_{v,p',z-1}$. Therefore $\Gamma_{c,z}$ must have the form

$$\begin{gathered}\dots f_{c,p',z} \to p' \to f_{v,p',z-1} \cdots \\ \dots f_{v,p',z-1} \to p' \to f_{c,p',z-1} \dots .\end{gathered}$$

By continuing the argument inductively in this manner (as we did for path p), we can show that $\Gamma_{c,z}$ must eventually have the form $\dots p' \to d_{p'} \dots$. However, Fact 9 implies that the route must return to p' via $d_{p'}$; i.e., the route must have the form $\dots p' \to d_{p'} \dots d_{p'} \to p' \dots$. The only way this can happen is if the route for demand $d_{r,y}$ in T passes through source node $s_{p'}$ as well as destination node $d_{p'}$. Any route in T that does this must have a length at least Z.

If situation A occurs for more than $\sqrt{Z}$ levels, then path p is regular at more than $Z - \sqrt{Z}$; levels, i.e., the demand has type-(4). Otherwise situation B occurs for at least $\sqrt{Z}$ levels which implies that π, the route for demand $d_{r,y}$ in T, has a length at least $Z\sqrt{Z}$; i.e., the demand has type-(3). □

3.2.3. Demands of type (1). Since G is constructed in a random fashion, most nodes in G are far from small cycles of length at most k. More formally, let $B(G)$ be the bad event that there are more than Y type-(1) demands. In this section we show that $B(G)$ does not happen with a constant probability. The analysis resembles that presented in [1]. We begin with a result whose proof is similar to an argument used in the proof of the Erdös–Sachs theorem [10]. (The Erdös–Sachs theorem states that *high-girth* graphs exist. The girth of a graph is the length of its shortest cycle.)

LEMMA 10. *Consider a random graph with ν nodes, and let $\{e_0, e_1, \dots, e_{\kappa-1}\}$ represent a set of $\kappa < k$ potential edges. If*

$$\Pr[e_0 \textit{ exists} | e_1, \dots, e_{\kappa-1} \textit{ exist}] \le \rho$$

for all such sets of edges and $\nu\rho \ge 2$, then the expected number of cycles of fixed length $k' \le k$ is at most $(\nu\rho)^{k'}$. This implies that, with probability $\frac{2}{3}$, the number of cycles of length at most k is at most $3(\nu\rho)^{k+1}$.

Proof. The total number of potential cycles of length k' is at most $\frac{1}{2k'} \frac{\nu!}{(\nu-k')!}$. Each such cycle occurs with probability at most $\rho^{k'}$. Therefore, the expected number of cycles of length k' is at most

$$\frac{\nu! \rho^{k'}}{2k'(\nu - k')!} \le (\nu\rho)^{k'}.$$

This implies that the expected number of cycles of length less than or equal to k is

$$\sum_{k'=1}^{k} (\nu\rho)^{k'} \leq (\nu\rho)^{k+1}$$

since $\nu\rho \geq 2$. By Markov's inequality, with probability $\frac{2}{3}$ the number of cycles of length at most k is at most $3(\nu\rho)^{k+1}$. □

LEMMA 11. $\Pr[B(G)] \leq \frac{1}{3}$*; i.e., with probability at least* $\frac{2}{3}$ *the number of type-*(1) *demands is at most* Y.

Proof. We calculate ν and ρ for the incidence graph G. We begin by counting the number of nodes, ν. The graph G has

- RY demand nodes, $d_{r,y}$;
- at most A_cA_vRY path nodes, p;
- at most $(Q_cI_c + Q_vI_v)Z$ image nodes.

Therefore, the number of nodes in G is at most $\nu \leq RY + A_cA_vRY + (Q_cI_c+Q_vI_v)Z$ which is crudely upper bounded by $Q_cA_cQ_vA_vRYZ$. It is also easy to verify that each node has a degree at most $\max\{A_cA_v, 2Z+1, A_c, A_v\}$ which is crudely upper bounded by A_cA_vZ. We now calculate the probability that a potential edge in G exists. If a fixed set of up to k edges already exist, then the following hold.

- The probability that demand node $d_{r,y}$ is connected to path node p is at most $1/(Y-k)$. (This is because for any accepting interaction (r, a_c, a_v) there is a random matching between the demands $d_{r,y}$ and the canonical paths corresponding to (r, a_c, a_v). See section 2.2.)
- The probability that path node p is connected to any given image node is at most $1/(\min\{I_c, I_v\} - k) \leq 1/(Y-k)$.

Hence for the incidence graph G, $\rho = 1/(Y-k)$.

By the definitions of Y and k in (14) and (13), we ensure that Y is (exponentially) larger than k. Hence $2 \leq \nu\rho \leq 2RQ_cA_cQ_vA_vZ$. Lemma 10 implies that, with probability $\frac{2}{3}$, the number of cycles of length at most k is at most $(6RQ_cA_cQ_vA_vZ)^{k+1}$.

There are at most k nodes on each such cycle. By our degree bound each node is within distance k of at most $(A_cA_vZ)^k$ other nodes. Hence with probability $\frac{2}{3}$ the number of nodes in G that are within distance k of a node that is part of a cycle of length at most k is at most

$$k(6Q_cA_cQ_vA_vRZ)^{k+1}(A_cA_vZ)^k \leq (6Q_cA_cQ_vA_vRZ)^{2k+1} = Y. \qquad □$$

3.2.4. Demands of types (2) and (3).

LEMMA 12. *If the number of type-*(2) *demands or the number of type-*(3) *demands is at least* $RY/4$*, then some edge in* T *has congestion* $\Omega(h)$.

Proof. Each of the type-(2) demands has a route of length at least k in G which by Lemma 7 implies that they have a route of length at least $k/2$ in T. Therefore the total number of edges used by these routes is at least $RYk/8$. By (15) the number of edges in T is at most $RY7^{\ell}(3Z+1)$. Therefore some edge has a congestion of at least

$$\frac{k}{8 \cdot 7^{\ell}(3Z+1)} = \frac{Z((40 \cdot 8^{\ell})^{h+1})^{\frac{1}{2}}}{8 \cdot 7^{\ell}(3Z+1)} = \Omega(h).$$

Each of the type-(3) demands has a route of length at least $Z\sqrt{Z}$ in T. If the number of type-(3) demands is at least $RY/4$, then some edge has a congestion of at least

$$\frac{Z\sqrt{Z}}{4\cdot 7^{\ell}(3Z+1)} = \frac{Z((40\cdot 8^{\ell})^{h+1})^{\frac{1}{2}}}{4\cdot 7^{\ell}(3Z+1)} = \Omega(h). \qquad \square$$

3.2.5. Type-(4) demands. Suppose the bad event $B(G)$ does not occur and the situation in Lemma 12 does not apply; then there are fewer than Y type-(1) demands, fewer than $RY/4$ type-(2) demands, and fewer than $RY/4$ type-(3) demands. Since there are RY demands in total, Lemma 8 implies at least $RY/4$ demands are of type (4). The routing of these type-(4) demands resembles the special case analyzed in section 3.1 as each demand is routed along a path that coincides with a canonical path of the demand at most levels. We use $\widetilde{\mathcal{T}}$ to denote the "almost canonical" routing of these type-(4) demands in T and $\mathcal{T}$ to denote the canonical solution to T that resembles $\widetilde{\mathcal{T}}$. Following the notation of section 3.1 we use $\mathcal{P}(\mathcal{T},\mathcal{M})$ and $\mathcal{T}(\mathcal{P},\mathcal{M})$ to relate the canonical solutions in P and T, where $\mathcal{M}$ defines the mapping between the two graphs. Therefore, each demand of type (4) has three routes of interest: one under solution $\widetilde{\mathcal{T}}$, one under $\mathcal{T}$, and one under $\mathcal{P}$. We are ready to reapply the analysis of section 3.1 to show $\widetilde{\mathcal{T}}$ has high congestion. We first show two lemmas analogous to Lemma 2.

LEMMA 13. *If type-(4) has more than $RY/4$ demands, then P has a heavy blob under $\mathcal{P}(\mathcal{T},\mathcal{M})$.*

Proof. For the purpose of contradiction let us assume every blob in P is light. At most $A_cD_c/(10A_c)$ demands pass through light edges in any c-blob in P; at most $A_vD_v/(10A_v)$ demands pass through light edges in any v-blob in P. Summing over all blobs, at most $RY/5$ demands in total pass through light edges in P. Therefore, at least

$$\frac{RY}{4} - \frac{RY}{5} = \frac{RY}{20}$$

demands pass through heavy edges in P, and they do not induce a heavy solution.

We choose one heavy edge uniformly at random in each query blob in P. At least $RY/(20h^2)$ demands are expected to go through these randomly chosen heavy edges only. As argued in the proof of Lemma 2, this implies that at least a fraction of $1/(20h^2)$ query pairs have accepting answers to the verifier. By the choice of h, $1/(20h^2)$ is higher than the error probability $\alpha^{-\ell}$. This contradicts Theorem 1 since ϕ is a no-instance. Therefore, the $RY/4$ canonical paths must form a heavy solution in P. $\square$

As in section 3.1 we now define a set of bad events. Let q be a query blob in P, let $H = \{a_1, \ldots, a_h\}$ be a set of h answer edges, and let E_{a_i} for $1 \le i \le h$ be a set of $D/10A$ canonical paths in P that pass through the answer edge a_i. For each such path p let L_p be a set of $Z - \sqrt{Z}$ levels that we say are *marked* for path p. Let L be the set of all sets L_p. Let $B(q, E_{a_1}, \ldots, E_{a_h}, L)$ be the bad event that under $\mathcal{M}$, the images in T of the paths in $E_{a_1}, \ldots, E_{a_h}$ have congestion less than $h/4$, *even if a path only contributes to congestion at a level at which it is marked*. We wish to analyze the probability of these bad events. The analysis is complicated by the fact that different paths may be marked at different levels. However, we make use of the following fact.

LEMMA 14. *For any $q, E_{a_1}, \ldots, E_{a_h}, L$ we can find a set of $Z/4$ levels such that at each of these levels z there are $h/4$ sets E_{a_i} that have at least $D/20A$ canonical paths that are marked at level z.*

Proof. Consider one particular set E_{a_i}. We first show that there exist $Z/2$ levels each of which has $D/(20A)$ paths in E_{a_i} that are marked at this level. (We stress that these paths need not be the same at every level.) To see this, the number of marked levels per path multiplied by the number of paths in E_{a_i} is at least $(Z-\sqrt{Z})D/(10A)$. Let x_z be the number of marked paths at level z. If $Z/2$ of these x_z's are smaller than $D/(20A)$, then $\sum_z x_z$ is less than $(Z/2)D/(20A) + (Z/2)D/(10A)$ which is less than $(Z-\sqrt{Z})D/(10A)$. This is a contradiction.

For each level z we now let y_z be the number of sets E_{a_i} that have $D/(20A)$ paths marked at level z. We have just shown that $\sum_z y_z \geq hZ/2$. If $3Z/4$ of the y_z's are smaller than $h/4$, then $\sum_z y_z < (3Z/4)(h/4) + (Z/4)h < hZ/2$, which is a contradiction. Therefore, we have $Z/4$ levels such that at each level z there are $h/4$ sets E_{a_i} each of which has at least $D/20A$ canonical paths marked at level z. (We stress that these $h/4$ sets may be different at different levels.) □

We can now analyze the probability that bad event $B(q, E_{a_1}, \ldots, E_{a_h}, L)$ occurs by concentrating on the $Z/4$ levels whose existence is guaranteed by Lemma 14. We derive the probability by applying a small variation of Lemma 3. The variation is due to the fact that for a heavy blob we now consider only $Z/4$ levels instead of Z levels in T, we consider only $h/4$ heavy edges instead of h per level, and we consider only $D/(20A)$ paths instead of $D/(10A)$ on a heavy edge at each level. We obtain the following.

Lemma 15. *For fixed $q, E_{a_1}, \ldots, E_{a_h}, L$, the probability that $B(q, E_{a_1}, \ldots, E_{a_h}, L)$ occurs is at most*

$$e^{-(40A_c)^{-h/4-1}D_cZ/4} \leq e^{-D_cZ^{2/3}} \quad \text{if } q \text{ is a c-blob},$$
$$e^{-(40A_v4^{\ell})^{-h/4-1}D_vZ/4} \leq e^{-D_vZ^{2/3}} \quad \text{if } q \text{ is a v-blob}.$$

For any path, the number of ways to choose $Z-\sqrt{Z}$ marked levels is $\binom{Z}{\sqrt{Z}}$. Therefore, the total number of events of the form $B(q, E_{a_1}, \ldots, E_{a_h}, L)$ for a v-blob q is upper bounded by

$$\begin{aligned} Q_v\binom{A_v}{h}\binom{D_v4^{\ell}}{D_v/(10A_v)}^h\binom{Z}{\sqrt{Z}}^{hD_v/(10A_v)} &= e^{o(D_v)}\cdot e^{(\log\sqrt{Z}+1)\sqrt{Z}D_vh/(10A_v)} \\ &= e^{o(Z^{2/3}D_v)}. \end{aligned}$$

The first equality holds due to the analysis in section 3.1. Similarly, the total number of events of the form $B(q, E_{a_1}, \ldots, E_{a_h}, L)$ for a c-blob q is upper bounded by

$$Q_c\binom{A_c}{h}\binom{D_c}{D_c/(10A_c)}^h\binom{Z}{\sqrt{Z}}^{hD_c/(10A_c)} = e^{o(Z^{2/3}D_c)}.$$

Hence by a union bound, the probability that *some* event $B(q, E_{a_1}, \ldots, E_{a_h}, L)$ happens is at most $e^{-D_cZ^{2/3}}e^{o(D_cZ^{2/3})} + e^{-D_vZ^{2/3}}e^{o(D_vZ^{2/3})}$, which is $1/\text{poly}(n)$.

Now suppose that under mapping $\mathcal{M}$ no such bad event occurs. Consider the solution $\widetilde{\mathcal{T}}$ in T and the related canonical solution $\mathcal{T}$. For any canonical path in $\mathcal{T}$ we mark $Z-\sqrt{Z}$ levels at which the path is regular in the solution $\widetilde{\mathcal{T}}$. (Since all demands routed in $\widetilde{\mathcal{T}}$ are type-(4) demands, this is always possible.) An important observation is that the congestion in $\widetilde{\mathcal{T}}$ is at least the congestion in T *under the assumption that a path contributes to congestion only at a level at which it is marked*. This is because a path in $\widetilde{\mathcal{T}}$ and the related canonical path in $\mathcal{T}$ pass through the same image edges at all levels at which the canonical path is marked.

We know from Lemma 13 that the solution $\mathcal{P}(\mathcal{T}, \mathcal{M})$ is heavy. By definition we can find a query blob q with h answer edges such that for each such edge $D/10A$

demands are routed on a canonical path that passes through the edge. Since no bad event occurs for this particular $\mathcal{M}$, the images of these canonical paths in T have congestion at least $h/4$ even if a path contributes to congestion only at a level at which it is marked. By our earlier remarks this implies that the solution $\widetilde{\mathcal{T}}$ has a congestion of at least $h/4$. We now have the equivalent of Theorem 6.

THEOREM 16. *If type* (4) *has more than $RY/4$ demands then with probability* $1 - 1/poly(n)$*, $\widetilde{\mathcal{T}}$ has congestion at least $h/4$.*

Combining Lemmas 8, 11, and 12 and Theorem 16, we have the following.

THEOREM 17. *If ϕ is a no-instance, T has congestion $\Omega(h)$ with a constant probability.*

3.3. Wrapping up. Recall that if ϕ is a yes-instance, then all demands can be routed with congestion 1. Therefore the "gap" between the congestion in the yes-instance case and the congestion in the no-instance case is $\Omega(h)$. We now describe this gap in terms of the size of T. Recall that M, the number of edges in T, is defined in (15). From the parameter definitions in section 2.3, it is easy to see that $\log M = O(\log Y)$ and $\log\log M = O(\log Z)$. Since $\log Z = \Theta(\ell h)$, $\ell = \Theta(\log\log\log n)$, and $h = \Theta(\log\log n/\log\log\log n)$, we have

$$\log\log M/\log\log\log M = O(h).$$

We also compute the complexity of our reduction. As discussed at the end of section 2.3 we have

$$M = e^{\text{polylog}(n)};$$

i.e., the reduction can be carried out in quasi-polynomial time. We have therefore shown that, for some constant γ, there is no $\gamma\log\log M/\log\log\log M$-approximation for the undirected congestion minimization problem unless $NP \subseteq coRTIME(n^{\text{polylog}(n)})$. A standard result states that if $NP \subseteq coRTIME(n^{\text{polylog}(n)})$, then $NP \subseteq ZPTIME(n^{\text{polylog}(n)})$; i.e., NP has randomized algorithms that always give the correct answer and have quasi-polynomial running time.

THEOREM 18. *There is no $\gamma\log\log M/\log\log\log M$-approximation for* MINCONGESTION *unless $NP \subseteq ZPTIME(n^{\text{polylog}(n)})$, where M is the size of the graph and γ is some positive constant.*

3.4. Remarks. Our construction utilized a large number of parameters. We now briefly discuss how the values of these parameters are chosen. Recall ℓ is the repetition parameter of the Raz verifier. We need the error probability $\alpha^{-\ell}$ to be smaller than $4/(5h^2)$ to reach a contradiction in Lemma 2. We also require ℓh to be $O(\log\log n)$ so that the reduction is quasi polynomial. In addition, we have shown that $\Theta(h)$ is the inapproximability gap. Therefore, we choose $h = \Theta(\log\log n/\log\log\log n)$, which is as large as possible. This in turn implies that $\ell = \Theta(\log\log\log n)$. Once ℓ is fixed, the Raz verifier related parameters (2)–(10) are fixed as well.

The value of Z is determined by Corollary 4 and Lemma 15 in the probabilistic analysis. The value of Y upper bounds the number of type-(1) demands and is determined in Lemma 11. The value of k is essentially $Z \cdot \zeta$, where $Z - \zeta$ is the number of regular levels in the definition of type-(3) and type-(4) demands. We note that ζ cannot be too small since Lemma 12 requires $\zeta = \Omega(h \cdot 7^{\ell})$. Meanwhile, ζ cannot be too big since the union bound that counts the number of bad events of the form $B(q, E_{a_1}, \ldots, E_{a_h}, L)$ has a factor $\binom{Z}{\zeta}^{hD_v/(10A_v)}$ and the number of bad events needs to be smaller than $e^{D_c Z^{2/3}}$. We choose $\zeta = \sqrt{Z}$. Our analysis works for $\zeta = Z^c$, where c can be any constant in $(0, 2/3)$, though the exact choice of the exponent does not affect the inapproximability gap of $\Theta(h)$.

3.5. Integrality gap. We conclude by showing that our example also gives a lower bound on the integrality gap for MinCongestion. As we mentioned in the Introduction, applying randomized rounding to the linear relaxation of MinCongestion yields a logarithmic approximation of the problem [12]. We observe that our construction of the problem instance on T yields a fractional solution of congestion of at most 1, even for no-instances. To see this, note that each demand has 7^ℓ canonical paths. Suppose that each canonical path carries a $7^{-\ell}$ fraction of the demand. By construction, if two canonical paths share an image edge in T, their corresponding canonical paths in P cannot share an answer edge in P. Since a c-blob in P has 7^ℓ answer edges and a v-blob has 2^ℓ answer edges, at most 7^ℓ canonical paths can share an image edge in T. Therefore, the congestion on an image edge in T is at most 1. The congestion on other edges is $7^{-\ell}$ since canonical paths can share only image edges in T.

As outlined above, for no-instances all integral solutions have congestion $\Omega(h)$ with high probability. Therefore, there exists an instance for which there is a fractional solution with 1, but for all integral solutions the congestion is $O(h) = O(\log\log M/\log\log\log M)$. Hence, the following holds.

Corollary 19. *The linear relaxation of* MinCongestion *has an integrality gap of* $\gamma \log\log M/\log\log\log M$ *for some positive constant* γ.

4. Conclusions. In this paper we have shown that for some constant γ there is no $\gamma \log\log M/\log\log\log M$, approximation for the undirected congestion minimization problem unless $NP \subseteq ZPTIME(n^{\mathrm{polylog}(n)})$. Our techniques also apply to directed graphs. However, the resulting gap of $\gamma \log\log M/\log\log\log M$ is weaker than that of [9].

A number of open problems remain. First, there is still an exponential gap between the best-known $O(\log M/\log\log M)$-approximation algorithm for MinCongestion and our $\gamma \log\log M/\log\log\log M$ hardness factor. Second, our reduction is quasi polynomial and randomized. It would be interesting to obtain a deterministic polynomial-time reduction. Although we believe that it would be possible to use deterministic constructions of high-girth graphs to create a deterministic instance of MinCongestion in which the bad event $B(G)$ does not hold, we know of no way to do this so that the bad events $B(q, E_{a_1}, \dots, E_{a_h}, L)$ do not hold.

Acknowledgment. The authors wish to thank Phil Whiting and Carl Nuzman for their help in analyzing the balls-and-bins game in section 3.1.

REFERENCES

[1] M. Andrews, *Hardness of buy-at-bulk network design*, in Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, Rome, Italy, 2004, pp. 115–124.

[2] M. Andrews, J. Chuzhoy, S. Khanna, and L. Zhang, *Hardness of the undirected edge-disjoint paths problem with congestion*, in Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, Pittsburgh, PA, 2005.

[3] M. Andrews and L. Zhang, *Bounds on fiber minimization in optical networks with fixed fiber capacity*, in Proceedings of IEEE INFOCOM '05, Miami, FL, 2005.

[4] M. Andrews and L. Zhang, *Hardness of the undirected edge-disjoint paths problem*, in Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, 2005.

[5] M. Andrews and L. Zhang, *Complexity of wavelength assignment in optical network optimization*, in Proceedings of IEEE INFOCOM '06, Barcelona, Spain, 2006.

[6] M. Andrews and L. Zhang, *Logarithmic hardness of the directed congestion minimization problem*, in Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, 2006.

[7] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, *Proof verification and the hardness of approximation problems*, J. ACM, 45 (1998), pp. 501–555.

[8] J. Chuzhoy and S. Khanna, *Hardness of Direct Routing with Congestion*, report TR06-109 ECCC, 2006. Available online at http://eccc.hpi-web.de/eccc/.

[9] J. Chuzhoy and J. Naor, *New hardness results for congestion minimization and machine scheduling*, in Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, 2004, pp. 28–34.

[10] P. Erdös and H. Sachs, *Reguläre graphen gegebener Taillenweite mit minimaler Knotenzahl*. Wiss. Z. Uni. Halle-Wittenburg (Math. Nat.), 12 (1963), pp. 251–257.

[11] R. M. Karp, *Reducibility among combinatorial problems*, in Complexity of Computer Computations, R. E. Miller and J. W. Thatcher, eds., Plenum Press, New York, 1972, pp. 85–103.

[12] P. Raghavan and C. D. Thompson, *Randomized rounding: A technique for provably good algorithms and algorithmic proofs*, Combinatorica, 7 (1991), pp. 365–374.

[13] R. Raz, *A parallel repetition theorem*, SIAM J. Comput., 27 (1998), pp. 763–803.

SIAM J. COMPUT.
Vol. 37, No. 1, pp. 132–163

CONSTRAINT SATISFACTION, LOGIC AND FORBIDDEN PATTERNS*

FLORENT MADELAINE† AND IAIN A. STEWART†

This paper is dedicated to the memory of our friend and colleague Clemens Lautemann

Abstract. In the 1990s, Feder and Vardi attempted to find a large subclass of NP which exhibits a dichotomy, that is, where every problem in the subclass is either solvable in polynomial-time or NP-complete. Their studies resulted in a candidate class of problems, namely, those definable in the logic MMSNP. While it remains open as to whether MMSNP exhibits a dichotomy, for various reasons it remains a strong candidate. Feder and Vardi added to the significance of MMSNP by proving that, although MMSNP strictly contains CSP, the class of constraint satisfaction problems, MMSNP and CSP are computationally equivalent. We introduce here a new class of combinatorial problems, the class of forbidden patterns problems FPP, and characterize MMSNP as the finite unions of problems from FPP. We use our characterization to detail exactly those problems that are in MMSNP but not in CSP. Furthermore, given a problem in MMSNP, we are able to decide whether the problem is in CSP or not (this whole process is effective). If the problem is in CSP, then we can construct a template for this problem; otherwise, for any given candidate for the role of template, we can build a counterexample (again, this process is effective).

Key words. finite model theory, constraint satisfaction, existential monadic second-order logic

AMS subject classifications. 68Q19, 68Q15, 0v3C13

DOI. 10.1137/050634840

1. Introduction.

Descriptive complexity theory seeks to classify problems, i.e., classes of finite structures, as to whether they can be defined using formulae of some specific logic, in relation to their computational complexity. One of the seminal results in descriptive complexity is Fagin's theorem [10], which states that a problem can be defined in existential second-order logic if and only if it is in the complexity class NP (throughout we equate a logic with the class of problems definable by the sentences of that logic). In a relatively recent paper and based upon Fagin's characterization of NP, Feder and Vardi [15] attempted to find a large (syntactically defined) subclass of NP which exhibits a dichotomy, that is, where every problem in the subclass is either solvable in polynomial-time or NP-complete (recall Ladner's theorem [22, 26], which states that if P $\neq$ NP, then there is an infinite number of distinct polynomial-time equivalence classes in NP). What emerged from Feder and Vardi's consideration was a (candidate) class of problems called MMSNP, defined by imposing syntactic restrictions upon the existential fragment of second-order logic. Their focus on a fragment of existential second-order logic was so that they might apply tools and techniques of finite model theory to possibly obtain a dichotomy result.

The logic MMSNP is defined by insisting that formulae of the fragment SNP of existential second-order logic must in addition be monotone, be monadic, and not involve inequalities (full definitions follow later). Feder and Vardi considered the imposition of combinations of these three restrictions (monadic, monotone, and without

*Received by the editors June 30, 2005; accepted for publication (in revised form) December 27, 2006; published electronically April 10, 2007.
http://www.siam.org/journals/sicomp/37-1/63484.html

†Department of Computer Science, University of Durham, Science Labs, South Road, Durham DH1 3LE, UK (f.r.madelaine@durham.ac.uk, i.a.stewart@durham.ac.uk). Much of this work was undertaken while the first author was at the Université de la Réunion.

inequalities) and showed that under any combination, excepting the imposition of all three restrictions, the resulting logic does not have a dichotomy (assuming P $\neq$ NP). They were unable to make any similar claim about the logic obtained by imposing all three restrictions. However, they proved that MMSNP properly contains CSP, the class of combinatorial problems known as *constraint satisfaction problems* and, further, that the two classes are closely related in a computational sense.

THEOREM 1 (Feder and Vardi [15]). *Every problem in CSP is definable by a sentence of MMSNP, and every problem definable by a sentence of MMSNP is computationally equivalent to a problem in CSP.*

(By "computationally equivalent" above we mean that the MMSNP problem can be reduced to the CSP problem by a randomized polynomial-time Turing reduction, and the CSP problem can be reduced to the MMSNP problem by a polynomial-time Karp reduction.)[1]

The class CSP of constraint satisfaction problems is of great importance in computer science and artificial intelligence and has strong ties with database theory, graph theory, and universal algebra (see, for instance, [7, 30, 18, 20, 21]). For example, it is well-known that constraint satisfaction problems can be modeled in terms of the existence of homomorphisms between structures [21], in that every constraint satisfaction problem can be realized as the class of structures for which there exists a homomorphism to some fixed template structure. The close relationship between CSP and MMSNP prompted Feder and Vardi [15] to make explicit their conjecture that every problem in CSP is either NP-complete or solvable in polynomial-time. There are numerous results supporting this conjecture. For example, Schaefer [30] proved that if the template structure corresponding to some constraint satisfaction problem has size 2, then the conjecture holds, with Bulatov [3] recently extending Schaefer's result to templates of size 3. Also, Hell and Nešetřil [18] proved that the conjecture holds for all constraint satisfaction problems involving undirected graphs. Various other related dichotomy results have recently been determined; see, for example, [4, 5, 6, 8, 9, 11, 12, 13, 14, 15, 27, 28].

It is with the "border" between CSP and MMSNP that we are concerned in this paper. Feder and Vardi exhibited specific problems in MMSNP that are not in CSP, with their proofs relying essentially on counting arguments (they did not examine in any detail the inclusion relationship between CSP and MMSNP as classes of problems). We gave more examples of such problems in [25] although our proofs were of a different nature; they involved the explicit construction of particular families of graphs. We attempt in this paper to generalize the constructions in [25] so that we might develop a method by which we can ascertain whether *any* problem definable in MMSNP is in CSP or not. To this end, we give a new combinatorial characterization of MMSNP as the class of finite unions of *forbidden patterns problems* (from the class FPP). We use our new combinatorial characterization to answer the following questions in the affirmative: "*Can we characterize exactly those problems that are in MMSNP but not in CSP*?"; "*given a problem in MMSNP, is it decidable whether it is in CSP or not*?"; and "*if a problem in MMSNP can be shown to be in CSP then can we construct a template witnessing its inclusion in CSP*?"

As we shall see, forbidden patterns problems are given by representations that involve a finite set of colored structures, and we introduce the key notion of a recoloring between representations. The notions of a representation and a recoloring somehow generalize the notion of a structure and a homomorphism. The concept of a recoloring,

[1] Gábor Kun has recently derandomized this computational equivalence.

together with two notions that were implicitly present in the proof of Theorem 1 (the notion of a template of a representation and of a Feder–Vardi transformation), allow us to derive for *any* forbidden patterns problem a normal representation. Given any normal representation, we are then able to decide (according to simple criteria) whether the corresponding problem is in CSP or not. If it is in CSP, then we show how to construct its template; if it is not, then we show how to construct a counterexample to any potential template. Finally, we extend these results about problems in FPP to answer the questions (about MMSNP) above.

We end this section with a brief word about MMSNP and our research direction. The logic MMSNP has recently been shown to be related to constraint satisfaction problems where the template is infinite. In particular, Bodirsky and Dalmau [2] have shown that any problem in MMSNP that is nontrivial and closed under disjoint unions can be realized as a constraint satisfaction problem with an ω-categorial template. As regards our interest in the differences between MMSNP and CSP, there are numerous decidability investigations into the relative expressibilities of different logics in the literature, and we highlight a selection of these investigations here. In [1], Benedikt and Segoufin extend the well-known result that on strings, it is decidable whether a monadic second-order problem (that is, a regular language) is definable in first-order logic, to trees. In [16], Gaifman et al. show that the problem of deciding whether a given Datalog program is equivalent to one without recursion (and therefore to a formula of existential positive first-order logic) is undecidable. Finally, one very recent (and pertinent) result is that the problem of deciding whether a constraint satisfaction problem is first-order definable is decidable; indeed, it is NP-complete [23]. It turns out that first-order definable constraint satisfaction problems are forbidden patterns problems with a single color (logically, they correspond to the first-order fragment of MMSNP). The dual question (that asks, given such a forbidden patterns problem, whether it is a constraint satisfaction problem or not) is directly related to a popular notion in structural combinatorics, namely, that of a duality pair. Duality pairs have been characterized by Tardif and Nešetřil [31].

This paper is organized as follows. In the next section, we formally define CSP and FPP. In section 3, we recall the definition of Feder and Vardi's logic MMSNP and show how it relates to the class of problems FPP. In section 4, we introduce normal representations and related notions. In section 5, we prove our main result, i.e., an exact characterization of problems in FPP as to whether they are in CSP or not, provided that they can be given by connected representations. Next, in section 6, we extend this result to the disconnected case (this requires us to generalize normal representations to what we call normal sets) and then extend our results from FPP to MMSNP. Finally, in section 7, we conclude with some closing remarks.

2. Preliminaries.

In this section, we give precise definitions of many of the concepts involved in this paper. We define many well-known notions in a slightly nonstandard way as many of these notions are extended very soon to analogous ones for colored structures.

Structures. A *signature* is a finite set of relation symbols (with each relation symbol having some finite arity). Let σ denote some fixed signature. A *σ-structure* $\mathcal{A}$ consists of a nonempty set A, the *domain*, together with an interpretation $R^{\mathcal{A}} \subseteq A^m$, for every m-ary relation symbol R in σ. Throughout this paper, we only ever consider finite σ-structures. Hence, in the following we simply write "a structure" instead of "a finite σ-structure." We denote structures by $\mathcal{A}, \mathcal{B}, \mathcal{C}$, etc., and their respective domains by A, B, C, etc., or alternatively by $|\mathcal{A}|, |\mathcal{B}|, |\mathcal{C}|$, etc.

Let $\mathcal{A}$ be a structure. We denote tuples of elements by $\mathbf{s}, \mathbf{t}$, etc., and we write "let $\mathbf{t}$ in A" as an abbreviation for "let $\mathbf{t}$ be a tuple of elements of A." Let R be a relation symbol in σ. We feel free to specify only when it is relevant the precise length of a tuple, and when we write "$R^{\mathcal{A}}(\mathbf{t})$" this automatically implies that the tuple of elements $\mathbf{t}$ has the same length as the arity of the relation symbol R. We write "a tuple $R^{\mathcal{A}}(\mathbf{t})$" as an abbreviation for "a tuple of elements $\mathbf{t}$ in A such that $R^{\mathcal{A}}(\mathbf{t})$ holds." We always use R to refer to a relation symbol of σ unless otherwise stated.

Let $\mathcal{A}$ and $\mathcal{B}$ be two structures. A *homomorphism from $\mathcal{A}$ to $\mathcal{B}$* is a mapping $h : A \to B$ such that for any relation symbol R in σ and for any tuple $R^{\mathcal{A}}(\mathbf{t})$, we have that $R^{\mathcal{B}}(h(\mathbf{t}))$, where $h(\mathbf{t})$ denotes the tuple obtained from $\mathbf{t}$ by a componentwise application of h. To denote that h is a homomorphism from $\mathcal{A}$ to $\mathcal{B}$, we write $\mathcal{A}\xrightarrow{h}\mathcal{B}$. If, furthermore, h is onto (respectively, one-to-one), then h is an *epimorphism* (respectively, a *monomorphism*), and we write $\mathcal{A}\overset{h}{\twoheadrightarrow}\mathcal{B}$ (respectively, $\mathcal{A}\overset{h}{\hookrightarrow}\mathcal{B}$). If both $\mathcal{A}\overset{h}{\twoheadrightarrow}\mathcal{B}$ and $\mathcal{A}\overset{h}{\hookrightarrow}\mathcal{B}$, then we write $\mathcal{A}\overset{h}{\lhook\joinrel\twoheadrightarrow}\mathcal{B}$. If $\mathcal{A}\overset{h}{\lhook\joinrel\twoheadrightarrow}\mathcal{B}$ and $\mathcal{A}\xrightarrow{h^{-1}}\mathcal{B}$, then h is an *isomorphism*, and we write $\mathcal{A} \approx \mathcal{B}$. If there exists a homomorphism (respectively, a monomorphism) of $\mathcal{A}$ to $\mathcal{B}$, then we write $\mathcal{A}\rightarrow\mathcal{B}$ (respectively, $\mathcal{A}\hookrightarrow\mathcal{B}$). When something does not hold, we use the same notation but place a / through the symbol. For example, we write $\mathcal{A} \nrightarrow \mathcal{B}$ if it is not the case that $\mathcal{A}\rightarrow\mathcal{B}$.

If $\mathcal{A}\overset{h}{\hookrightarrow}\mathcal{B}$, then $\mathcal{A}$ is a *substructure* of $\mathcal{B}$, and if, furthermore, for any tuple $R^{\mathcal{B}}(h(\mathbf{t}))$, we have that $R^{\mathcal{A}}(\mathbf{t})$ holds, then $\mathcal{A}$ is an *induced substructure* of $\mathcal{B}$. If $\mathcal{A}\overset{h}{\twoheadrightarrow}\mathcal{B}$ and every tuple $R^{\mathcal{B}}(\mathbf{t}')$ is in the image of h (more formally, there exists a tuple $\mathbf{t}$ in $\mathcal{A}$ such that $h(\mathbf{t}) = \mathbf{t}'$ and $R^{\mathcal{A}}(\mathbf{t})$ holds), then $\mathcal{B}$ is an *homomorphic image* of $\mathcal{A}$. If $\mathcal{A}\xrightarrow{h}\mathcal{B}$, then the homomorphic image of $\mathcal{A}$ under h, which we denote by $h(\mathcal{A})$, is the substructure of $\mathcal{B}$ that consists only of those tuples $R^{\mathcal{B}}(\mathbf{t}')$ that are in the image of h.

A *retract* of a structure $\mathcal{B}$ is a structure $\mathcal{A}$ for which there are two homomorphisms $\mathcal{A}\overset{i}{\hookrightarrow}\mathcal{B}$ and $\mathcal{B}\overset{s}{\twoheadrightarrow}\mathcal{A}$ such that $s \circ i = \mathrm{id}_{\mathcal{A}}$ (where $\mathrm{id}_{\mathcal{A}}$ denotes the identity homomorphism on $\mathcal{A}$, so, in particular, if $\mathcal{A}$ is a retract of $\mathcal{B}$, then $\mathcal{A}$ is isomorphic to an induced substructure of $\mathcal{B}$). Moreover, $\mathcal{A}$ is a *proper retract* whenever $\mathcal{A} \not\approx \mathcal{B}$. If $\mathcal{B}$ does not have any proper retracts, then $\mathcal{B}$ is *automorphic* (we use the terminology of [17]). An automorphic retract of $\mathcal{B}$ is called a *core*. It is well known that a core is unique up to isomorphism (see [17] or [19]).

Let $\mathcal{A}$ be a structure, let s and t be in A, and let $n \geq 1$. A *path of length n in $\mathcal{A}$ joining s and t* consists of n tuples $R_1^{\mathcal{A}}(\mathbf{t}_1), R_2^{\mathcal{A}}(\mathbf{t}_2), \ldots, R_n^{\mathcal{A}}(\mathbf{t}_n)$ such that each R_i is a relation symbol in σ of arity at least two (these relation symbols need not be distinct nor need the tuples), s occurs in $\mathbf{t}_1$, t occurs in $\mathbf{t}_n$, and for every $1 \leq i < n$, the tuples of elements $\mathbf{t}_i$ and $\mathbf{t}_{i+1}$ have a common element. If a path joins two distinct elements s and t, then they are *connected*. A structure $\mathcal{A}$ is *connected* if and only if any two distinct elements are connected.

Let $\mathcal{B}$ and $\mathcal{C}$ be two substructures of $\mathcal{A}$ and let $x \in A$. If

- $B \cap C = \{x\}$;
- $B \cup C = A$;
- for every relation symbol R of σ that has arity at least two and for every tuple $R^{\mathcal{A}}(\mathbf{t})$, either $R^{\mathcal{B}}(\mathbf{t})$ or $R^{\mathcal{C}}(\mathbf{t})$ holds but not both;
- for every monadic symbol M and for every element y in B (respectively, C), $M(y)$ holds in $\mathcal{B}$ (respectively, $\mathcal{C}$) if and only if $M(y)$ holds in $\mathcal{A}$; and
- each substructure $\mathcal{B}$ and $\mathcal{C}$ has at least one tuple $R(\mathbf{t})$ (where R has arity at least two),

then we say that $\mathcal{A}$ admits a *decomposition with components $\mathcal{B}$ and $\mathcal{C}$ in the articu-*

lation point x, and we write $\mathcal{A} = \mathcal{B}\overset{x}{32}\ \mathcal{C}$. If $\mathcal{A}$ is connected and does not admit any decomposition, then $\mathcal{A}$ is *biconnected*.

Let $\mathcal{A}$ be a structure. A tuple $R^{\mathcal{A}}(\mathbf{t})$ is said to be *antireflexive* if and only if no element in $\mathbf{t}$ occurs more than once. A *cycle of size* 1 in $\mathcal{A}$ consists of one tuple $R^{\mathcal{A}}(\mathbf{t})$ that is not antireflexive. An element that occurs more than once in a cycle of size 1 is called an *articulation point* of the cycle. A *cycle of size* 2 in $\mathcal{A}$ consists of two antireflexive tuples $R_1^{\mathcal{A}}(\mathbf{t}_1)$ and $R_2^{\mathcal{A}}(\mathbf{t}_2)$, for which we have that if $R_1 = R_2$, then $\mathbf{t}_1$ and $\mathbf{t}_2$ differ and which have at least two distinct common elements, each of which is called an *articulation point* of the cycle. Let $n > 2$. A *cycle of size* n in $\mathcal{A}$ consists of n tuples $R_1^{\mathcal{A}}(\mathbf{t}_1), R_2^{\mathcal{A}}(\mathbf{t}_2), \ldots, R_n^{\mathcal{A}}(\mathbf{t}_n)$ such that:

- for every $1 \leq i \leq n$, the tuple $R_i^{\mathcal{A}}(\mathbf{t}_i)$ is antireflexive;
- for every $1 \leq i < j \leq n$, if $j = i + 1$ or ($i = 1$ and $j = n$), the tuples $\mathbf{t}_i$ and $\mathbf{t}_j$ have one, and only one, common element $a_{i,j}$; otherwise, they have none; and
- the elements $a_{i,j}$, each of which is called an *articulation point* of the cycle, are pairwise distinct.

Colored structures. Let $\mathcal{T}$ be a structure. A *$\mathcal{T}$-colored structure* is a pair $(\mathcal{A}, a)$, where $\mathcal{A}$ is a structure and $\mathcal{A}\xrightarrow{a}\mathcal{T}$. We call: $\mathcal{T}$ the *target* of $(\mathcal{A}, a)$; a the *coloring*; and $\mathcal{A}$ the *underlying structure*. Let $(\mathcal{A}, a)$ and $(\mathcal{B}, b)$ be two $\mathcal{T}$-colored structures. A *$\mathcal{T}$-colored homomorphism* of $(\mathcal{A}, a)$ to $(\mathcal{B}, b)$ is a homomorphism $\mathcal{A}\xrightarrow{h}\mathcal{B}$ such that $a = b \circ h$. All notions defined above extend to $\mathcal{T}$-colored structures, so that colorings are respected by morphisms. For example, a retract of a $\mathcal{T}$-colored structure $(\mathcal{B}, b)$ is a $\mathcal{T}$-colored structure $(\mathcal{A}, a)$ for which there are two homomorphisms $\mathcal{A}\overset{i}{\hookrightarrow}\mathcal{B}$ and $\mathcal{B}\overset{s}{\twoheadrightarrow}\mathcal{A}$ such that $s \circ i = \mathrm{id}_{\mathcal{A}}$, $b \circ i = a$ and $a \circ s = b$. We use the same terminology but add the prefix "$\mathcal{T}$-colored," e.g., as in "$\mathcal{T}$-colored retract," and we use the same notation, e.g., $(\mathcal{A}, a)\xrightarrow{h}(\mathcal{B}, b)$ for a $\mathcal{T}$-colored homomorphism from $(\mathcal{A}, a)$ to $(\mathcal{B}, b)$. However, for simplicity, we may drop the prefix $\mathcal{T}$-colored when it does not cause confusion. At times, we deal with different targets, and so to avoid confusion, we sometimes write the target as a superscript, e.g., as in $(\mathcal{A}, a^{\mathcal{T}})$. We often refer to the elements of $|\mathcal{T}| = T$ as colors. We shall use the following lemmas later on, but we include them here so that readers can familiarize themselves with colored structures.

LEMMA 2. *Let $(\mathcal{A}, a^{\mathcal{T}})$ be a $\mathcal{T}$-colored structure, let $\mathcal{T}'$ be a structure such that $\mathcal{T}'\overset{e}{\hookrightarrow}\mathcal{T}$, and let $(\mathcal{A}, a^{\mathcal{T}'})$ be a $\mathcal{T}'$-colored structure, where $a^{\mathcal{T}} = e \circ a^{\mathcal{T}'}$. If $(\mathcal{A}, a^{\mathcal{T}})$ is automorphic, then $(\mathcal{A}, a^{\mathcal{T}'})$ is automorphic.*

Proof. Suppose that $(\mathcal{A}, a^{\mathcal{T}})$ is automorphic, and suppose that $(\mathcal{B}, b^{\mathcal{T}'})$ is a proper retract of $(\mathcal{A}, a^{\mathcal{T}'})$. That is, we have that $(\mathcal{B}, b^{\mathcal{T}'})\overset{i}{\hookrightarrow}(\mathcal{A}, a^{\mathcal{T}'})$ and $(\mathcal{A}, a^{\mathcal{T}'})\overset{s}{\twoheadrightarrow}(\mathcal{B}, b^{\mathcal{T}'})$, where $s \circ i = \mathrm{id}_{\mathcal{B}}$, $a^{\mathcal{T}'} \circ i = b^{\mathcal{T}'}$, and $b^{\mathcal{T}'} \circ s = a^{\mathcal{T}'}$ (cf. the left commutative diagram of Figure 1) so that $(\mathcal{A}, a^{\mathcal{T}'}) \not\approx (\mathcal{B}, b^{\mathcal{T}'})$. We can compose the two $\mathcal{T}'$-colorings

$$\begin{tikzcd}
\mathcal{B} \arrow[dr, hook, "i"] \arrow[drr, "b^{\mathcal{T}'}"] \arrow[dd, "id_{\mathcal{B}}"'] & & \\
& \mathcal{A} \arrow[r, "a^{\mathcal{T}'}"] \arrow[dl, two heads, "s"] & \mathcal{T}' \\
\mathcal{B} \arrow[urr, "b^{\mathcal{T}'}"'] & &
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathcal{B} \arrow[dr, hook, "i"] \arrow[drr, "b^{\mathcal{T}} = e \circ b^{\mathcal{T}'}"] \arrow[dd, "id_{\mathcal{B}}"'] & & \\
& \mathcal{A} \arrow[r, "a^{\mathcal{T}} = e \circ a^{\mathcal{T}'}"] \arrow[dl, two heads, "s"] & \mathcal{T} \\
\mathcal{B} \arrow[urr, "b^{\mathcal{T}} = e \circ b^{\mathcal{T}'}"'] & &
\end{tikzcd}$$

FIG. 1. *Proof of Lemma* 2.

with e to yield $\mathcal{T}$-colored structures, i.e., let $a^{\mathcal{T}} := e \circ a^{\mathcal{T}'}$ and $b^{\mathcal{T}} := e \circ b^{\mathcal{T}'}$ (cf. the right commutative diagram of Figure 1). Thus, $(\mathcal{B}, b^{\mathcal{T}})$ is a retract of $(\mathcal{A}, a^{\mathcal{T}})$, and so is isomorphic to $(\mathcal{A}, a^{\mathcal{T}})$. Thus, i is an isomorphism, with inverse s. Consequently, $(\mathcal{A}, a^{\mathcal{T}'}) \approx (\mathcal{B}, b^{\mathcal{T}'})$. But $(\mathcal{B}, b^{\mathcal{T}'})$ is a proper retract of $(\mathcal{A}, a^{\mathcal{T}'})$ which yields a contradiction. The result follows. $\square$

The proofs of the next two lemmas are almost identical to analogous proofs in [19], for example, but are included here to allow readers to familiarize themselves with colored structures.

LEMMA 3. *The $\mathcal{T}$-colored structure $(\mathcal{A}, a^{\mathcal{T}})$ is automorphic if and only if whenever $(\mathcal{A}, a^{\mathcal{T}}) \xrightarrow{f} (\mathcal{A}, a^{\mathcal{T}})$, we have that $f(\mathcal{A}, a^{\mathcal{T}}) \approx (\mathcal{A}, a^{\mathcal{T}})$.*

Proof. Assume that $(\mathcal{A}, a^{\mathcal{T}})$ is automorphic and also that $(\mathcal{A}, a^{\mathcal{T}}) \xrightarrow{f} (\mathcal{A}, a^{\mathcal{T}})$. From all such homomorphisms, choose g such that $g(\mathcal{A}, a^{\mathcal{T}})$ has a minimal number of elements and from those structures also a minimal number of tuples. Define h to be g restricted to $g(\mathcal{A}, a^{\mathcal{T}})$.

Note that $g(\mathcal{A}, a^{\mathcal{T}}) \xrightarrow{h} g(\mathcal{A}, a^{\mathcal{T}})$, and so h is one-to-one and onto, as otherwise $(\mathcal{A}, a^{\mathcal{T}}) \xrightarrow{h \circ g} (\mathcal{A}, a^{\mathcal{T}})$ contradicts the minimality of g. So, h is an isomorphism. Thus, $(\mathcal{A}, a^{\mathcal{T}}) \overset{h^{-1} \circ g}{\twoheadrightarrow} g(\mathcal{A}, a^{\mathcal{T}})$ and $g(\mathcal{A}, a^{\mathcal{T}}) \overset{i}{\hookrightarrow} (\mathcal{A}, a^{\mathcal{T}})$, where i is the identity on $g(\mathcal{A}, a^{\mathcal{T}})$. For any $x \in |g(\mathcal{A}, a^{\mathcal{T}})|$, $h^{-1} \circ g \circ i(x) = h^{-1} \circ g(x) = h^{-1} \circ h(x) = x$. Hence, $g(\mathcal{A}, a^{\mathcal{T}})$ is a retract of $(\mathcal{A}, a^{\mathcal{T}})$, and so $g(\mathcal{A}, a^{\mathcal{T}}) \approx (\mathcal{A}, a^{\mathcal{T}})$. Consequently, $f(\mathcal{A}, a^{\mathcal{T}}) \approx (\mathcal{A}, a^{\mathcal{T}})$ by minimality of g.

Conversely, assume that whenever $(\mathcal{A}, a^{\mathcal{T}}) \xrightarrow{f} (\mathcal{A}, a^{\mathcal{T}})$, we have $f(\mathcal{A}, a^{\mathcal{T}}) \approx (\mathcal{A}, a^{\mathcal{T}})$. Suppose that $(\mathcal{B}, b^{\mathcal{T}}) \overset{i}{\hookrightarrow} (\mathcal{A}, a^{\mathcal{T}})$ and $(\mathcal{A}, a^{\mathcal{T}}) \overset{s}{\twoheadrightarrow} (\mathcal{B}, b^{\mathcal{T}})$, with $s \circ i = id_{\mathcal{B}}$. Define $f := i \circ s$. Thus, $f(\mathcal{A}, a^{\mathcal{T}}) \approx (\mathcal{A}, a^{\mathcal{T}})$, with i an epimorphism and s a monomorphism. Consequently, $(\mathcal{B}, b^{\mathcal{T}}) \approx (\mathcal{A}, a^{\mathcal{T}})$, and $(\mathcal{A}, a^{\mathcal{T}})$ is automorphic. $\square$

LEMMA 4. *Every $\mathcal{T}$-colored structure has a $\mathcal{T}$-colored core that is unique up to $\mathcal{T}$-colored isomorphism.*

Proof. Trivially, every $\mathcal{T}$-colored structure has a $\mathcal{T}$-colored core. Suppose that $(\mathcal{A}_1, a_1)$ and $(\mathcal{A}_2, a_2)$ are cores of $(\mathcal{B}, b)$ such that $(\mathcal{A}_1, a_1) \not\approx (\mathcal{A}_2, a_2)$. In particular:

- $(\mathcal{A}_1, a_1) \overset{i_1}{\hookrightarrow} (\mathcal{B}, b)$ and $(\mathcal{B}, b) \overset{s_1}{\twoheadrightarrow} (\mathcal{A}_1, a_1)$ such that $s_1 \circ i_1 = \mathrm{id}_{\mathcal{A}_1}$, $b \circ i_1 = a_1$, and $s_1 \circ a_1 = b$; and
- $(\mathcal{A}_2, a_2) \overset{i_2}{\hookrightarrow} (\mathcal{B}, b)$ and $(\mathcal{B}, b) \overset{s_2}{\twoheadrightarrow} (\mathcal{A}_2, a_2)$ such that $s_2 \circ i_2 = \mathrm{id}_{\mathcal{A}_2}$, $b \circ i_2 = a_2$, and $s_2 \circ a_2 = b$.

Then $f_1 := s_2 \circ i_1 : (\mathcal{A}_1, a_1^{\mathcal{T}}) \to (\mathcal{A}_2, a_2^{\mathcal{T}})$ is a homomorphism as is $f_2 := s_1 \circ i_2 : (\mathcal{A}_2, a_2^{\mathcal{T}}) \to (\mathcal{A}_1, a_1^{\mathcal{T}})$. Hence, by Lemma 3, $f_2 \circ f_1(\mathcal{A}_1, a_1^{\mathcal{T}}) \approx (\mathcal{A}_1, a_1^{\mathcal{T}})$ and $f_1 \circ f_2(\mathcal{A}_2, a_2^{\mathcal{T}}) \approx (\mathcal{A}_2, a_2^{\mathcal{T}})$. Consequently, $(\mathcal{A}_1, a_1^{\mathcal{T}})$ and $(\mathcal{A}_2, a_2^{\mathcal{T}})$ are isomorphic, and the result follows. $\square$

Patterns and representations. A structure $(\mathcal{A}, a^{\mathcal{T}})$ is a *$\mathcal{T}$-pattern* whenever for every $y \in A$, there exists a relation symbol R in σ and a tuple $\mathbf{t}$ in A in which y occurs such that $R^{\mathcal{A}}(\mathbf{t})$ holds (that is, every element occurs in some tuple in some relation of $\mathcal{A}$; i.e., $\mathcal{A}$ has no isolated elements). A $\mathcal{T}$-pattern $(\mathcal{A}, a^{\mathcal{T}})$ is *conform* if and only if $\mathcal{A}$ consists solely of an antireflexive tuple $R^{\mathcal{A}}(\mathbf{t})$: That is, there exists a relation symbol R in σ such that $R^{\mathcal{A}} = \{\mathbf{t}\}$, where every element of A occurs in $\mathbf{t}$ exactly once, and for every other relation symbol R' in σ, we have $R'^{\mathcal{A}} = \emptyset$. We denote conform patterns explicitly as in $(R(\mathbf{t}), a^{\mathcal{T}})$.

A *representation* is a pair $(\mathscr{F}, \mathcal{T})$, where $\mathcal{T}$ is a structure, called the *target*, and $\mathscr{F}$ is a finite set of $\mathcal{T}$-patterns, called the *forbidden patterns*. If every forbidden pattern in $\mathscr{F}$ is connected, then we say that $(\mathscr{F}, \mathcal{T})$ is *connected*. Let $(\mathscr{F}, \mathcal{T})$ be a representation. A $\mathcal{T}$-colored structure $(\mathcal{A}, a^{\mathcal{T}})$ is *valid* (respectively, *weakly valid*) with respect to $(\mathscr{F}, \mathcal{T})$ if and only if there is no forbidden pattern $(\mathcal{B}, b^{\mathcal{T}}) \in \mathscr{F}$ such that

$(\mathcal{B}, b^{\mathcal{T}}) \to (\mathcal{A}, a^{\mathcal{T}})$ (respectively, $(\mathcal{B}, b^{\mathcal{T}}) \hookrightarrow (\mathcal{A}, a^{\mathcal{T}})$). A structure $\mathcal{A}$ is *valid* (respectively, *weakly valid*) with respect to $(\mathscr{F}, \mathcal{T})$ if and only if there exists a homomorphism $\mathcal{A} \overset{a^{\mathcal{T}}}{\to} \mathcal{T}$ such that $(\mathcal{A}, a^{\mathcal{T}})$ is valid (respectively, weakly valid) with respect to $(\mathscr{F}, \mathcal{T})$.

Constraint satisfaction problems. It is well-known that constraint satisfaction problems can be modeled in terms of the existence of homomorphisms between structures [21]. Recall that the *nonuniform constraint satisfaction problem with template* $\mathcal{T}$, denoted by CSP($\mathcal{T}$), is the problem defined as follows:

- instances: structures $\mathcal{A}$ (over the same signature as $\mathcal{T}$);
- yes instances: those instances $\mathcal{A}$ for which $\mathcal{A} \to \mathcal{T}$.

We denote by CSP the class of nonuniform constraint satisfaction problems. Note that in [21], the adjective "nonuniform" was coined to distinguish such problems from *uniform* constraint satisfaction problems where the template $\mathcal{T}$ is not fixed but may range over a class of structures (all structures in general) and is part of the input. Since we do not deal with uniform problems in this paper, from now on we drop the phrase nonuniform.

Forbidden patterns problems. The *forbidden patterns problem* given by the representation $(\mathscr{F}, \mathcal{T})$, and denoted by FPP$(\mathscr{F}, \mathcal{T})$, is the problem defined as follows:

- instances: structures $\mathcal{A}$ (over the same signature as $\mathcal{T}$);
- yes instances: those instances $\mathcal{A}$ that are valid w.r.t. $(\mathscr{F}, \mathcal{T})$.

We denote by FPP the class of forbidden patterns problems. If two representations define the same forbidden patterns problem, then we say that the representations are *equivalent*.

Remark 5. A problem in CSP is clearly monotone, i.e., closed under substructures. Furthermore, it is closed under inverse homomorphisms. To see this, let $\mathcal{B}$ and $\mathcal{T}$ be two structures. If $\mathcal{B} \in$ CSP($\mathcal{T}$), then $\mathcal{A} \in$ CSP($\mathcal{T}$) for any $\mathcal{A}$ such that $\mathcal{A} \to \mathcal{B}$. It is not difficult to check that if $\mathcal{B} \in$ FPP$(\mathscr{F}, \mathcal{T})$, then $\mathcal{A} \in$ FPP$(\mathscr{F}, \mathcal{T})$ for any $\mathcal{A}$ such that $\mathcal{A} \to \mathcal{B}$. Moreover, note that the containment problem, i.e., given two structures $\mathcal{T}$ and $\mathcal{T}'$, decide whether CSP($\mathcal{T}$) $\subseteq$ CSP($\mathcal{T}'$), is nothing other than the uniform constraint satisfaction problem (as CSP($\mathcal{T}$) $\subseteq$ CSP($\mathcal{T}'$) if and only if $\mathcal{T} \to \mathcal{T}'$).

THEOREM 6. CSP $\subsetneq$ FPP.

Proof. The inclusion is clear, as a problem from CSP with template $\mathcal{T}$ can be given equivalently as the forbidden patterns problem with representation $(\emptyset, \mathcal{T})$. It follows from counterexamples given in [15, 25] that this inclusion is strict. □

This provokes the following question, which is intrinsic to this paper: *When is a forbidden patterns problem not a constraint satisfaction problem*?

3. Feder and Vardi's logic. The logic SNP is the fragment of existential second-order logic, ESO, consisting of formulae Φ of the form $\exists \mathbf{S} \forall \mathbf{t} \varphi$, where $\mathbf{S}$ is a tuple of relation symbols (not in σ), $\mathbf{t}$ is a tuple of (first-order) variables, and φ is quantifier-free. Furthermore: Φ is in *monadic* SNP whenever $\mathbf{S}$ is a tuple of monadic relation symbols; Φ is in *monotone* SNP whenever every occurrence in φ of a symbol R from σ appears in the scope of an odd number of $\neg$ symbols; and Φ is in SNP *without inequalities* whenever the symbol $=$ does not appear in φ (either positively or negatively). If one thinks about the intuitive properties of the existence of a homomorphism from one structure to another, one might find it plausible to consider imposing some of the above restrictions on ESO. For instance, the existence (cf. the existential second-order quantifiers) of a homomorphism from an arbitrary source graph to a fixed target graph is equivalent to finding a partition of the domain of the source graph into sets (cf. the monadic restriction), one for each element of the target graph, so that every edge of the source graph (cf. the universal prefix of first-order quantifiers) maps to an edge of the target graph (cf. the monotone restriction above, reflecting that we are interested only in positive information, that is, about mappings of edges, not about mappings of "nonedges"). The "without inequalities" aspect of MMSNP comes about as homomorphisms do not distinguish between different elements.

Feder and Vardi considered the imposition of combinations of these three restrictions (monadic, monotone, and without inequalities) and showed that under any combination excepting the imposition of all three restrictions, the resulting logic does not have a dichotomy (assuming P $\neq$ NP). However, they were unable to make any similar claim about the logic obtained by imposing all three restrictions, and they observed that this logic subsumes CSP. This motivated the following definition.

DEFINITION 7. *Monotone monadic SNP without inequality (MMSNP) is the fragment of ESO consisting of those formulae Φ of the following form*:

$$\exists \mathbf{M} \forall \mathbf{t} \bigwedge_i \neg\big(\alpha_i(\sigma, \mathbf{t}) \wedge \beta_i(\mathbf{M}, \mathbf{t})\big),$$

where $\mathbf{M}$ *is a tuple of monadic relation symbols (not in* σ*),* $\mathbf{t}$ *is a tuple of (first-order) variables, and for every negated conjunct* $\neg(\alpha_i \wedge \beta_i)$:

- α_i *consists of a conjunction of positive atoms involving relation symbols from* σ *and variables from* $\mathbf{t}$; *and*
- β_i *consists of a conjunction of atoms or negated atoms involving relation symbols from* $\mathbf{M}$ *and variables from* $\mathbf{t}$.

(Notice that the equality symbol does not occur *in* Φ.*)*

Feder and Vardi showed that CSP is subsumed by MMSNP and, furthermore, that MMSNP is computationally equivalent to CSP. (Theorem 8 is a more detailed reformulation of Theorem 1 and is included for completeness.)

THEOREM 8 (Feder and Vardi [15]). *Every problem in CSP is definable by a sentence of MMSNP, but there are problems in MMSNP that are not in CSP. However, for every problem* $\Omega \in$ *MMSNP, there exists a problem* $\Omega' \in$ *CSP such that* Ω *reduces to* Ω' *via a polynomial-time Karp reduction, and* Ω' *reduces to* Ω *via a randomized polynomial-time Turing reduction.*[2]

(A more detailed proof of Theorem 8 than that in [15] can be found in [24].)

In the remainder of this section, we show that the logic MMSNP essentially corresponds to the class FPP of forbidden patterns problems. Let us begin by looking at some illustrative examples.

Example 9. Consider the signature $\sigma_2 = \langle E \rangle$, where E is a binary relation symbol. Define Φ_1 as

$$\begin{aligned}\exists C\, \forall x\, \forall y\, \forall z\, \big(\neg\big(E(x,y) \wedge E(y,z) \wedge E(z,x) \wedge C(x) \wedge C(y) \wedge C(z)\big) \\ \wedge \neg\big(E(x,y) \wedge E(y,z) \wedge E(z,x) \wedge \neg C(x) \wedge \neg C(y) \wedge \neg C(z)\big)\big).\end{aligned}$$

We can easily ascertain that Φ_1 defines the forbidden patterns problem with representation $(\mathscr{F}, \mathcal{T})$, where $|\mathcal{T}| := \{0,1\}$, $E^{\mathcal{T}} := |\mathcal{T}|^2$, and $\mathscr{F}$ contains two forbidden patterns, one for each negated conjunct, both having as the underlying structure a directed triangle (domain $\{x,y,z\}$ and relation $E = \{(x,y),(y,z),(z,x)\}$): In the first forbidden pattern all vertices of this directed triangle are colored 0, whereas in the second forbidden pattern the vertices are all colored 1 (the colorings are given by C and correspond to $x,y,z \mapsto 0$ and $x,y,z \mapsto 1$, respectively, and the colors are the

[2] As mentioned earlier, Gábor Kun has recently derandomized this reduction.

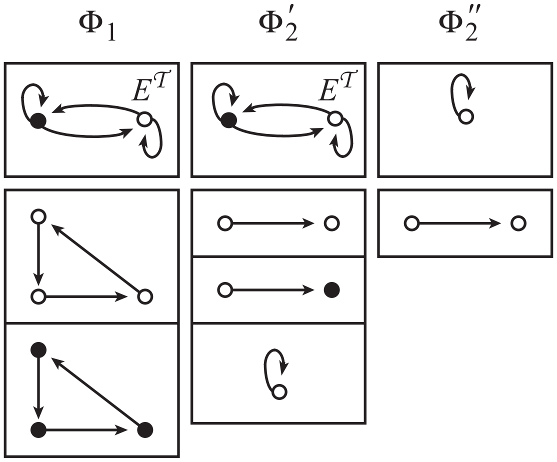


FIG. 2. *Primitive sentence and representations.*

names of the elements of the template). For simplicity, from now on we usually give representations in a pictorial fashion. For example, the representation we have just defined is depicted on the left in Figure 2; the top cell depicts the template, and the other cells depict the forbidden patterns. Note that the template is not a colored structure; however, to depict the homomorphisms from the forbidden patterns to the template, we have colored the elements of the template accordingly.

It is not so clear to which forbidden patterns problem the following sentence corresponds:

$$\Phi_2 := \exists C\ \forall x\ \forall y\ \big(\neg\big(E(x,y) \wedge C(x)\big) \wedge \neg\big(E(x,x) \wedge C(x) \wedge C(y)\big)\big).$$

However, it can be transformed into equivalent sentences as follows. First, we list all possibilities for the monadic predicate, to ensure that we have "fully colored" structures:

$$\exists C\ \forall x\ \forall y\ \big(\neg\big(E(x,y) \wedge C(x) \wedge C(y)\big) \wedge \neg\big(E(x,y) \wedge C(x) \wedge \neg C(y)\big) \\ \wedge \neg\big(E(x,x) \wedge C(x) \wedge \neg C(y)\big)\big).$$

The last negated conjunct is comprised of two "independent" parts, namely, $(E(x,x) \wedge C(x))$ and $C(y)$, and does not correspond to a pattern (y does not appear in any atomic σ-relation). We can rewrite the above formula as the disjunction of two formulae Φ_2' and Φ_2'', where

$$\Phi_2' = \exists C\ \forall x\ \forall y\ \big(\neg\big(E(x,y) \wedge C(x) \wedge C(y)\big) \\ \wedge \neg\big(E(x,y) \wedge C(x) \wedge \neg C(y)\big) \wedge \neg\big(E(x,x) \wedge C(x)\big)\big)$$

and

$$\Phi_2'' = \exists C\ \forall x\ \forall y\ \big(\neg\big(E(x,y) \wedge C(x) \wedge C(y)\big) \\ \wedge \neg\big(E(x,y) \wedge C(x) \wedge \neg C(y)\big) \wedge \neg\big(\neg C(y)\big)\big)$$

(we leave the fact that Φ_2 can be so decomposed as a simple exercise). Now from each formula we can extract a suitable representation: This is easy in the case of Φ_2'; and, in the case of Φ_2'', note that the last negated conjunct essentially forces us to use a single color, so we can ignore all negated conjuncts which mention $\neg C(z)$ for some variable z. Finally, this leads to the representations depicted in the middle and on the right in Figure 2, respectively.

The above examples motivate the following definition and proposition.

DEFINITION 10. *A sentence Φ of MMSNP, where Φ is as in Definition 7, is* primitive *if and only if for every negated conjunct $\neg(\alpha \wedge \beta)$:*

- *for every first-order variable x that occurs in $\neg(\alpha \wedge \beta)$ and for every monadic symbol C in $\mathbf{M}$, exactly one of $C(x)$ and $\neg C(x)$ occurs in β; and*
- *unless x is the only first-order variable that occurs in $\neg(\alpha \wedge \beta)$, an atom of the form $R(\mathbf{t})$, where x occurs in $\mathbf{t}$ and R is a relation symbol from σ, must occur in α.*

PROPOSITION 11. *Every sentence of MMSNP is logically equivalent to a finite disjunction of primitive sentences.*

Proof. Let Φ be a sentence of MMSNP that is not primitive. Assume that Φ does not satisfy the first property of Definition 10. Let $\neg\big(\alpha(\sigma, \mathbf{t}) \wedge \beta(\mathbf{M}, \mathbf{t})\big)$ be a negated conjunct in Φ where there exists a (first-order) variable x that occurs in this negated conjunct and a monadic symbol C in $\mathbf{M}$ such that neither $C(x)$ nor $\neg C(x)$ occurs in β. Replace $\neg(\alpha \wedge \beta)$ in Φ by the conjunction of two negated conjuncts:

$$\neg\big(\alpha \wedge \beta \wedge C(x)\big) \wedge \neg\big(\alpha \wedge \beta \wedge \neg C(x)\big).$$

This new formula belongs to MMSNP and is logically equivalent to Φ. We iterate this process until the sentence satisfies the first property of Definition 10. Let Φ' denote this new sentence.

It may be the case that the second property does not hold for Φ' because of a negated conjunct of the form $\neg\big(\alpha(\sigma, \mathbf{t}) \wedge \beta_0(\mathbf{M}, \mathbf{t}) \wedge \beta_1(\mathbf{M}, x)\big)$, where x does not occur in $\mathbf{t}$, where $\alpha(\sigma, \mathbf{t})$ may be empty, and where β_1 is the conjunction of all atoms and negated atoms of β involving symbols from $\mathbf{M}$ and the variable x (β_0 is a conjunction of the remaining atoms and negated atoms of β). Let $\Phi'' = \Phi'_1 \vee \Phi'_2$, where

- Φ'_1 is obtained from Φ' by replacing $\neg(\alpha \wedge (\beta_0 \wedge \beta_1))$ in Φ' by $\neg(\alpha \wedge \beta_0)$; and
- Φ'_2 is obtained from Φ' by replacing $\neg(\alpha \wedge (\beta_0 \wedge \beta_1))$ in Φ' by $\neg\beta_1$.

First, note that Φ'_1 and Φ'_2 are both in MMSNP. Second, it is not hard to check that Φ'' is logically equivalent to Φ'. We iterate this transformation until each sentence in the disjunction satisfies the second property of Definition 10. □

We are now ready to state exactly what the correspondence is between MMSNP and FPP.

THEOREM 12. *The class of problems captured by the primitive fragment of the logic MMSNP is exactly the class FPP of forbidden patterns problems.*

Proof. Let $\Phi = \exists\mathbf{M}\forall\mathbf{t}\varphi$ be a primitive sentence of MMSNP. We shall build a representation $(\mathscr{F}, \mathcal{T})$ from Φ. A conjunction $\chi(\mathbf{M}, x)$ of atoms and negated atoms involving only relation symbols from $\mathbf{M}$ and the sole first-order variable x, where for each relation symbol C in $\mathbf{M}$, exactly one of $C(x)$ or $\neg C(x)$ occurs, is referred to as an $\mathbf{M}$-color. So, associated with every negated conjunct $\neg(\alpha \wedge \beta)$ in Φ (more precisely, with β in every such negated conjunct) and every variable occurring in this negated conjunct is a unique $\mathbf{M}$-color; in fact, β can be written as the conjunction of these $\mathbf{M}$-colors. Construct the structure $\mathcal{T}$ from Φ as follows:

- Its domain T consists of all $\mathbf{M}$-colors $\chi(\mathbf{M}, x)$ that are not explicitly forbidden in Φ by some negated conjunct $\neg(\alpha \wedge \beta)$ of φ having the form $\neg\chi(\mathbf{M}, x)$, i.e., so that α is empty and β is the $\mathbf{M}$-color χ; and
- for every relation symbol R of arity m in σ, set $R^{\mathcal{T}} := T^m$.

Start with $\mathscr{F} := \emptyset$, and for every negated conjunct $\neg(\alpha \wedge \beta)$ in φ, add to $\mathscr{F}$ the structure $(\mathcal{A}_\alpha, a^{\mathcal{T}}_\beta)$, where

- $\mathcal{A}_\alpha$ is the structure defined as follows:
  - the domain consists of all first-order variables that occur in the negated conjunct $\neg(\alpha \wedge \beta)$; and
  - for every relation symbol R in σ, there is a tuple $R^{\mathcal{A}_\alpha}(\mathbf{t})$ if and only if the atom $R(\mathbf{t})$ appears in α;
- for every $x \in |\mathcal{A}_\alpha|$, set $a^{\mathcal{T}}_\beta(x) := \chi$, where χ is the $\mathbf{M}$-color of x in β.

(The fact that Φ is primitive makes these definitions well-defined.)

Let $\mathcal{B}$ be a structure such that $\mathcal{B} \models \Phi$. So, there exists an assignment $\Pi : \mathbf{M} \to 2^B$ (where 2^B denotes the power set of B) such that $\mathcal{B} \models \forall\mathbf{t}\varphi(\Pi(\mathbf{M}), \mathbf{t})$ (here, $\varphi(\Pi(\mathbf{M}), \mathbf{t})$ denotes the formula φ where every monadic predicate is instantiated as the subset of B given by the assignment Π). Since $\Phi = \exists\mathbf{M}\forall\mathbf{t}\varphi$ is primitive, the formula φ is of the form:

$$\neg\chi_1(\mathbf{M}, x) \wedge \neg\chi_2(\mathbf{M}, x) \wedge \cdots \wedge \neg\chi_k(\mathbf{M}, x) \wedge \psi(\sigma, \mathbf{M}, \mathbf{t}),$$

where $k \geq 0$, and for every $1 \leq i \leq k$, χ_i is an $\mathbf{M}$-color (with all such $\mathbf{M}$-colors distinct) and ψ is a conjunction of negated conjuncts that are not $\mathbf{M}$-colors.

The assignment Π induces a map $\pi^{\mathcal{T}}$ from B to the set T that sends an element $u \in B$ to χ, where χ is the unique $\mathbf{M}$-color for which $\chi(\Pi(\mathbf{M}), u)$ holds (note that $\chi \neq \chi_i$ for $i = 1, 2, \ldots, k$, as $\neg\chi_i(\Pi(\mathbf{M}), u)$ holds for all $u \in B$).

Let $\neg(\alpha \wedge \beta)$ be a negated conjunct of φ, where α is nonempty, and suppose that $(\mathcal{A}_\alpha, a^{\mathcal{T}}_\beta) \xrightarrow{h} (\mathcal{B}, \pi^{\mathcal{T}})$.

Let $R(x_1, x_2, \ldots, x_a)$ be an atom appearing in α. So, $R^{\mathcal{A}_\alpha}(x_1, x_2, \ldots, x_a)$ holds and consequently $R^{\mathcal{B}}(h(x_1), h(x_2), \ldots, h(x_a))$ holds. Thus, if $\mathbf{t}'$ is the tuple of variables appearing in $\neg(\alpha \wedge \beta)$, then $\alpha^{\mathcal{B}}(h(\mathbf{t}'))$ holds. Also, $\pi^{\mathcal{T}} \circ h = a^{\mathcal{T}}_\beta$ and so $\pi^{\mathcal{T}}(h(\mathbf{t}')) = a^{\mathcal{T}}_\beta(\mathbf{t}')$. That is, $\beta^{\mathcal{B}}(\Pi(\mathbf{M}), h(\mathbf{t}'))$ holds. Thus $(\alpha \wedge \beta)^{\mathcal{B}}(\Pi(\mathbf{M}), h(\mathbf{t}'))$ holds, which contradicts the fact that $\mathcal{B} \models \Phi$, witnessed by $\Pi(\mathbf{M})$. Hence, $\mathcal{B} \in \mathrm{FPP}(\mathscr{F}, \mathcal{T})$.

Conversely, suppose that $\mathcal{B} \in \mathrm{FPP}(\mathscr{F}, \mathcal{T})$, witnessed by the homomorphism $\mathcal{B} \xrightarrow{\pi^{\mathcal{T}}} \mathcal{T}$. Clearly, $\pi^{\mathcal{T}}$ gives rise to an assignment $\Pi : \mathbf{M} \to 2^B$, where $u \in \Pi(C)$ for some $C \in \mathbf{M}$ and $u \in B$, if and only if $C(y)$ appears in $\chi(y)$, where $\pi^{\mathcal{T}}(u) = \chi$. Assume that $\mathcal{B} \models \alpha(h(\mathbf{t}')) \wedge \beta(\Pi(\mathbf{M}), h(\mathbf{t}'))$ for some map $h : |\mathcal{A}_\alpha| \to |\mathcal{B}|$, where $\neg(\alpha \wedge \beta)$ is a negated conjunct of φ, and $\mathbf{t}'$ is the tuple of variables appearing in $\alpha \wedge \beta$.

If $R^{\mathcal{A}_\alpha}(x_1, x_2, \ldots, x_a)$ holds, then $R^{\mathcal{B}}(h(x_1), h(x_2), \ldots, h(x_a))$ holds. If β is of the form $\bigwedge_{i=1}^{d} \chi^i(x_i)$, where $\mathbf{t}' = (x_1, x_2, \ldots, x_d)$ and each χ^i is an $\mathbf{M}$-color, then $\chi^i(\Pi(\mathbf{M}), h(x_i))$ holds for each $i = 1, 2, \ldots, d$. However, by definition $a^{\mathcal{T}}_\beta(x_i) = \chi^i$, and so $\pi^{\mathcal{T}}(h(x_i)) = a^{\mathcal{T}}_\beta(x_i)$ for each $i = 1, 2, \ldots, d$. Hence, $(\mathcal{A}_\alpha, a^{\mathcal{T}}_\beta) \xrightarrow{h} (\mathcal{B}, \pi^{\mathcal{T}})$, which yields a contradiction. Thus, $\mathcal{B} \models \Phi$, witnessed by the assignment $\Pi(\mathbf{M})$, and the implication follows.

Conversely, given a representation $(\mathscr{F}, \mathcal{T})$, we shall build a corresponding primitive sentence of MMSNP. Let $\mathbf{M} = \{C_1, C_2, \ldots, C_k\}$ be a set of monadic predicates that are not in σ such that $k = \lceil \log_2 |\mathcal{T}| \rceil$. To each element x_i of $|\mathcal{T}|$, we associate some arbitrary $\mathbf{M}$-color χ_{x_i}. Let $\chi_{|\mathcal{T}|+1}, \ldots, \chi_{2^k}$ denote the remaining $\mathbf{M}$-colors (if $|\mathcal{T}| < 2^k$). Let $\Phi = \exists\mathbf{M}\forall\mathbf{t}\phi$, where $\forall\mathbf{t}\phi$ is the universal closure of the conjunction of the following negated conjuncts:

- If $|\mathcal{T}| < 2^k$, then for every i such that $|\mathcal{T}| < i \leq 2^k$, we add the negated conjunct $\neg\chi_i(y)$.

- For each tuple $R(i_1, i_2, \dots, i_r)$ that does not hold in $\mathcal{T}$, we add the negated conjunct $\neg(R(y_1, y_2, \dots, y_r) \wedge \chi_{i_1}(y_1) \wedge \chi_{i_2}(y_2) \wedge \cdots \wedge \chi_{i_r}(y_r))$, where the variables $y_1, y_2, \dots, y_r$ are pairwise distinct.
- For each forbidden pattern $(\mathcal{A}, a^{\mathcal{T}})$ in $\mathscr{F}$, we add the negated conjunct $\neg(\alpha \wedge \beta)$, where α is the conjunction of the tuples of $\mathcal{F}$, and β is the conjunction $\bigwedge_{x \in |\mathcal{A}|} \chi_{a^{\mathcal{T}}(x)}(x)$.

The first type of negated conjunct ensures that we may use only the $\mathcal{M}$-colors that correspond to elements of $\mathcal{T}$. The second type of negated conjunct describes that there is a homomorphism to $\mathcal{T}$. Finally, the last type of negated conjunct enforces that this homomorphism is not compatible with any of the forbidden patterns. Consequently, a structure $\mathcal{B}$ is a yes instance of the forbidden patterns problem with representation $(\mathscr{F}, \mathcal{T})$ if and only if $\mathcal{B} \models \Phi$. The formal proof of this equivalence is similar to that of the first implication. This concludes the proof. □

By Proposition 11, every forbidden patterns problem is described by a primitive sentence of MMSNP. Since the disjunction of two sentences of MMSNP is logically equivalent to a sentence of MMSNP, we get the following corollary from the above theorem.

COROLLARY 13. *The class of problems captured by the logic MMSNP corresponds exactly to the class of finite unions of problems in FPP.*

4. A normal form for problems in FPP. In this section, we introduce normal representations and show how any representation can be effectively rewritten into an equivalent normal representation. The transformation is achieved through a combination of different operations so as to enforce various properties. We shall make clear later, in section 5, why we need these properties.

However, before we proceed, let us try and give some idea here of the direction of travel by stating the properties we wish to enforce and our intended goal. We shall state the properties again at the appropriate point in the text, as we do with the definition and result stated below. Let $(\mathscr{F}, \mathcal{T})$ be a representation. The properties we wish to enforce upon $(\mathscr{F}, \mathcal{T})$ are as follows.

($\mathfrak{p}1$) Any structure is valid if and only if it is weakly valid.
($\mathfrak{p}2$) Every pattern of $\mathscr{F}$ is automorphic.
($\mathfrak{p}3$) It is not the case that $(\mathcal{B}_1, b_1^{\mathcal{T}}) \hookrightarrow (\mathcal{B}_2, b_2^{\mathcal{T}})$ for any distinct patterns $(\mathcal{B}_1, b_1^{\mathcal{T}})$ and $(\mathcal{B}_2, b_2^{\mathcal{T}})$ of $\mathscr{F}$.
($\mathfrak{p}4$) No pattern of $\mathscr{F}$ is conform.
($\mathfrak{p}5$) Every forbidden pattern is biconnected.
($\mathfrak{p}6$) The representation $(\mathscr{F}, \mathcal{T})$ is automorphic.

We say that a connected representation for which properties $\mathfrak{p}1$ to $\mathfrak{p}6$ hold is a *normal representation*. In the process of reducing our representation to a normal representation, we will show that this can be done by an effective procedure.

4.1. Our first batch of reductions. Let $(\mathscr{F}, \mathcal{T})$ be a representation. We now define a number of operations on representations so that we might enforce certain properties. However, before we start, we wish our representation to have the following property:

($\mathfrak{p}1$) Any structure is valid if and only if it is weakly valid.

Let $\mathbf{H}\mathscr{F}$ be the set of homomorphic images of the patterns from $\mathscr{F}$, up to isomorphism. Recall that a forbidden pattern is a colored structure; hence, an homomorphic image of a forbidden pattern $(\mathcal{B}, b^{\mathcal{T}}) \in \mathscr{F}$ is a colored structure $(\mathcal{C}, c^{\mathcal{T}})$ such that there exists an epimorphism $\mathcal{B} \overset{h}{\twoheadrightarrow} \mathcal{C}$ with the properties that:

$$\begin{tikzcd} \mathcal{B} \arrow[r, two heads, "h"] \arrow[dr, "b^{\mathcal{T}}"'] & \mathcal{C} \arrow[d, "c^{\mathcal{T}}"] \\ & \mathcal{T} \end{tikzcd}$$

FIG. 3. *A commuting diagram.*

- for each symbol $R \in \sigma$ and for each tuple $R^{\mathcal{C}}(\tilde{\mathbf{t}})$, there exists a tuple $R^{\mathcal{B}}(\mathbf{t})$ such that $h(\mathbf{t}) = \tilde{\mathbf{t}}$ and
- the diagram in Figure 3 commutes.

LEMMA 14. *The representation* $(\mathbf{H}\mathscr{F}, \mathcal{T})$ *is equivalent to* $(\mathscr{F}, \mathcal{T})$.

Proof. Let $\mathcal{A}$ be valid w.r.t. $(\mathscr{F}, \mathcal{T})$, witnessed by $\mathcal{A}\xrightarrow{a^{\mathcal{T}}}\mathcal{T}$. Assume for contradiction that $(\mathcal{A}, a^{\mathcal{T}})$ is not valid w.r.t. $(\mathbf{H}\mathscr{F}, \mathcal{T})$, and let $(\mathcal{C}, c^{\mathcal{T}}) \in \mathbf{H}\mathscr{F}$ (defined from $(\mathcal{B}, b^{\mathcal{T}}) \in \mathscr{F}$, using h as above) be such that $(\mathcal{C}, c^{\mathcal{T}})\xrightarrow{f}(\mathcal{A}, a^{\mathcal{T}})$. By composition, it follows that $(\mathcal{B}, b^{\mathcal{T}})\xrightarrow{f\circ h}(\mathcal{A}, a^{\mathcal{T}})$. This yields a contradiction, and so $(\mathcal{A}, a^{\mathcal{T}})$ is valid w.r.t. $(\mathbf{H}\mathscr{F}, \mathcal{T})$.

Conversely, if $\mathcal{A}$ is valid w.r.t. $(\mathbf{H}\mathscr{F}, \mathcal{T})$, then $\mathcal{A}$ is valid w.r.t. $(\mathscr{F}, \mathcal{T})$ since $\mathscr{F} \subseteq \mathbf{H}\mathscr{F}$. ◻

LEMMA 15. *The representation* $(\mathbf{H}\mathscr{F}, \mathcal{T})$ *satisfies* $\mathfrak{p}1$.

Proof. Let $(\mathcal{A}, a^{\mathcal{T}})$ be weakly valid w.r.t. $(\mathbf{H}\mathscr{F}, \mathcal{T})$. Assume for contradiction that $(\mathcal{A}, a^{\mathcal{T}})$ is not valid w.r.t. $(\mathbf{H}\mathscr{F}, \mathcal{T})$, and let $(\mathcal{C}, c^{\mathcal{T}}) \in \mathbf{H}\mathscr{F}$ (defined from $(\mathcal{B}, b^{\mathcal{T}}) \in \mathscr{F}$, using h as above) be such that $(\mathcal{C}, c^{\mathcal{T}})\xrightarrow{f}(\mathcal{A}, a^{\mathcal{T}})$. By construction, $f(\mathcal{C}, c^{\mathcal{T}})$ belongs to $\mathbf{H}\mathscr{F}$, and $f(\mathcal{C}, c^{\mathcal{T}})\hookrightarrow(\mathcal{A}, a^{\mathcal{T}})$. This yields a contradiction.

Conversely, if $(\mathcal{A}, a^{\mathcal{T}})$ is valid w.r.t. $(\mathbf{H}\mathscr{F}, \mathcal{T})$, then it is trivially weakly valid. The result follows. ◻

Our next property to enforce is the following:

($\mathfrak{p}2$) Every pattern of $\mathscr{F}$ is automorphic.

DEFINITION 16. *Let* $(\mathscr{F}, \mathcal{T})$ *be a representation, and let* $(\mathscr{F}', \mathcal{T})$ *be the representation obtained by replacing a pattern of* $\mathscr{F}$ *with its core. We call this a* core reduction *on* $(\mathscr{F}, \mathcal{T})$.

Note that Definition 16 is well-defined by Lemma 4.

LEMMA 17. *Let the representation* $(\mathscr{F}', \mathcal{T})$ *be obtained from the representation* $(\mathscr{F}, \mathcal{T})$ *by a core reduction.*

- $(\mathscr{F}', \mathcal{T})$ *is equivalent to* $(\mathscr{F}, \mathcal{T})$.
- *If* $(\mathscr{F}, \mathcal{T})$ *satisfies property* $\mathfrak{p}1$*, then so does* $(\mathscr{F}', \mathcal{T})$.

Proof. If $(\mathcal{C}, c^{\mathcal{T}})$ is the core of $(\mathcal{B}, b^{\mathcal{T}})$, then $(\mathcal{B}, b^{\mathcal{T}})\to(\mathcal{A}, a^{\mathcal{T}})$ if and only if $(\mathcal{C}, c^{\mathcal{T}})\to(\mathcal{A}, a^{\mathcal{T}})$ for any structure $(\mathcal{A}, a^{\mathcal{T}})$. Hence, $(\mathscr{F}', \mathcal{T})$ is equivalent to $(\mathscr{F}, \mathcal{T})$.

Assume that $(\mathscr{F}, \mathcal{T})$ satisfies property $\mathfrak{p}1$. Suppose that $(\mathcal{A}, a^{\mathcal{T}})$ is weakly valid w.r.t. $(\mathscr{F}', \mathcal{T})$. If $(\mathcal{C}, c^{\mathcal{T}})\to(\mathcal{A}, a^{\mathcal{T}})$ for some $(\mathcal{C}, c^{\mathcal{T}}) \in \mathscr{F}'$, then we must have that $(\mathcal{B}, b^{\mathcal{T}})\to(\mathcal{A}, a^{\mathcal{T}})$ for some $(\mathcal{B}, b^{\mathcal{T}}) \in \mathscr{F}$. As $(\mathscr{F}, \mathcal{T})$ satisfies property $\mathfrak{p}1$, $(\mathcal{D}, d^{\mathcal{T}})\hookrightarrow(\mathcal{A}, a^{\mathcal{T}})$ for some $(\mathcal{D}, d^{\mathcal{T}}) \in \mathscr{F}$. If $(\mathcal{D}, d^{\mathcal{T}}) \in \mathscr{F}'$, then we obtain a contradiction; otherwise, the core of $(\mathcal{D}, d^{\mathcal{T}})$ is in $\mathscr{F}'$, and we still obtain that some forbidden pattern of $\mathscr{F}'$ embeds into $(\mathcal{A}, a^{\mathcal{T}})$, yielding a contradiction. Hence, $(\mathscr{F}', \mathcal{T})$ satisfies property $\mathfrak{p}1$. ◻

Our next property to enforce is the following:

($\mathfrak{p}3$) It is not the case that $(\mathcal{B}_1, b_1^{\mathcal{T}})\hookrightarrow(\mathcal{B}_2, b_2^{\mathcal{T}})$ for any distinct patterns $(\mathcal{B}_1, b_1^{\mathcal{T}})$ and $(\mathcal{B}_2, b_2^{\mathcal{T}})$ of $\mathscr{F}$.

Definition 18. *Let $(\mathscr{F},\mathcal{T})$ be a representation, and let $(\mathcal{B}_1,b_1^{\mathcal{T}})$ and $(\mathcal{B}_2,b_2^{\mathcal{T}})$ be distinct patterns of $\mathscr{F}$ such that $(\mathcal{B}_1,b_1^{\mathcal{T}})\hookrightarrow(\mathcal{B}_2,b_2^{\mathcal{T}})$. Let $(\mathscr{F}',\mathcal{T})$ be the representation obtained by removing the pattern $(\mathcal{B}_2,b_2^{\mathcal{T}})$ from $\mathscr{F}$. We call this an* embed reduction *on $(\mathscr{F},\mathcal{T})$.*

Lemma 19. *Let the representation $(\mathscr{F}',\mathcal{T})$ be obtained from the representation $(\mathscr{F},\mathcal{T})$ by an embed reduction.*

- *$(\mathscr{F}',\mathcal{T})$ is equivalent to $(\mathscr{F},\mathcal{T})$.*
- *If $(\mathscr{F},\mathcal{T})$ satisfies property $\mathfrak{p}1$, then so does $(\mathscr{F}',\mathcal{T})$.*

Proof. Trivially, $\mathrm{FPP}(\mathscr{F},\mathcal{T})\subseteq\mathrm{FPP}(\mathscr{F}',\mathcal{T})$. If $(\mathcal{B}_2,b_2^{\mathcal{T}})\to(\mathcal{A},a^{\mathcal{T}})$ (with reference to Definition 18), then $(\mathcal{B}_1,b_1^{\mathcal{T}})\to(\mathcal{A},a^{\mathcal{T}})$ for any structure $(\mathcal{A},a^{\mathcal{T}})$, and so $\mathrm{FPP}(\mathscr{F}',\mathcal{T})\subseteq\mathrm{FPP}(\mathscr{F},\mathcal{T})$.

Assume that $(\mathscr{F},\mathcal{T})$ satisfies property $\mathfrak{p}1$. Suppose that $(\mathcal{A},a^{\mathcal{T}})$ is weakly valid w.r.t. $(\mathscr{F}',\mathcal{T})$. If $(\mathcal{B},b^{\mathcal{T}})\to(\mathcal{A},a^{\mathcal{T}})$ for some pattern $(\mathcal{B},b^{\mathcal{T}})\in\mathscr{F}'$, and so some pattern in $\mathscr{F}$, then we have that $(\mathcal{A},a^{\mathcal{T}})$ is not weakly valid w.r.t. $(\mathscr{F},\mathcal{T})$. That is, $(\mathcal{C},c^{\mathcal{T}})\hookrightarrow(\mathcal{A},a^{\mathcal{T}})$ for some pattern $(\mathcal{C},c^{\mathcal{T}})\in\mathscr{F}$. If $(\mathcal{C},c^{\mathcal{T}})\in\mathscr{F}'$, then we obtain a contradiction; otherwise, $(\mathcal{C},c^{\mathcal{T}})$ is the pattern removed by the embed reduction and $(\mathcal{D},d^{\mathcal{T}})\hookrightarrow(\mathcal{C},c^{\mathcal{T}})$ for some pattern $(\mathcal{D},d^{\mathcal{T}})\in\mathscr{F}'$. Thus, we still obtain a contradiction, and $(\mathscr{F}',\mathcal{T})$ satisfies $\mathfrak{p}1$. □

Our next property to enforce is the following:

($\mathfrak{p}4$) No pattern of $\mathscr{F}$ is conform.

Definition 20. *Let $(\mathscr{F},\mathcal{T})$ be a representation, and let $(R(\mathbf{t}),\pi^{\mathcal{T}})$ be a conform pattern of $\mathscr{F}$. Let $\mathcal{T}'$ be the structure obtained from $\mathcal{T}$ by the removal of the tuple $R(\pi^{\mathcal{T}}(\mathbf{t}))$, and let e be the monomorphism $\mathcal{T}'\overset{e}{\hookrightarrow}\mathcal{T}$ defined by inclusion. Let $\mathscr{F}'$ denote the set of patterns of $\mathscr{F}$ that are also $\mathcal{T}'$-patterns; that is, the patterns $(\mathcal{B},b^{\mathcal{T}})\in\mathscr{F}$ for which $b^{\mathcal{T}}(\mathbf{u})\neq\pi^{\mathcal{T}}(\mathbf{t})$ for any tuple $R^{\mathcal{B}}(\mathbf{u})$. The representation $(\mathscr{F}',\mathcal{T}')$ has been obtained from $(\mathscr{F},\mathcal{T})$ by a* conform reduction.

Lemma 21. *Let the representation $(\mathscr{F}',\mathcal{T}')$ be obtained from the representation $(\mathscr{F},\mathcal{T})$ by a conform reduction.*

- *$(\mathscr{F}',\mathcal{T}')$ is equivalent to $(\mathscr{F},\mathcal{T})$.*
- *If $(\mathscr{F},\mathcal{T})$ satisfies property $\mathfrak{p}1$, then so does $(\mathscr{F}',\mathcal{T}')$.*

Proof. We denote a pattern $(\mathcal{B},b^{\mathcal{T}})\in\mathscr{F}$ that is also a $\mathcal{T}'$-pattern by $(\mathcal{B},b^{\mathcal{T}'})$ also, where $b^{\mathcal{T}'}$ is the homomorphism $\mathcal{B}\xrightarrow{b^{\mathcal{T}'}}\mathcal{T}'$ obtained directly from $b^{\mathcal{T}}$; that is, $b^{\mathcal{T}}=e\circ b^{\mathcal{T}'}$.

Assume that $(\mathcal{A},a^{\mathcal{T}'})$ is valid w.r.t. $(\mathscr{F}',\mathcal{T}')$ and define $a^{\mathcal{T}}:=e\circ a^{\mathcal{T}'}$ (so $\mathcal{A}\xrightarrow{a^{\mathcal{T}}}\mathcal{T}$ and $a^{\mathcal{T}}(\mathbf{u})\neq\pi^{\mathcal{T}}(\mathbf{t})$ for any tuple $R^{\mathcal{A}}(\mathbf{u})$). Suppose that some pattern $(\mathcal{B},b^{\mathcal{T}})\in\mathscr{F}$ is such that $(\mathcal{B},b^{\mathcal{T}})\to(\mathcal{A},a^{\mathcal{T}})$. Thus, $(\mathcal{B},b^{\mathcal{T}})$ is actually a $\mathcal{T}'$-pattern, and $(\mathcal{B},b^{\mathcal{T}'})\to(\mathcal{A},a^{\mathcal{T}'})$, which yields a contradiction.

Conversely, suppose that $(\mathcal{A},a^{\mathcal{T}})$ is valid w.r.t. $(\mathscr{F},\mathcal{T})$. There are two cases: either the map $a^{\mathcal{T}}$ yields a homomorphism $\mathcal{A}\to\mathcal{T}'$, or it doesn't.

Suppose that the map $a^{\mathcal{T}}$ yields a homomorphism $\mathcal{A}\xrightarrow{a^{\mathcal{T}'}}\mathcal{T}'$; thus, $a^{\mathcal{T}}=e\circ a^{\mathcal{T}'}$. If $(\mathcal{B},b^{\mathcal{T}'})\in\mathscr{F}'$ is such that $(\mathcal{B},b^{\mathcal{T}'})\to(\mathcal{A},a^{\mathcal{T}'})$, then we have that $(\mathcal{B},b^{\mathcal{T}})\to(\mathcal{A},a^{\mathcal{T}})$ (where $b^{\mathcal{T}}=e\circ b^{\mathcal{T}'}$, recall), which yields a contradiction. Thus, $(\mathcal{A},a^{\mathcal{T}'})$ is valid w.r.t. $(\mathscr{F}',\mathcal{T}')$.

Suppose that the map $a^{\mathcal{T}}$ does not yield a homomorphism from $\mathcal{A}$ to $\mathcal{T}'$. There must exist some tuple $R^{\mathcal{A}}(\hat{\mathbf{t}})$ such that $a^{\mathcal{T}}(\hat{\mathbf{t}})=\pi^{\mathcal{T}}(\mathbf{t})$. Define $h:|R(\mathbf{t})|\to|\mathcal{A}|$ as the map which takes $\mathbf{t}$ to $\hat{\mathbf{t}}$ (note that this is well-defined as $\mathbf{t}$ is antireflexive). Consequently, $(R(\mathbf{t}),\pi^{\mathcal{T}})\xrightarrow{h}(\mathcal{A},a^{\mathcal{T}})$, which yields a contradiction as $(\mathcal{A},a^{\mathcal{T}})$ is valid w.r.t. $(\mathscr{F},\mathcal{T})$. Hence, $(\mathscr{F}',\mathcal{T}')$ is equivalent to $(\mathscr{F},\mathcal{T})$.

Assume that $(\mathscr{F},\mathcal{T})$ satisfies property $\mathfrak{p}1$. Suppose that $(\mathcal{A},a^{\mathcal{T}'})$ is weakly valid w.r.t. $(\mathscr{F}',\mathcal{T}')$ and that there exists a pattern $(\mathcal{B},b^{\mathcal{T}'})\in\mathscr{F}'$ such that $(\mathcal{B},b^{\mathcal{T}'})\to(\mathcal{A},a^{\mathcal{T}'})$.

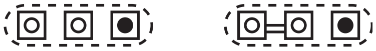


FIG. 4. *Depiction of tuples.*

Define $b^{\mathcal{T}} := e \circ b^{\mathcal{T}'}$ and $a^{\mathcal{T}} := e \circ a^{\mathcal{T}'}$. Thus, $(\mathcal{B}, b^{\mathcal{T}}) \in \mathscr{F}$ and $(\mathcal{B}, b^{\mathcal{T}}) \rightarrow (\mathcal{A}, a^{\mathcal{T}})$. Hence, $(\mathcal{A}, a^{\mathcal{T}})$ is not weakly valid w.r.t. $(\mathscr{F}, \mathcal{T})$. That is, $(\mathcal{C}, c^{\mathcal{T}}) \hookrightarrow (\mathcal{A}, a^{\mathcal{T}})$ for some pattern $(\mathcal{C}, c^{\mathcal{T}}) \in \mathscr{F}$. But as $a^{\mathcal{T}} = e \circ a^{\mathcal{T}'}$, so $(\mathcal{C}, c^{\mathcal{T}})$ is also a $\mathcal{T}'$-pattern and so is in $\mathscr{F}'$. This yields a contradiction, and the result follows. □

Remark 22. Applying embed reductions clearly preserves property $\mathfrak{p2}$. Note also that applying conform reductions preserves property $\mathfrak{p2}$. This follows directly from Lemma 2.

Example 23. Consider a representation $(\mathscr{F}, \mathcal{T})$ over the signature consisting of a binary relation symbol E and a ternary relation symbol R. The domain of $\mathcal{T}$ consists of two elements (or colors) $\circ$ and $\bullet$, with $E^{\mathcal{T}} = \{\circ, \bullet\}^2$ and $R^{\mathcal{T}} = \{\circ, \bullet\}^3$.

Consider the conform forbidden pattern consisting of the single tuple $R(x, y, z)$, where both x and y take the color $\circ$ and z takes the color $\bullet$. We depict this pattern by the left diagram in Figure 4. In the case where $x = y$, we depict the pattern by the right diagram in Figure 4.

The first (leftmost) column in Figure 5 depicts the four forbidden patterns in $\mathscr{F}$ (the top three are such that $R = \emptyset$, and the bottom is such that $E = \emptyset$). The second column depicts the representation $(\mathbf{H}\mathscr{F}, \mathcal{T})$, formed by adding all homomorphic images of the forbidden patterns in $\mathscr{F}$ (up to isomorphism). In the third column, we have performed core and embed reductions to obtain an equivalent representation satisfying properties $\mathfrak{p1}$, $\mathfrak{p2}$, and $\mathfrak{p3}$. In the fourth column, we have performed conform reductions to obtain an equivalent representation satisfying properties $\mathfrak{p1}$, $\mathfrak{p2}$, $\mathfrak{p3}$, and $\mathfrak{p4}$.

Note that, in general, starting from a representation satisfying property $\mathfrak{p1}$, if we apply core, embed, and conform reductions arbitrarily, then after a finite sequence of reductions, by the lemmas of this subsection, we will obtain an equivalent representation satisfying properties $\mathfrak{p1}$, $\mathfrak{p2}$, $\mathfrak{p3}$, and $\mathfrak{p4}$ (a simple induction suffices).

4.2. Feder–Vardi reductions. The reductions introduced so far do not suffice for us to obtain the normal form for which we are aiming. We need to interleave applications of these reductions with another reduction that we define now.

From now on, we make an important assumption regarding the representations we deal with: Until otherwise specified, we assume them to be connected (we shall deal with the disconnected case in section 6.1.1).

We say that a pattern is *biconnected* if its underlying structure is biconnected. Our aim is to enforce the following property (using techniques inspired from the proof of Theorem 8 in [15]):

($\mathfrak{p5}$) Every forbidden pattern is biconnected.

Before proceeding, we need some definitions relating to the grouping together of forbidden patterns. A *compact* $\mathcal{T}$*-structure* $\{\mathcal{A}, \overline{\alpha}\}$ is a structure $\mathcal{A}$ together with a map $\overline{\alpha} : A \to 2^T$ so that every map $a^{\mathcal{T}} : A \to T$ with the property that $a^{\mathcal{T}}(y) \in \overline{\alpha}(y)$, for all $y \in A$, yields a $\mathcal{T}$-colored structure $(\mathcal{A}, a^{\mathcal{T}})$. This notion is only a useful shorthand to denote a set of colored structures, as a compact structure can be *expanded* to obtain a set of colored structures, each with the same underlying structure; but we shall need this notion later on when we prove the termination of a particular sequence of transformations we employ towards the end of this section (this notion was not necessary in Feder and Vardi's original proof as the negated conjuncts correspond

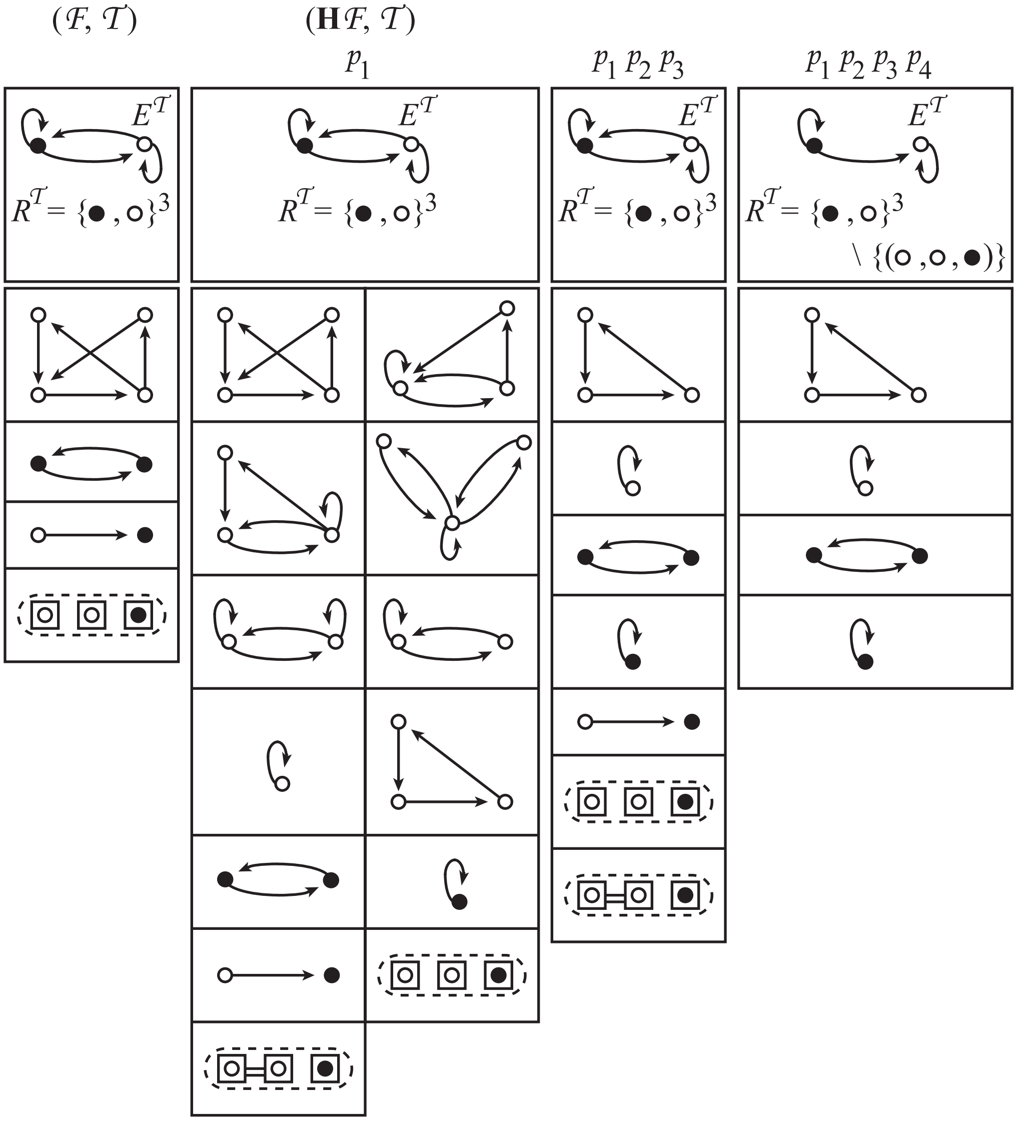


Fig. 5. *Towards a normal representation: step one.*

in general to *partially* colored structures; by choosing to work with *fully* colored structures in our combinatorial setting, this is the price we have to pay). Bearing this in mind, we can extend the definition of a representation to allow compact forbidden patterns and call it a *compact representation*, with the problem defined by a compact representation being the problem defined by the representation obtained by expanding all of the compact forbidden patterns.

Clearly, we may assume that every representation is compact, by replacing every forbidden pattern $(\mathcal{A}, a^{\mathcal{T}})$ by the compact forbidden pattern $\{\mathcal{A}, \overline{\alpha}\}$, where for every x in A, $\overline{\alpha}(x) := \{a^{\mathcal{T}}(x)\}$. We say that $(\mathcal{A}, a^{\mathcal{T}})$ is a forbidden pattern of the compact representation $(\mathscr{F}, \mathcal{T})$, and write $(\mathcal{A}, a^{\mathcal{T}}) \in \mathscr{F}$, if it is one of the forbidden patterns obtained by expanding one of the compact forbidden patterns. Notice that the notion

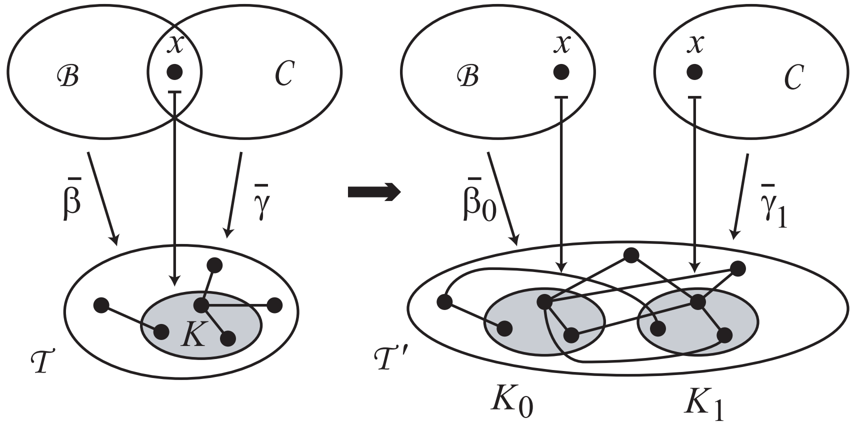


FIG. 6. *Feder–Vardi reduction.*

of a decomposition involves only the underlying structure; thus it generalizes to compact structures (of course, the definition of a decomposition in section 2 generalizes to colored structures).

DEFINITION 24 (Feder–Vardi reduction). *Let $(\mathscr{F}, \mathcal{T})$ be a compact representation with $\mathscr{F} = \mathscr{G} \cup \{\{\mathcal{B}, \overline{\beta}\}\overset{x}{32}\ \{\mathcal{C}, \overline{\gamma}\}\}$, and let $K = \overline{\beta}(x) = \overline{\gamma}(x)$. The new sets K_0 and K_1 are defined as $\{k_i : k \in K\}$ (that is, k_0 and k_1 are two new elements that stand as copies of k) for $i = 0, 1$, and we assume that K, K_0, and K_1 are mutually disjoint. Let $\mathcal{T}'$ be the structure obtained from $\mathcal{T}$ as follows:*

- *Replace K by K_0 and K_1 in $|\mathcal{T}|$.*
- *Set $R^{\mathcal{T}'}(\mathbf{t})$ whenever $R^{\mathcal{T}}(\tilde{\mathbf{t}})$, with $\mathbf{t}$ obtained from $\tilde{\mathbf{t}}$ by replacing every occurrence of an element $k \in K$ by either k_0 or k_1 (where two different occurrences of an element k might be replaced by k_0 and k_1; so, one tuple $R^{\mathcal{T}}(\tilde{\mathbf{t}})$ with i occurrences of elements of K gives rise to 2^i tuples $R^{\mathcal{T}'}(\mathbf{t})$).*

Let $\mathscr{F}'$ be the set of compact forbidden patterns induced from $\mathscr{F}$ as follows:

- *The compact forbidden pattern $\{\mathcal{B}, \overline{\beta}\}\overset{x}{32}\ \{\mathcal{C}, \overline{\gamma}\}$ is replaced by the two compact forbidden patterns induced from the decomposition so that*
  - *in the compact forbidden pattern $\{\mathcal{B}, \overline{\beta}_0\}$, $\overline{\beta}_0(x) = K_0$, and*
  - *in the compact forbidden pattern $\{\mathcal{C}, \overline{\gamma}_1\}$, $\overline{\gamma}_1(x) = K_1$.*
- *Every remaining occurrence of a color $k \in K$ in a compact forbidden pattern (including the two described above) is replaced by both k_0 and k_1; that is, every forbidden pattern obtained by expanding a compact forbidden pattern of $\mathscr{F}$ is replaced by a set of forbidden patterns, one for each possible assignment of k_0 and k_1 to occurrences of k.*

We call $(\mathscr{F}', \mathcal{T}')$ the Feder–Vardi reduction of $(\mathscr{F}, \mathcal{T})$ with respect to $\{\mathcal{B}, \overline{\beta}\}\overset{x}{32}\ \{\mathcal{C}, \overline{\gamma}\}$.

Part of a Feder–Vardi reduction can be visualized as in Figure 6. Note that if $(\mathscr{F}, \mathcal{T})$ is a connected representation, then a Feder–Vardi reduction of $(\mathscr{F}, \mathcal{T})$ is also connected.

We reiterate that working with compact forbidden patterns is just, to some extent, a notational convenience and that a Feder–Vardi reduction has the effect of "splitting" a set of forbidden patterns in one go.

We also need to define the essential notion of a recoloring. Intuitively, a recoloring is to a (compact) representation what a homomorphism is to a structure.

$$\begin{tikzcd}
 & & \mathcal{T}' \arrow[r, "r"] & \mathcal{T} \\
\mathcal{B} \arrow[rru, "b^{\mathcal{T}'}"] \arrow[r, "g"'] & \mathcal{A} \arrow[ru, "a^{\mathcal{T}'}"] \arrow[rru, "a^{\mathcal{T}}"'] \arrow[r, "f"'] & \mathcal{C} \arrow[u, "c^{\mathcal{T}'}"'] \arrow[ru, "c^{\mathcal{T}}"'] &
\end{tikzcd}$$

FIG. 7. *Proof of Proposition* 26.

DEFINITION 25 (Recoloring). *Let* $(\mathscr{F},\mathcal{T})$ *and* $(\mathscr{F}',\mathcal{T}')$ *be two compact representations. A* recoloring r of $(\mathscr{F}',\mathcal{T}')$ to $(\mathscr{F},\mathcal{T})$ *is a homomorphism* $\mathcal{T}'\xrightarrow{r}\mathcal{T}$ *such that any* inverse image $(\mathcal{A},a^{\mathcal{T}'})$ *of a forbidden pattern* $(\mathcal{A},a^{\mathcal{T}})$ *of* $\mathscr{F}$ *is not valid w.r.t.* $(\mathscr{F}',\mathcal{T}')$*, where by "inverse image" we mean that* $r\circ a^{\mathcal{T}'}=a^{\mathcal{T}}$*. We denote the fact that* r *is a recoloring by* $(\mathscr{F}',\mathcal{T}')\xrightarrow{r}(\mathscr{F},\mathcal{T})$ *(we use the same notation as for homomorphisms without causing any confusion). If, furthermore,* r *is onto (respectively, one-to-one), then* r *is an* epirecoloring *(respectively,* monorecoloring*). If* $(\mathscr{F}',\mathcal{T}')\overset{r}{\lhook\joinrel\twoheadrightarrow}(\mathscr{F},\mathcal{T})$ *and* $(\mathscr{F},\mathcal{T})\xrightarrow{r^{-1}}(\mathscr{F}',\mathcal{T}')$*, then* r *is an* isorecoloring*, and we write* $(\mathscr{F},\mathcal{T})\approx(\mathscr{F}',\mathcal{T}')$*.*

The fact that for CSP, $\mathrm{CSP}(\mathcal{A})\subseteq\mathrm{CSP}(\mathcal{B})$ whenever $\mathcal{A}\to\mathcal{B}$, generalizes to FPP.

PROPOSITION 26. *Let* $(\mathscr{F},\mathcal{T})$ *and* $(\mathscr{F}',\mathcal{T}')$ *be two compact representations. If* $(\mathscr{F}',\mathcal{T}')\to(\mathscr{F},\mathcal{T})$*, then* $\mathrm{FPP}(\mathscr{F}',\mathcal{T}')\subseteq\mathrm{FPP}(\mathscr{F},\mathcal{T})$*.*

Proof. Let $(\mathscr{F}',\mathcal{T}')\xrightarrow{r}(\mathscr{F},\mathcal{T})$, and let $\mathcal{C}$ be a structure that is not valid w.r.t. $(\mathscr{F},\mathcal{T})$. If $\mathcal{C}\nrightarrow\mathcal{T}'$, then $\mathcal{C}$ is not valid w.r.t. $(\mathscr{F}',\mathcal{T}')$, and we are done. Thus, let $\mathcal{C}\xrightarrow{c^{\mathcal{T}'}}\mathcal{T}'$ and define $c^{\mathcal{T}}:=r\circ c^{\mathcal{T}'}$. By assumption, there exists a forbidden pattern $(\mathcal{A},a^{\mathcal{T}})\in\mathscr{F}$ such that $(\mathcal{A},a^{\mathcal{T}})\xrightarrow{f}(\mathcal{C},c^{\mathcal{T}})$; so define $a^{\mathcal{T}'}:=c^{\mathcal{T}'}\circ f$, with the result that $a^{\mathcal{T}}=r\circ a^{\mathcal{T}'}$ (see Figure 7). Since r is a recoloring, there exists a forbidden pattern $(\mathcal{B},b^{\mathcal{T}'})\in\mathscr{F}'$ such that $(\mathcal{B},b^{\mathcal{T}'})\xrightarrow{g}(\mathcal{A},a^{\mathcal{T}'})$. This can be summarized by the commuting diagram of Figure 7.

Hence, we can see that $(\mathcal{B},b^{\mathcal{T}'})\xrightarrow{f\circ g}(\mathcal{C},c^{\mathcal{T}'})$, which proves that $(\mathcal{C},c^{\mathcal{T}'})$ is not valid w.r.t. $(\mathscr{F}',\mathcal{T}')$, and we are done. □

PROPOSITION 27. *Let* $(\mathscr{F}',\mathcal{T}')$ *be obtained from* $(\mathscr{F},\mathcal{T})$ *via a Feder–Vardi reduction, as in Definition* 24. Then $(\mathscr{F}',\mathcal{T}')$ and $(\mathscr{F},\mathcal{T})$ are equivalent.

Proof. Let $\mathcal{T}'\xrightarrow{r}\mathcal{T}$ be the homomorphism that identifies $k_i\in K_i$ for $i=0,1$, with $k\in K$, and leaves all other elements fixed. We begin by proving that r is a recoloring.

By construction, the inverse images of any forbidden pattern of $\mathscr{G}$ belong to $\mathscr{F}'$. So, it remains to check the inverse images of the patterns expanded from the compact forbidden pattern $\{\mathcal{B},\overline{\beta}\}\overset{x}{32}\ \{\mathcal{C},\overline{\gamma}\}$. Assume without loss of generality (w.l.o.g.) that we are checking an inverse image where x takes a color from K_0. Consider the substructure of the inverse image induced by B. By construction, this substructure is one of the patterns expanded from the compact forbidden pattern $\{\mathcal{B},\overline{\beta}_0\}$ (constructed as in Definition 24), which is a compact forbidden pattern of $\mathscr{F}'$. The case when x takes a color from K_1 is similar. Hence, r is a recoloring. By Proposition 26, $\mathrm{FPP}(\mathscr{F}',\mathcal{T}')\subseteq\mathrm{FPP}(\mathscr{F},\mathcal{T})$.

Conversely, suppose that $(\mathcal{A},a^{\mathcal{T}})$ is valid w.r.t. $(\mathscr{F},\mathcal{T})$. We construct a coloring $a^{\mathcal{T}'}$ from $a^{\mathcal{T}}$ as follows:

- For any $y\in A$ such that $a^{\mathcal{T}}(y)\notin K$, set $a^{\mathcal{T}'}(y)=a^{\mathcal{T}}(y)$;
- Suppose that $a^{\mathcal{T}}(y)=k\in K$. As $(\mathcal{A},a^{\mathcal{T}})$ is valid w.r.t. $(\mathscr{F},\mathcal{T})$, there does not exist a homomorphism from any forbidden pattern expanded from the

$$\begin{tikzcd}
 & \mathcal{T}' \arrow[r, "r"] & \mathcal{T} \\
\mathcal{D} \arrow[ur, "d^{\mathcal{T}'}"] \arrow[urr, "d^{\mathcal{T}}"'] \arrow[r, "h"'] & \mathcal{A} \arrow[u, "a^{\mathcal{T}'}"'] \arrow[ur, "a^{\mathcal{T}}"']
\end{tikzcd}$$

FIG. 8. *The first case.*

$$\begin{tikzcd}
 & \mathcal{T}' \arrow[r, "r"] & \mathcal{T} \\
\mathcal{B} \arrow[ur, "b^{\mathcal{T}'}"] \arrow[urr, "b^{\mathcal{T}}"'] \arrow[r, "h"'] & \mathcal{A} \arrow[u, "a^{\mathcal{T}'}"'] \arrow[ur, "a^{\mathcal{T}}"']
\end{tikzcd}$$

FIG. 9. *The second case.*

compact forbidden pattern $\{\mathcal{B},\overline{\beta}\}\overset{x}{32}\ \{\mathcal{C},\overline{\gamma}\}$ to $(\mathcal{A},a^{\mathcal{T}})$. That is, there does not exist $(\mathcal{B},b^{\mathcal{T}})\in\{\mathcal{B},\overline{\beta}_0\}$ and $(\mathcal{C},c^{\mathcal{T}})\in\{\mathcal{C},\overline{\gamma}_1\}$ such that both $(\mathcal{B},b^{\mathcal{T}})\xrightarrow{h_{\mathcal{B}}}(\mathcal{A},a^{\mathcal{T}})$ and $(\mathcal{C},c^{\mathcal{T}})\xrightarrow{h_{\mathcal{C}}}(\mathcal{A},a^{\mathcal{T}})$, with $h_{\mathcal{B}}(x)=h_{\mathcal{C}}(x)=y$. Thus

- if there exists $(\mathcal{B},b^{\mathcal{T}})\in\{\mathcal{B},\overline{\beta}_0\}$ such that $(\mathcal{B},b^{\mathcal{T}})\xrightarrow{h_{\mathcal{B}}}(\mathcal{A},a^{\mathcal{T}})$, with $h_{\mathcal{B}}(x)=y$, then set $a^{\mathcal{T}'}(y)=k_1$;
- otherwise, set $a^{\mathcal{T}'}(y)=k_0$.

Suppose that $(\mathcal{D},d^{\mathcal{T}'})\xrightarrow{h}(\mathcal{A},a^{\mathcal{T}'})$, where $(\mathcal{D},d^{\mathcal{T}'})$ is derived from some forbidden pattern $(\mathcal{D},d^{\mathcal{T}})$ of (some compact forbidden pattern of) $\mathscr{G}$ (according to the Feder–Vardi reduction). Thus, we have the commutative diagram of Figure 8. This yields a contradiction as $(\mathcal{A},a^{\mathcal{T}})$ is valid w.r.t. $(\mathscr{F},\mathcal{T})$.

Suppose that $(\mathcal{B},b^{\mathcal{T}'})\xrightarrow{h}(\mathcal{A},a^{\mathcal{T}'})$, where $(\mathcal{B},b^{\mathcal{T}'})$ is derived from the compact forbidden pattern $\{\mathcal{B},\overline{\beta}\}\overset{x}{32}\ \{\mathcal{C},\overline{\gamma}\}$ (according to the Feder–Vardi reduction). Thus, we have the commutative diagram of Figure 9. In particular, $(\mathcal{B},b^{\mathcal{T}})\xrightarrow{h}(\mathcal{A},a^{\mathcal{T}})$, where $(\mathcal{B},b^{\mathcal{T}})\in\{\mathcal{B},\overline{\beta}_0\}$. Set $h(x)=y$. By definition of $a^{\mathcal{T}'}$, $a^{\mathcal{T}'}\circ h(x)\in K_1$. However, by definition of $\{\mathcal{B},\overline{\beta}_0\}$, $b^{\mathcal{T}'}(x)\in K_0$. The fact that $b^{\mathcal{T}'}=a^{\mathcal{T}'}\circ h$ yields a contradiction.

Suppose that it is not the case that $(\mathcal{B},b^{\mathcal{T}'})\xrightarrow{h}(\mathcal{A},a^{\mathcal{T}'})$ for any $(\mathcal{B},b^{\mathcal{T}'})$ derived from the compact forbidden pattern $\{\mathcal{B},\overline{\beta}\}\overset{x}{32}\ \{\mathcal{C},\overline{\gamma}\}$ (according to the Feder–Vardi reduction) but that $(\mathcal{C},c^{\mathcal{T}'})\xrightarrow{h}(\mathcal{A},a^{\mathcal{T}'})$ for some $(\mathcal{C},c^{\mathcal{T}'})$ derived from the compact forbidden pattern $\{\mathcal{B},\overline{\beta}\}\overset{x}{32}\ \{\mathcal{C},\overline{\gamma}\}$. A contradiction follows by reasoning analogously to the preceding case. Hence, we have that $\mathrm{FPP}(\mathscr{F},\mathcal{T})\subseteq\mathrm{FPP}(\mathscr{F}',\mathcal{T}')$. □

PROPOSITION 28. *Let $(\mathscr{F}',\mathcal{T}')$ be obtained from $(\mathscr{F},\mathcal{T})$ via a Feder–Vardi reduction, as in Definition* 24. *If property* $\mathtt{p1}$ *holds for $(\mathscr{F},\mathcal{T})$, then it holds for $(\mathscr{F}',\mathcal{T}')$.*

Proof. Define $\mathcal{T}'\xrightarrow{r}\mathcal{T}$ to be the homomorphism that identifies $k_i\in K_i$ for $i=0,1$, with $k\in K$, and leaves all other elements fixed. Let $\mathcal{A}$ be nonvalid w.r.t. $(\mathscr{F}',\mathcal{T}')$. Since $(\mathscr{F},\mathcal{T})$ is equivalent to $(\mathscr{F}',\mathcal{T}')$, by Proposition 27, it follows that $\mathcal{A}$ is nonvalid w.r.t. $(\mathscr{F},\mathcal{T})$. We may assume that $\mathcal{A}\to\mathcal{T}'$. Let $\mathcal{A}\xrightarrow{a^{\mathcal{T}'}}\mathcal{T}'$ and define $a^{\mathcal{T}}:=r\circ a^{\mathcal{T}'}$. As $\mathtt{p1}$ holds for $(\mathscr{F},\mathcal{T})$, there exists $(\mathcal{D},d^{\mathcal{T}})\in\mathscr{F}$ such that $(\mathcal{D},d^{\mathcal{T}})\overset{f}{\hookrightarrow}(\mathcal{A},a^{\mathcal{T}})$. In particular,

$r \circ a^{\mathcal{T}'} \circ f = d^{\mathcal{T}}$; so, $r \circ d^{\mathcal{T}'} = d^{\mathcal{T}}$ when we define $d^{\mathcal{T}'} := a^{\mathcal{T}'} \circ f$. If $(\mathcal{D}, d^{\mathcal{T}})$ is a pattern of $\mathscr{G}$, then $(\mathcal{D}, d^{\mathcal{T}'}) \in \mathscr{F}'$. If $(\mathcal{D}, d^{\mathcal{T}})$ is a pattern of $\{\mathcal{B}, \overline{\beta}\}\overset{x}{32}\ \{\mathcal{C}, \overline{\gamma}\}$, then a pattern of either $\{\mathcal{B}, \overline{\beta}_0\}$ or $\{\mathcal{C}, \overline{\gamma}_1\}$ (where these are the compact forbidden $\mathcal{T}'$-patterns as constructed in Definition 24) is a (colored) substructure of $(\mathcal{D}, d^{\mathcal{T}'})$. Hence, regardless, there exists $(\mathcal{E}, e^{\mathcal{T}'}) \in \mathscr{F}'$ such that $(\mathcal{E}, e^{\mathcal{T}'}) \overset{g}{\hookrightarrow} (\mathcal{D}, d^{\mathcal{T}'})$. As $d^{\mathcal{T}'} = a^{\mathcal{T}'} \circ f$, we have that $(\mathcal{D}, d^{\mathcal{T}'}) \overset{f}{\hookrightarrow} (\mathcal{A}, a^{\mathcal{T}'})$, and so $(\mathcal{E}, e^{\mathcal{T}'}) \overset{f \circ g}{\hookrightarrow} (\mathcal{A}, a^{\mathcal{T}'})$. Consequently, if $\mathcal{A}$ is weakly valid w.r.t. $(\mathscr{F}', \mathcal{T}')$, then it is valid w.r.t. $(\mathscr{F}', \mathcal{T}')$. □

We define the *rank* of a (connected) compact structure to be the number of applications of the operator 32 in order that all resulting compact structures are biconnected. We associate with a compact representation a *rank polynomial* $P(X) = \Sigma_i a_i X^i$, where a_i is the number of compact forbidden patterns of rank i. Let $(\mathscr{F}', \mathcal{T}')$ be obtained from $(\mathscr{F}, \mathcal{T})$ via a Feder–Vardi reduction, with P the rank polynomial of $(\mathscr{F}, \mathcal{T})$ and P' that of $(\mathscr{F}', \mathcal{T}')$. It is easy to check that $P' < P$, where $<$ denotes the standard well-ordering of polynomials. Consequently, any sequence of Feder–Vardi reductions is necessarily finite. It is in order to prove this finiteness that we consider compact representations; given that we now that any sequence of Feder–Vardi reductions is necessarily finite, we can now revert to dealing with standard, as opposed to compact, representations. Of course, all of the results in this section mentioning compact representations also hold for standard representations.

4.3. Enforcing $\mathfrak{p}1$ to $\mathfrak{p}5$. We now use the reductions developed so far to obtain, from any connected representation, an equivalent representation satisfying properties $\mathfrak{p}1$, $\mathfrak{p}2$, $\mathfrak{p}3$, $\mathfrak{p}4$, and $\mathfrak{p}5$. We remind the reader that we are still assuming all representations to be connected, and we note that all reductions so far defined preserve the property of a representation being connected.

Definition 29. *Let $(\mathscr{F}, \mathcal{T})$ be a representation where every forbidden pattern of $\mathscr{F}$ is automorphic and nonconform. Define*

$$\rho(\mathscr{F}, \mathcal{T}) = \max\{||(\mathcal{B}, b^{\mathcal{T}})|| : (\mathcal{B}, b^{\mathcal{T}}) \textit{ is a forbidden pattern of } \mathscr{F} \\ \textit{that is not biconnected}\},$$

where $||(\mathcal{B}, b^{\mathcal{T}})||$ is the number of tuples in $\mathcal{B}$; that is, the sum of the numbers of $\{||R^{\mathcal{B}}|| : R$ is a relation symbol of the underlying signature$\}$, where $||R^{\mathcal{B}}||$ is the number of tuples in the relation $R^{\mathcal{B}}$.

Consider the following process, starting with a (connected) representation $(\mathscr{F}, \mathcal{T})$. Replace $(\mathscr{F}, \mathcal{T})$ with the representation $(\mathbf{H}\mathscr{F}, \mathcal{T})$, and so, by Lemmas 14 and 15, $(\mathbf{H}\mathscr{F}, \mathcal{T})$ is equivalent to $(\mathscr{F}, \mathcal{T})$ and satisfies $\mathfrak{p}1$. Perform a maximal sequence of core, embed, and conform reductions and denote the resulting representation by $(\mathscr{F}_1, \mathcal{T}_1)$. In particular, every forbidden pattern of $\mathscr{F}_1$ is a core and nonconform, and so $\rho(\mathscr{F}_1, \mathcal{T}_1)$ is well-defined. If $\rho(\mathscr{F}_1, \mathcal{T}_1) = 0$, then halt.

Otherwise, perform a maximal sequence of Feder–Vardi reductions, followed by a maximal sequence of core, embed, and conform reductions. Denote the resulting representation by $(\mathscr{F}_2, \mathcal{T}_2)$. In particular, every forbidden pattern of $\mathscr{F}_2$ is a core and nonconform, and so $\rho(\mathscr{F}_2, \mathcal{T}_2)$ is well-defined. Also, the sequence of reductions performed in order to obtain $(\mathscr{F}_2, \mathcal{T}_2)$ from $(\mathscr{F}_1, \mathcal{T}_1)$ is such that:

- every forbidden pattern of $\mathscr{F}_1$ that is a biconnected ($\mathcal{T}_1$-colored) core gives rise to forbidden patterns of $\mathscr{F}_2$ that are also biconnected ($\mathcal{T}_2$-colored) cores (see Remark 22); and,
- any non-biconnected core of $\mathscr{F}_1$ is split into forbidden patterns, each of which has strictly less tuples than the original non-biconnected core of $\mathscr{F}_1$.

That is, $\rho(\mathscr{F}_2, \mathcal{T}_2) < \rho(\mathscr{F}_1, \mathcal{T}_1)$.

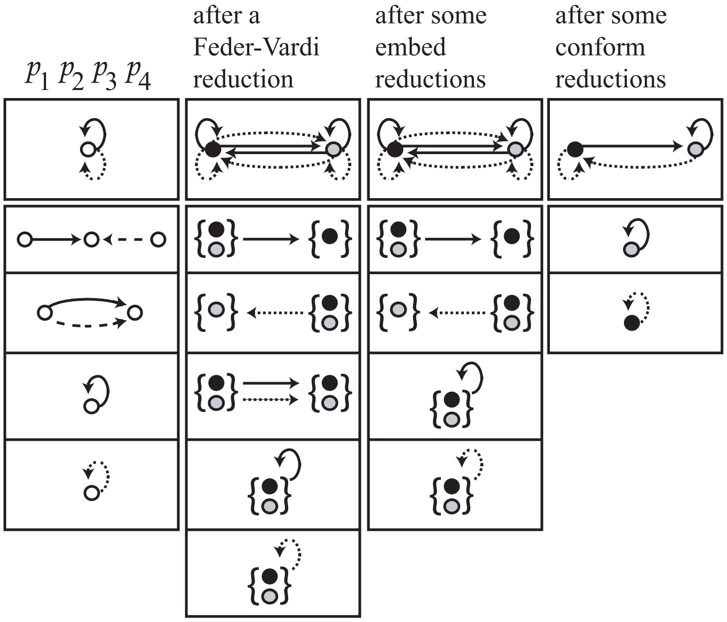

p_1 p_2 p_3 p_4
after a Feder-Vardi reduction
after some embed reductions
after some conform reductions

FIG. 10. *Applying our reductions.*

By iterating this process, we eventually obtain a connected representation $(\mathscr{F}',\mathcal{T}')$ that is equivalent to $(\mathscr{F},\mathcal{T})$ and satisfies properties $\mathfrak{p}1$, $\mathfrak{p}2$, $\mathfrak{p}3$, $\mathfrak{p}4$, and $\mathfrak{p}5$.

Example 30. Consider a representation $(\mathscr{F},\mathcal{T})$ over the signature consisting of two binary relation symbols E and F, where $\mathcal{T}$ and the forbidden patterns of $\mathscr{F}$ are as in the first column of Figure 10 (we represent "E-edges" by solid arrowed lines and "F-edges" by dashed arrowed lines).

As can be seen, $(\mathscr{F},\mathcal{T})$ satisfies properties $\mathfrak{p}1$, $\mathfrak{p}2$, $\mathfrak{p}3$, and $\mathfrak{p}4$. However, one forbidden pattern is not biconnected, and so we perform a Feder–Vardi reduction so that the resulting compact forbidden pattern is as depicted in the second column of Figure 10. This messes up the aforementioned properties, and so we perform some embed reductions to obtain the compact representation in the third column of Figure 10 (we have left the depiction of this representation in its compact form so that the figure does not become cluttered). Finally, we perform some conform reductions to obtain the representation in the fourth column of Figure 10 which is equivalent to the original one and satisfies properties $\mathfrak{p}1$, $\mathfrak{p}2$, $\mathfrak{p}3$, $\mathfrak{p}4$, and $\mathfrak{p}5$.

4.4. Enforcing $\mathfrak{p}1$–$\mathfrak{p}6$. Given our notion of a recoloring of a representation, we can define a retract of a representation as follows.

DEFINITION 31. *A representation $(\mathscr{F}',\mathcal{T}')$ is a* retract *of the representation $(\mathscr{F},\mathcal{T})$ if there exists a monorecoloring $(\mathscr{F}',\mathcal{T}')\overset{r}{\hookrightarrow}(\mathscr{F},\mathcal{T})$ and an epirecoloring $(\mathscr{F},\mathcal{T})\overset{s}{\twoheadrightarrow}(\mathscr{F}',\mathcal{T}')$ such that $s\circ r = id_{\mathcal{T}'}$. We call a representation $(\mathscr{F},\mathcal{T})$* automorphic *if whenever $(\mathscr{F}',\mathcal{T}')$ is a retract of $(\mathscr{F},\mathcal{T})$, then $(\mathscr{F}',\mathcal{T}') \approx (\mathscr{F},\mathcal{T})$.*

It is not difficult to see that given any representation $(\mathscr{F},\mathcal{T})$, there is an automorphic representation $(\mathscr{F}',\mathcal{T}')$ that is a retract of $(\mathscr{F},\mathcal{T})$ (and thus defines the same forbidden patterns problem by Proposition 26). We remark that the notion of a "core" for representations does not possess the properties that it does in the case of (colored) structures; e.g., it is not unique up to isorecoloring, but we resist the temptation to go into further details here as this has no consequence on what follows.

$$\begin{tikzcd}
\mathcal{C} \arrow[r, "g"] \arrow[dr, "c^{\mathcal{T}}"'] & \mathcal{B} \arrow[r, "f"] \arrow[d, "a_1^{\mathcal{T}} \circ f"'] \arrow[dr, "b^{\mathcal{T}'}"] & \mathcal{A} \arrow[dl, "a_1^{\mathcal{T}}"'] \arrow[d, "a^{\mathcal{T}'}"] \arrow[dr, "a_2^{\mathcal{T}}"] & \\
 & \mathcal{T} \arrow[r, two heads, "s"'] & \mathcal{T}' \arrow[r, hook, "r"'] & \mathcal{T}
\end{tikzcd}$$

FIG. 11. *A commutative diagram.*

$$\begin{tikzcd}
\mathcal{C} \arrow[r, "h_2"] \arrow[d, "c^{\mathcal{T}_1}"'] & \mathcal{B} \arrow[r, "h_1"] \arrow[dl, "a^{\mathcal{T}_1} \circ h_1"'] \arrow[d, "b^{\mathcal{T}_2}"] & \mathcal{A} \arrow[dll, "a^{\mathcal{T}_1}"'] \arrow[dl, "f \circ a^{\mathcal{T}_1}"] \arrow[d, "a^{\mathcal{T}_3}"] \\
\mathcal{T}_1 \arrow[r, "f"'] & \mathcal{T}_2 \arrow[r, "g"'] & \mathcal{T}_3
\end{tikzcd}$$

FIG. 12. *Proving transitivity.*

The next property we wish to enforce is as follows:

$(\mathfrak{p}6)$ The representation $(\mathscr{F}, \mathcal{T})$ is automorphic.

Suppose that $(\mathscr{F}, \mathcal{T})$ is not automorphic and that $(\mathscr{F}', \mathcal{T}') \xhookrightarrow{r} (\mathscr{F}, \mathcal{T})$, $(\mathscr{F}, \mathcal{T}) \overset{s}{\twoheadrightarrow} (\mathscr{F}', \mathcal{T}')$, and $s \circ r = id_{\mathcal{T}'}$, with $(\mathscr{F}', \mathcal{T}')$ automorphic. Define

$$\mathscr{F}'' = \{(\mathcal{A}, a^{\mathcal{T}'}) : (\mathcal{A}, a^{\mathcal{T}}) \in \mathscr{F} \text{ and } a^{\mathcal{T}} = r \circ a^{\mathcal{T}'}\}.$$

Let $(\mathcal{A}, a^{\mathcal{T}}) \in \mathscr{F}$, and let $a^{\mathcal{T}} = r \circ a^{\mathcal{T}'}$. By construction, $(\mathcal{A}, a^{\mathcal{T}'}) \in \mathscr{F}''$ and as such is not valid w.r.t. $(\mathscr{F}'', \mathcal{T}')$. Thus, $(\mathscr{F}'', \mathcal{T}') \xhookrightarrow{r} (\mathscr{F}, \mathcal{T})$. However, we also want to show that $(\mathscr{F}, \mathcal{T}) \overset{s}{\twoheadrightarrow} (\mathscr{F}'', \mathcal{T}')$.

Let $(\mathcal{A}, a^{\mathcal{T}'}) \in \mathscr{F}''$, and let $(\mathcal{A}, a_1^{\mathcal{T}})$ be such that $s \circ a_1^{\mathcal{T}} = a^{\mathcal{T}'}$; i.e., $(\mathcal{A}, a^{\mathcal{T}})$ is an inverse image of $(\mathcal{A}, a^{\mathcal{T}'})$ via s. Also, because $(\mathcal{A}, a^{\mathcal{T}'}) \in \mathscr{F}''$, by definition there exists $(\mathcal{A}, a_2^{\mathcal{T}}) \in \mathscr{F}$ such that $a_2^{\mathcal{T}} = r \circ a^{\mathcal{T}'}$. As $(\mathscr{F}', \mathcal{T}') \xhookrightarrow{r} (\mathscr{F}, \mathcal{T})$, there exists $(\mathcal{B}, b^{\mathcal{T}'}) \in \mathscr{F}'$ such that $(\mathcal{B}, b^{\mathcal{T}'}) \xrightarrow{f} (\mathcal{A}, a^{\mathcal{T}'})$. Hence, $(\mathcal{B}, a_1^{\mathcal{T}} \circ f)$ is an inverse image of $(\mathcal{B}, b^{\mathcal{T}'})$ via s, and so there exists $(\mathcal{C}, c^{\mathcal{T}}) \in \mathscr{F}$ such that $(\mathcal{C}, c^{\mathcal{T}}) \xrightarrow{g} (\mathcal{B}, a_1^{\mathcal{T}} \circ f)$ (see Figure 11). Thus, $(\mathcal{C}, c^{\mathcal{T}}) \xrightarrow{f \circ g} (\mathcal{A}, a_1^{\mathcal{T}})$ and $(\mathscr{F}, \mathcal{T}) \overset{s}{\twoheadrightarrow} (\mathscr{F}'', \mathcal{T}')$. In particular, $(\mathscr{F}'', \mathcal{T}')$ is a retract of $(\mathscr{F}, \mathcal{T})$.

We need the notion of a recoloring to be transitive.

LEMMA 32. *If $(\mathscr{F}_1, \mathcal{T}_1) \xrightarrow{f} (\mathscr{F}_2, \mathcal{T}_2)$ and $(\mathscr{F}_2, \mathcal{T}_2) \xrightarrow{g} (\mathscr{F}_3, \mathcal{T}_3)$ are recolorings, then $(\mathscr{F}_1, \mathcal{T}_1) \xrightarrow{g \circ f} (\mathscr{F}_3, \mathcal{T}_3)$ is a recoloring.*

Proof. Let $(\mathcal{A}, a^{\mathcal{T}_3}) \in \mathscr{F}_3$, and let $(\mathcal{A}, a^{\mathcal{T}_1}) \xrightarrow{g \circ f} (\mathcal{A}, a^{\mathcal{T}_3})$. As g is a recoloring of $(\mathscr{F}_2, \mathcal{T}_2)$ to $(\mathscr{F}_3, \mathcal{T}_3)$, there exists a forbidden pattern $(\mathcal{B}, b^{\mathcal{T}_2}) \in \mathscr{F}_2$ for which $(\mathcal{B}, b^{\mathcal{T}_2}) \overset{h_1}{\not\rightarrow} (\mathcal{A}, f \circ a^{\mathcal{T}_1})$. As f is a recoloring of $(\mathscr{F}_1, \mathcal{T}_1)$ to $(\mathscr{F}_2, \mathcal{T}_2)$, there exists a forbidden pattern $(\mathcal{C}, c^{\mathcal{T}_1}) \in \mathscr{F}_1$ for which $(\mathcal{C}, c^{\mathcal{T}_1}) \xrightarrow{h_2} (\mathcal{B}, a^{\mathcal{T}_1} \circ h_1)$. The situation can be depicted as in Figure 12. Consequently, $(\mathcal{C}, c^{\mathcal{T}_1}) \xrightarrow{h_1 \circ h_2} (\mathcal{A}, a^{\mathcal{T}_1})$, and $(\mathscr{F}_1, \mathcal{T}_1) \xrightarrow{g \circ f} (\mathscr{F}_3, \mathcal{T}_3)$. ◻

We have that $(\mathscr{F}'', \mathcal{T}') \xhookrightarrow{r} (\mathscr{F}, \mathcal{T})$ and $(\mathscr{F}, \mathcal{T}) \overset{s}{\twoheadrightarrow} (\mathscr{F}', \mathcal{T}')$, and consequently by Lemma 32, the identity map on $\mathcal{T}'$ is an isorecoloring from $(\mathscr{F}'', \mathcal{T}')$ to $(\mathscr{F}', \mathcal{T}')$. Thus, $(\mathscr{F}'', \mathcal{T}')$ is automorphic.

$$\begin{tikzcd}
\mathcal{B} \arrow[r, hook, "f"] \arrow[dr, "a^{\mathcal{T}'}\circ f"'] \arrow[drr, "b^{\mathcal{T}}"] & \mathcal{A} \arrow[d, "a^{\mathcal{T}'}"] \arrow[dr, "r\circ a^{\mathcal{T}'}"] & \\
 & \mathcal{T}' \arrow[r, hook, "r"'] & \mathcal{T}
\end{tikzcd}$$

Fig. 13. *Verifying property* $\mathfrak{p}1$.

We now need to affirm the properties $\mathfrak{p}1$, $\mathfrak{p}2$, $\mathfrak{p}3$, $\mathfrak{p}4$, and $\mathfrak{p}5$ for $(\mathscr{F}'', \mathcal{T}')$; we deal with $\mathfrak{p}1$ first (note that if $(\mathscr{F}, \mathcal{T})$ is connected, then so is $(\mathscr{F}'', \mathcal{T}')$). Assume that $\mathcal{A}$ is not valid w.r.t. $(\mathscr{F}'', \mathcal{T}')$. Consequently, by Proposition 26, $\mathcal{A}$ is not valid w.r.t. $(\mathscr{F}, \mathcal{T})$. Suppose that $\mathcal{A} \xrightarrow{a^{\mathcal{T}'}} \mathcal{T}'$. Thus, $\mathcal{A} \xrightarrow{r\circ a^{\mathcal{T}'}} \mathcal{T}$. Hence, there exists $(\mathcal{B}, b^{\mathcal{T}}) \in \mathscr{F}$ such that $(\mathcal{B}, b^{\mathcal{T}}) \overset{f}{\hookrightarrow} (\mathcal{A}, r \circ a^{\mathcal{T}'})$; see Figure 13. So, $(\mathcal{B}, a^{\mathcal{T}'} \circ f) \in \mathscr{F}''$ and $(\mathcal{B}, a^{\mathcal{T}'} \circ f) \hookrightarrow (\mathcal{A}, a^{\mathcal{T}'})$. Thus, $\mathcal{A}$ is not weakly valid w.r.t. $(\mathscr{F}'', \mathcal{T}')$, and property $\mathfrak{p}1$ holds for $(\mathscr{F}'', \mathcal{T}')$.

Consider property $\mathfrak{p}2$. As every pattern of $(\mathscr{F}, \mathcal{T})$ is automorphic, by Lemma 2, so is every pattern of $(\mathscr{F}'', \mathcal{T}')$.

Consider property $\mathfrak{p}3$. Suppose that $(\mathcal{A}, a^{\mathcal{T}'}), (\mathcal{B}, b^{\mathcal{T}'}) \in \mathscr{F}''$ are distinct and such that $(\mathcal{B}, b^{\mathcal{T}'}) \overset{f}{\hookrightarrow} (\mathcal{A}, a^{\mathcal{T}'})$. Thus, we have that $(\mathcal{A}, r \circ a^{\mathcal{T}'}), (\mathcal{B}, r \circ b^{\mathcal{T}'}) \in \mathscr{F}$ and also that $(\mathcal{B}, r \circ b^{\mathcal{T}'}) \overset{f}{\hookrightarrow} (\mathcal{A}, r \circ a^{\mathcal{T}'})$. This yields a contradiction as $(\mathscr{F}, \mathcal{T})$ satisfies property $\mathfrak{p}3$, and so $(\mathscr{F}'', \mathcal{T}')$ satisfies property $\mathfrak{p}3$.

Trivially, $(\mathscr{F}'', \mathcal{T}')$ satisfies properties $\mathfrak{p}4$ and $\mathfrak{p}5$.

Definition 33. *We say that a connected representation for which properties* $\mathfrak{p}1$–$\mathfrak{p}6$ *hold is a* normal representation*.*

Consequently, we have proven the following result.

Theorem 34. *Let $(\mathscr{F}, \mathcal{T})$ be a connected representation. Then there is an effective procedure by which we can obtain a normal representation equivalent to $(\mathscr{F}, \mathcal{T})$.*

We end this section with a theorem crucial to what follows.

Theorem 35. *Let $(\mathscr{F}, \mathcal{T})$ be a normal representation. If $\mathscr{F} \neq \emptyset$, then the target $\mathcal{T}$ is not valid w.r.t. $(\mathscr{F}, \mathcal{T})$.*

Proof. Assume for contradiction that $(\mathcal{T}, t)$ is valid w.r.t. $(\mathscr{F}, \mathcal{T})$. If t is one-to-one, then t is an isomorphism, and thus, as $\mathscr{F} \neq \emptyset$, there exists $(\mathcal{A}, a^{\mathcal{T}}) \in \mathscr{F}$ such that $t^{-1} \circ a^{\mathcal{T}}$ is a homomorphism from $(\mathcal{A}, a^{\mathcal{T}})$ to $(\mathcal{T}, t)$. This yields a contradiction, and so we may assume that t is not one-to-one.

Consider repeatedly applying the homomorphism t to obtain the homomorphism $t^k : \mathcal{T} \to \mathcal{T}$ for each $k \geq 1$. For some $k \geq 1$, it must be the case that t restricted to the image of t^k is one-to-one and thus an isomorphism. For such a k, denote the image of t^k by $\mathcal{T}'$ and the isomorphism from $\mathcal{T}'$ to $\mathcal{T}'$ induced by t by s. In particular, s^{-1} exists.

Suppose that there exists $(\mathcal{A}, a^{\mathcal{T}}) \in \mathscr{F}$ such that the image of $a^{\mathcal{T}}$ is contained in $\mathcal{T}'$. Clearly, the homomorphism $s^{-1} \circ a^{\mathcal{T}} : \mathcal{A} \to \mathcal{T}$ is well-defined and is a $\mathcal{T}$-colored homomorphism of $(\mathcal{A}, a^{\mathcal{T}})$ to $(\mathcal{T}, t)$. This contradicts our assumption that $(\mathcal{T}, t)$ is valid w.r.t. $(\mathscr{F}, \mathcal{T})$. Consequently, for every $(\mathcal{A}, a^{\mathcal{T}}) \in \mathscr{F}$, the image of $a^{\mathcal{T}}$ is not contained in $\mathcal{T}'$.

Consider the representation $(\emptyset, \mathcal{T}')$. Trivially, $(\emptyset, \mathcal{T}')$ is a retract of $(\mathscr{F}, \mathcal{T})$, and $\mathcal{T}'$ is not isomorphic to $\mathcal{T}$ (as t is not one-to-one). Thus, $(\emptyset, \mathcal{T}')$ is a proper retract of $(\mathscr{F}, \mathcal{T})$, which contradicts the fact that $(\mathscr{F}, \mathcal{T})$ is automorphic. The result follows. □

5. A generic construction of counterexamples. We prove in this section that any problem given by a normal representation $(\mathscr{F}, \mathcal{T})$ for which $\mathscr{F} \neq \emptyset$ is not in CSP. The proof involves a generic construction of a family of structures that provides, in a sense, a counterexample for any candidate for the role of a template; such a family of structures is called a witness family. The essence of the proof strategy employed originated in the proofs in [25] that certain graph problems are not in CSP.

DEFINITION 36 (witness family). *Let $(\mathscr{F}, \mathcal{T})$ be a representation. A family of structures $\mathscr{W}$ is said to be a* witness family for $(\mathscr{F}, \mathcal{T})$ *if and only if* $\mathscr{W} \subseteq \mathrm{FPP}(\mathscr{F}, \mathcal{T})$ *and for any structure $\mathcal{B}$ (over the underlying signature), there exists $\mathcal{W} \in \mathscr{W}$ such that either $\mathcal{W} \nrightarrow \mathcal{B}$ or for some $\mathcal{W} \xrightarrow{h} \mathcal{B}$, the homomorphic image $h(\mathcal{W})$ does not belong to* $\mathrm{FPP}(\mathscr{F}, \mathcal{T})$ *(the structure $\mathcal{W}$ is said to be a* witness *for $\mathcal{B}$).*

LEMMA 37. *If a representation (connected or otherwise) has a witness family, then the problem given by the representation does not belong to CSP.*

Proof. Let $\mathscr{W}$ be a witness family for some representation $(\mathscr{F}, \mathcal{T})$. Assume for contradiction that $\mathrm{FPP}(\mathscr{F}, \mathcal{T}) = \mathrm{CSP}(\mathcal{B})$ for some structure $\mathcal{B}$. By definition, there exists $\mathcal{W} \in \mathscr{W}$ such that either $\mathcal{W} \nrightarrow \mathcal{B}$ or for some $\mathcal{W} \xrightarrow{h} \mathcal{B}$, $h(\mathcal{W}) \notin \mathrm{FPP}(\mathscr{F}, \mathcal{T})$. Both cases immediately lead to a contradiction. □

We now state the main result of this section and a corollary.

THEOREM 38. *Let $(\mathscr{F}, \mathcal{T})$ be a normal representation. If $\mathscr{F} \neq \emptyset$, then there is a witness family for $(\mathscr{F}, \mathcal{T})$.*

COROLLARY 39. *If $(\mathscr{F}, \mathcal{T})$ is a normal representation for which $\mathscr{F} \neq \emptyset$, then* $\mathrm{FPP}(\mathscr{F}, \mathcal{T}) \notin \mathrm{CSP}$.

The remainder of this section is devoted to a proof of the above theorem and corollary. Throughout the remainder of this section, $(\mathscr{F}, \mathcal{T})$ is a normal representation for which $\mathscr{F} \neq \emptyset$ and where the underlying signature is σ.

Opening up a structure. By Theorem 35, the structure $\mathcal{T}$ is not valid w.r.t. $(\mathscr{F}, \mathcal{T})$. Let $t^{\mathcal{T}}$ be some homomorphism $\mathcal{T} \xrightarrow{t^{\mathcal{T}}} \mathcal{T}$ (there is at least one such homomorphism: the identity). As $(\mathscr{F}, \mathcal{T})$ is normal, we may assume that some biconnected and nonconform forbidden pattern $(\mathcal{A}, a^{\mathcal{T}})$ embeds into $(\mathcal{T}, t^{\mathcal{T}})$, via some embedding f. Let $(\mathcal{D}, d^{\mathcal{T}})$ be identical to $(\mathcal{T}, t^{\mathcal{T}})$.

It is straightforward to show that any biconnected and nonconform pattern must contain a cycle; choose one of minimal size, and let C be the image of this cycle under f (and so C is a cycle). Let x be an articulation point of C, and let $\mathbf{t}$ be a tuple of C that is incident with x (thus $R^{\mathcal{D}}(\mathbf{t})$ holds for some relation symbol R). Introduce a new element x' into the domain of $\mathcal{D}$.

- Suppose that C has size 1; i.e., $\mathbf{t}$ is not antireflexive. Replace the first occurrence of x in $R^{\mathcal{D}}(\mathbf{t})$ with the new element x' (leaving all other occurrences of all elements as is).
- Suppose that C has size 2; i.e., C consists of the antireflexive tuples $R^{\mathcal{D}}(\mathbf{t})$ and $R_1^{\mathcal{D}}(\mathbf{t}_1)$, where $\mathbf{t}$ and $\mathbf{t}_1$ have at least two distinct elements in common (one of which is x) and where if $R = R_1$, then $\mathbf{t}$ and $\mathbf{t}_1$ differ. Replace the solitary occurrence of x in $R^{\mathcal{D}}(\mathbf{t})$ by x'.
- Suppose that C has size greater than 2. Replace the solitary occurrence of x in $R^{\mathcal{D}}(\mathbf{t})$ by x'.

The elements x and x' of our amended structure are called *plug points of sort* 1. We define that $d^{\mathcal{T}}(x') = d^{\mathcal{T}}(x)$ and denote the amended $\mathcal{T}$-colored structure by $(\mathcal{D}, d^{\mathcal{T}})$ also.

If there exists a forbidden pattern of $\mathscr{F}$ that embeds into $(\mathcal{D}, d^{\mathcal{T}})$, then we proceed as above by choosing an appropriate cycle and an articulation point y of this cycle and then "breaking" the cycle by introducing a new element y' and amending a specific tuple of $\mathcal{D}$ (note that if we have a cycle of size 1 or 2, then we may need more than one amendment to "break" the cycle). Again, we define $d^{\mathcal{T}}(y') = d^{\mathcal{T}}(y)$ and denote the amended $\mathcal{T}$-colored structure by $(\mathcal{D}, d^{\mathcal{T}})$ also. As above, we refer to y and y' as plug points. If y was either x or x', then y and y' are plug points of sort 1; otherwise, they are plug points of sort 2.

We proceed iteratively in this fashion until no forbidden pattern of $\mathscr{F}$ embeds into $(\mathcal{D}, d^{\mathcal{T}})$, at each stage of the iteration fixing the sort of plug points to be inherited from the corresponding articulation point or to be of a new sort (the smallest positive integer as yet unused to describe sorts) if the corresponding articulation point had not been assigned a sort. Note that this process terminates as ultimately we would obtain a cycle-free structure (into which no forbidden pattern can embed).

Denote the resulting $\mathcal{T}$-colored σ-structure by $(\mathcal{G}, g^{\mathcal{T}})$ and call it the *gadget*. Note that $(\mathcal{G}, g^{\mathcal{T}})$ is valid w.r.t. $(\mathscr{F}, \mathcal{T})$ as no forbidden pattern embeds into $(\mathcal{G}, g^{\mathcal{T}})$ (recall that $(\mathscr{F}, \mathcal{T})$ is normal). Note also that $(\mathcal{G}, g^{\mathcal{T}}) \xrightarrow{r} (\mathcal{T}, t^{\mathcal{T}})$, where r is the homomorphism which identifies plug points of the same sort and otherwise leaves elements fixed.

Preparing for plugging. Suppose that the gadget $(\mathcal{G}, g^{\mathcal{T}})$ has p_i plug points of sort i for $i = 1, 2, \dots, k$ (and possibly other elements that have not been assigned a sort). For each $i = 1, 2, \dots, k$, define the signature σ_i as consisting of the relation symbol P_i of arity p_i. For each $i = 1, 2, \dots, k$ and each $m_i \geq p_i$, define the σ_i-structure $\mathcal{Q}_i^{m_i}$ to have domain $\{0, 1, \dots, m_i - 1\}$ and relation $P_i^{\mathcal{Q}_i^{m_i}}$ defined as

$$\{(u_1, u_2, \dots, u_{p_i}) : u_1 < u_2 < \cdots < u_{p_i}\}.$$

LEMMA 40. *Fix $b \geq 2$, fix $i \in \{1, 2, \dots, k\}$, and suppose that $m_i \geq b(p_i - 1) + 1$. For every mapping $h : |\mathcal{Q}_i^{m_i}| \to \{0, 1, \dots, b - 1\}$, there must exist at least one tuple $P_i^{\mathcal{Q}_i^{m_i}}(u_1, u_2, \cdots, u_{p_i})$ such that $h(u_1) = h(u_2) = \dots = h(u_{p_i})$.*

Proof. Suppose otherwise for the mapping h. So, there exist at most $p_i - 1$ distinct elements x of $|\mathcal{Q}_i^{m_i}|$ for which $h(x) = j$ for any $j \in \{0, 1, \dots, b - 1\}$. Thus, $|\mathcal{Q}_i^{m_i}| = m_i \leq b(p_i - 1)$, which yields a contradiction. □

Now define the signature $\overline{\sigma}$ to consist of the relation symbol P of arity $p = \Sigma_{i=1}^k p_i$. For any $m_1, m_2, \dots, m_k$ for which $m_i \geq p_i$, for each $i = 1, 2, \dots, k$, define the $\overline{\sigma}$-structure $\overline{\mathcal{Q}}$ to have a domain consisting of the disjoint union of the domains $|\mathcal{Q}_1^{m_1}|, |\mathcal{Q}_2^{m_2}|, \dots, |\mathcal{Q}_k^{m_k}|$ and relation $P^{\overline{\mathcal{Q}}}$ defined as

$$\{(\mathbf{u}^1, \mathbf{u}^2, \dots, \mathbf{u}^k) : \mathbf{u}^i \in |\mathcal{Q}_i^{m_i}|^{p_i} \text{ and } u_1^i < u_2^i < \cdots < u_{p_i}^i \\ \text{for each } i = 1, 2, \dots, k\}$$

(the notation is such that u_j^i is the jth component of the tuple $\mathbf{u}^i$). So, in a sense, $\overline{\mathcal{Q}}$ is a sort of "amalgamation" of $\mathcal{Q}_i^{m_1}, \mathcal{Q}_2^{m_2}, \dots, \mathcal{Q}_k^{m_k}$ (note that we have suppressed the parameters "$m_1, m_2, \dots, m_k$" in the denotation of $\overline{\mathcal{Q}}$ for ease of readability).

LEMMA 41. *Fix $b \geq 2$ and suppose that $m_i \geq b(p_i - 1) + 1$ for each $i = 1, 2, \dots, k$. For every mapping $h : |\overline{\mathcal{Q}}| \to \{0, 1, \dots, b - 1\}$, there must exist at least one tuple $P^{\overline{\mathcal{Q}}}(\mathbf{u}^1, \mathbf{u}^2, \dots, \mathbf{u}^k)$ such that $h(u_1^i) = h(u_2^i) = \cdots = h(u_{p_i}^i)$ for all $i = 1, 2, \dots, k$.*

Proof. The proof is immediate from Lemma 40. □

The *girth* of a structure is the length of its shortest cycle (and so if there are no cycles, then the structure has infinite girth). The following theorem is due to Feder and Vardi [15] (and generalizes a result due to Erdös; see [15]).

THEOREM 42. *Fix two positive integers r and s. For every structure $\mathcal{B}$ of size n, there exists a structure $\mathcal{B}'$ (over the same signature) of size n^a (where a depends solely on r and s) such that:*

- *the girth of $\mathcal{B}'$ is greater than r;*
- *$\mathcal{B}' \to \mathcal{B}$; and*
- *for every structure $\mathcal{C}$ of size at most s (over the same signature), $\mathcal{B} \to \mathcal{C}$ if and only if $\mathcal{B}' \to \mathcal{C}$.*

Furthermore, $\mathcal{B}'$ can be constructed from $\mathcal{B}$ in randomized polynomial time.

Remark 43. We have already mentioned that Gábor Kun has derandomized Theorem 1. To be more precise, he achieved this by giving a deterministic polynomial-time algorithm for the $\mathcal{B}'$ in the above theorem.

For each forbidden pattern $(\mathcal{A}, a^{\mathcal{T}})$ of $\mathscr{F}$, define $\gamma_{\mathcal{A}}$ to be the length of the longest cycle of $\mathcal{A}$. Define γ to be the maximum of $\{\gamma_{\mathcal{A}} : (\mathcal{A}, a^{\mathcal{T}}) \in \mathscr{F}\}$.

Fix $b \geq 2$. By applying Theorem 42, there is a $\overline{\sigma}$-structure $\overline{\mathcal{Q}}'$ of girth greater than γ for which $\overline{\mathcal{Q}}' \to \overline{\mathcal{Q}}$ and for which for every structure $\mathcal{C}$ of size at most b, $\overline{\mathcal{Q}} \to \mathcal{C}$ if and only if $\overline{\mathcal{Q}}' \to \mathcal{C}$ (of course, we assume that $m_1, m_2, \ldots, m_k$ satisfy the hypothesis of Lemma 41).

LEMMA 44. *For every mapping $h : |\overline{\mathcal{Q}}'| \to \{0, 1, \ldots, b-1\}$, there must exist at least one tuple $P^{\overline{\mathcal{Q}}'}(\mathbf{u}^1, \mathbf{u}^2, \ldots, \mathbf{u}^k)$ such that $h(u^i_1) = h(u^i_2) = \cdots = h(u^i_{p_i})$ for all $i = 1, 2, \ldots, k$.*

Proof. The condition in the statement of the lemma (and also the statement of Lemma 41, with the same value b) is equivalent to there not being a homomorphism from $\overline{\mathcal{Q}}'$ to the $\overline{\sigma}$-structure with domain $\{0, 1, \ldots, b-1\}$ and relation

$$P = \{0, 1, \ldots, b-1\}^p \setminus \{(b_1^{p_1}, b_2^{p_2}, \ldots, b_k^{p_k}) : b_i \in \{0, 1, \ldots, b-1\} \\ \text{for every } i = 1, 2, \ldots, k\}$$

(where $b_i^{p_i}$ is the p_i-tuple with each component equal to b_i). The result follows by Lemma 41 and the properties of $\overline{\mathcal{Q}}'$ detailed above. □

Building the witness family. Fix some σ-structure $\mathcal{B}$ of size b. We are now in a position to build a σ-structure $\mathcal{W}_{\mathcal{B}}$ which will act as a witness for $\mathcal{B}$ (see Definition 36).

- Initialize the domain of $\mathcal{W}_{\mathcal{B}}$ to be that of $\overline{\mathcal{Q}}'$.
- For every tuple $P^{\overline{\mathcal{Q}}'}(\mathbf{u}^1, \mathbf{u}^2, \ldots, \mathbf{u}^k)$, where each $\mathbf{u}^i \in |\overline{\mathcal{Q}}'|^{p_i}$, plug a copy of the gadget $\mathcal{G}$ by identifying the p_i sort-i plug points of $\mathcal{G}$ with the p_i "socket-points" $\mathbf{u}^i$ of $|\overline{\mathcal{Q}}'|$ for each $i = 1, 2, \ldots, k$.

All such copies of the gadget should be disjoint, except that two copies of the gadget may have plug points in common within $\mathcal{W}_{\mathcal{B}}$ and except where the gadget (possibly) contains a tuple $R^{\mathcal{G}}(\mathbf{t})$ with every element of $\mathbf{t}$ a plug point. Let us label every tuple of every relation $R^{\mathcal{W}_{\mathcal{B}}}$ with the name of the tuple of $P^{\overline{\mathcal{Q}}'}$ to which the copy of the gadget from which it comes corresponds. As just mentioned, there may be difficulties where the gadget contains a tuple $R^{\mathcal{G}}(\mathbf{t})$ with every element of $\mathbf{t}$ a plug point, as this tuple might require more than one label. In such a case, simply arbitrarily choose one label from the set of potential candidates. Finally, note that $\mathcal{W}_{\mathcal{B}} = \mathcal{W}_{\mathcal{B}'}$ whenever $|\mathcal{B}| = |\mathcal{B}'|$; i.e., the definition of $\mathcal{W}_{\mathcal{B}}$ depends solely upon b and not on the tuples of $\mathcal{B}$.

PROPOSITION 45. *The structure $\mathcal{W}_{\mathcal{B}}$ is a witness for $\mathcal{B}$.*

Proof. We begin by proving that there exists a homomorphism $\mathcal{W}_{\mathcal{B}} \xrightarrow{w^{\mathcal{T}}} \mathcal{T}$.

From above, $\overline{\mathcal{Q}}' \to \overline{\mathcal{Q}}$ via some homomorphism q. Recall that the domain of $\overline{\mathcal{Q}}$ is the disjoint union of $|\mathcal{Q}_1^{m_1}|, |\mathcal{Q}_2^{m_2}|, \ldots, |\mathcal{Q}_k^{m_k}|$. Hence, we can partition $|\overline{\mathcal{Q}}'|$ into disjoint subsets $S_1, S_2, \dots, S_k$, where for each $i = 1, 2, \dots, k$, $S_i = \{u \in |\overline{\mathcal{Q}}'| : q(u) \in |\mathcal{Q}_i^{m_i}|\}$. By definition of $P^{\overline{\mathcal{Q}}}$, if $P^{\overline{\mathcal{Q}}}(\mathbf{u}^1, \mathbf{u}^2, \dots, \mathbf{u}^k)$ holds, where $\mathbf{u}^i$ is a p_i-tuple of elements, then $\mathbf{u}^i \in S_i^{p_i}$ for $i = 1, 2, \dots, k$. In particular, in any copy of the gadget $\mathcal{G}$, plug points of sort i are always identified with "socket elements" from S_i for $i = 1, 2, \dots, k$. Consequently, the homomorphism $\mathcal{G} \xrightarrow{g^{\mathcal{T}}} \mathcal{T}$, under which plug points of the same sort are always mapped to the same element of $|\mathcal{T}|$, can be extended to a homomorphism $\mathcal{W}_{\mathcal{B}} \xrightarrow{w^{\mathcal{T}}} \mathcal{T}$.

Suppose that $(\mathcal{W}_{\mathcal{B}}, w^{\mathcal{T}})$ is not valid w.r.t. $(\mathscr{F}, \mathcal{T})$. So, some biconnected, nonconform forbidden pattern $(\mathcal{A}, a^{\mathcal{T}})$ embeds into $(\mathcal{W}_{\mathcal{B}}, w^{\mathcal{T}})$. As no forbidden pattern embeds into the gadget and each forbidden pattern is biconnected and nonconform, there must exist a cycle C in $\mathcal{W}_{\mathcal{B}}$ of length less than γ and involving tuples from at least two copies of the gadget within $\mathcal{W}_{\mathcal{B}}$ (we reiterate that each forbidden pattern is biconnected, and so if there were no such cycles, then we would have an articulation point) or equivalently, involving tuples labeled with at least two distinct tuples of $P^{\overline{\mathcal{Q}}'}$ (according to our labeling process as detailed prior to the statement of this proposition). However, the cycle C of $\mathcal{W}_{\mathcal{B}}$ yields a closed path of tuples in $\overline{\mathcal{Q}}'$ (by following the labels). Continuing, this closed path of tuples in $\overline{\mathcal{Q}}'$ yields a cycle in $\overline{\mathcal{Q}}'$ of length at least 2 and less than γ; this contradicts the fact that $\overline{\mathcal{Q}}'$ has girth greater than γ. Thus, $(\mathcal{W}_{\mathcal{B}}, w^{\mathcal{T}})$ is valid w.r.t. $(\mathscr{F}, \mathcal{T})$.

If $\mathcal{W}_{\mathcal{B}} \not\to \mathcal{B}$, then we are done. So, suppose that $\mathcal{W}_{\mathcal{B}} \xrightarrow{h} \mathcal{B}$. The homomorphism h induces a map $\hat{h} : |\overline{\mathcal{Q}}'| \to \{0, 1, \dots, b-1\}$, and so by Lemma 44, there exists a tuple $P^{\overline{\mathcal{Q}}'}(\mathbf{u}^1, \mathbf{u}^2, \dots, \mathbf{u}^k)$, where $\mathbf{u}^i \in |\overline{\mathcal{Q}}'|^{p_i}$ and $\hat{h}(u_1^i) = \hat{h}(u_2^i) = \cdots = \hat{h}(u_{p_i}^i)$ for $i = 1, 2, \dots, k$. Thus, by construction of $\mathcal{W}_{\mathcal{B}}$, $h(\mathcal{W}_{\mathcal{B}})$ contains a homomorphic image of the gadget $\mathcal{G}$ where all plug points of the same sort are mapped to the same element.

($\star$) Consequently, $h(\mathcal{W}_{\mathcal{B}})$ contains a homomorphic image of the structure $\mathcal{T}$, via some homomorphism $\tilde{h}$.

Suppose that $h(\mathcal{W}_{\mathcal{B}}) \xrightarrow{f} \mathcal{T}$. So, $\mathcal{T} \xrightarrow{f \circ \tilde{h}} \mathcal{T}$ and, by Theorem 35, there exists a forbidden pattern $(\mathcal{A}, a^{\mathcal{T}}) \in (\mathscr{F}, \mathcal{T})$ such that $(\mathcal{A}, a^{\mathcal{T}}) \xrightarrow{\tilde{f}} (\mathcal{T}, f \circ \tilde{h})$. Hence, we have that $(\mathcal{A}, a^{\mathcal{T}}) \xrightarrow{\tilde{h} \circ \tilde{f}} (h(\mathcal{W}_{\mathcal{B}}), f)$, and $h(\mathcal{W}_{\mathcal{B}}) \notin \mathrm{FPP}(\mathscr{F}, \mathcal{T})$, as required. □

Thus, we have proven Theorem 38. Lemma 37 immediately yields Corollary 39.

6. MMSNP versus CSP.

We now deal with the disconnected case before turning to the more general situation involving MMSNP and CSP.

6.1. Normal sets of representations.

6.1.1. The disconnected case.

We first turn to the situation when a representation is not necessarily connected. Let $(\mathscr{F}, \mathcal{T})$ be a representation such that there exists a disconnected forbidden pattern $(\mathcal{A}, a^{\mathcal{T}}) \in \mathscr{F}$; that is, $(\mathcal{A}, a^{\mathcal{T}})$ is the disjoint union of two colored structures $(\mathcal{B}, b^{\mathcal{T}})$ and $(\mathcal{C}, c^{\mathcal{T}})$. Define $\mathscr{F}' = (\mathscr{F} \setminus \{(\mathcal{A}, a^{\mathcal{T}})\}) \cup \{(\mathcal{B}, b^{\mathcal{T}})\}$ and $\mathscr{F}'' = (\mathscr{F} \setminus \{(\mathcal{A}, a^{\mathcal{T}})\}) \cup \{(\mathcal{C}, c^{\mathcal{T}})\}$. Trivially, we have that

$$\mathrm{FPP}(\mathscr{F}, \mathcal{T}) = \mathrm{FPP}(\mathscr{F}', \mathcal{T}) \cup \mathrm{FPP}(\mathscr{F}'', \mathcal{T}).$$

By iterating this construction, we can transform $(\mathscr{F}, \mathcal{T})$ into a set of connected representations so that a structure is in $\mathrm{FPP}(\mathscr{F}, \mathcal{T})$ if and only if it is in at least one of the forbidden patterns problems corresponding to the derived connected representations.

Next, we compute the normal representation of each connected representation, just as we did in section 4. Finally, we enforce the following property on our set of normal representations:

($\mathfrak{p}7$) For any two normal representations $(\mathscr{F}', \mathcal{T}')$ and $(\mathscr{F}'', \mathcal{T}'')$, we have that $(\mathscr{F}', \mathcal{T}') \nrightarrow (\mathscr{F}'', \mathcal{T}'')$.

This property is enforced by simply removing the normal representation $(\mathscr{F}', \mathcal{T}')$ from the collection should there exist another (different) normal representation $(\mathscr{F}'', \mathcal{T}'')$ for which $(\mathscr{F}', \mathcal{T}') \rightarrow (\mathscr{F}'', \mathcal{T}'')$.

Consequently, we may assume that any representation $(\mathscr{F}, \mathcal{T})$ corresponds to a collection $\mathfrak{N}$ of normal representations (possibly containing only one such representation) for which property $\mathfrak{p}7$ holds; we call $\mathfrak{N}$ the *normal set* corresponding to $(\mathscr{F}, \mathcal{T})$. By Proposition 26, the problem FPP$(\mathscr{F}, \mathcal{T})$ is the union of the forbidden patterns problems of the representations in the normal set $\mathfrak{N}$; that is,

$$\mathrm{FPP}(\mathscr{F}, \mathcal{T}) = \bigcup \{\mathrm{FPP}(\mathscr{F}', \mathcal{T}') : (\mathscr{F}', \mathcal{T}') \in \mathfrak{N}\}.$$

6.1.2. Finite unions of forbidden patterns problems. The notion of a normal set extends naturally to finite unions of forbidden patterns problems: Given a finite set of representations, we split every disconnected representation into a set of connected representations as above, take the union of all of these sets, and simplify these sets so as to enforce $\mathfrak{p}7$. We write FPP$(\mathfrak{N})$ for $\bigcup_{(\mathfrak{F}, \mathcal{T}) \in \mathfrak{N}} \mathrm{FPP}(\mathfrak{F}, \mathcal{T})$.

PROPOSITION 46. *Let $\mathfrak{N}$ be a normal set that contains a representation $(\mathscr{F}', \mathcal{T}')$ such that $\mathscr{F}' \neq \emptyset$. Then $\mathcal{T}'$ is a no instance of* FPP$(\mathfrak{N})$.

Proof. By Theorem 35, if $\mathcal{T}'$ is valid w.r.t. $(\mathscr{F}', \mathcal{T}')$, then $\mathscr{F}' = \emptyset$. Thus, $\mathcal{T}'$ is not valid w.r.t. $(\mathscr{F}', \mathcal{T}')$.

Suppose that $\mathcal{T}'$ is valid w.r.t. $(\mathscr{F}'', \mathcal{T}'')$, where $(\mathscr{F}'', \mathcal{T}'')$ is a representation in $\mathfrak{N}$ distinct from $(\mathscr{F}', \mathcal{T}')$. That is, there exists a homomorphism $r : \mathcal{T}' \to \mathcal{T}''$ such that for every forbidden pattern $(\mathcal{A}'', a^{\mathcal{T}''}) \in \mathscr{F}''$, $(\mathcal{A}'', a^{\mathcal{T}''}) \nrightarrow (\mathcal{T}', r)$. In particular, if $(\mathcal{A}'', a^{\mathcal{T}''}) \in \mathscr{F}''$, then there does not exist a homomorphism $a^{\mathcal{T}'} : \mathcal{A}'' \to \mathcal{T}'$ for which $r \circ a^{\mathcal{T}'} = a^{\mathcal{T}''}$. Consequently, r is (trivially) a recoloring of $(\mathscr{F}', \mathcal{T}')$ to $(\mathscr{F}'', \mathcal{T}'')$. This yields a contradiction, and so $\mathcal{T}'$ is not valid w.r.t. $(\mathscr{F}'', \mathcal{T}'')$. The result follows. □

6.2. Finite unions.

DEFINITION 47 (strong witness family). *Let $\mathfrak{N}$ be a set of representations. A family of structures $\mathscr{W}$ is said to be a* strong witness family for $\mathfrak{N}$ *if and only if $\mathscr{W} \subseteq$* FPP$(\mathfrak{N})$ *and for any finite set of structures $\{\mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_n\}$ (over the underlying signature), there exists $\mathcal{W} \in \mathscr{W}$ such that for every $1 \leq i \leq n$, either $\mathcal{W} \nrightarrow \mathcal{B}_i$ or for some $\mathcal{W} \xrightarrow{h} \mathcal{B}_i$, the homomorphic image $h(\mathcal{W})$ does not belong to* FPP$(\mathfrak{N})$ *(the structure $\mathcal{W}$ is said to be a* strong witness *for $\mathcal{B}$).*

LEMMA 48. *If a set of representations $\mathfrak{N}$ has a strong witness family, then the problem* FPP$(\mathfrak{N})$ *is not a finite union of constraint satisfaction problems.*

Proof. Let $\mathscr{W}$ be a strong witness family for some representation $(\mathscr{F}, \mathcal{T})$. Assume for contradiction that

$$\mathrm{FPP}(\mathfrak{N}) = \bigcup_{(\mathfrak{F}, \mathcal{T}) \in \mathfrak{N}} \mathrm{FPP}(\mathfrak{F}, \mathcal{T}) = \bigcup_{1 \leq i \leq n} \mathrm{CSP}(\mathcal{B}_i)$$

for some finite set of structures $\{\mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_n\}$. By definition, there exists a strong witness $\mathcal{W} \in \mathscr{W}$. Since $\mathcal{W}$ is a yes instance of FPP$(\mathfrak{N})$, we have that $\mathcal{W} \in \mathrm{CSP}(\mathcal{B}_i)$ for some $1 \leq i \leq n$. Hence, by definition of a strong witness, there is a homomorphism $\mathcal{W} \xrightarrow{h} \mathcal{B}_i$ such that $h(\mathcal{W}) \notin \mathrm{FPP}(\mathfrak{N})$. However, $h(\mathcal{W}) \in \mathrm{CSP}(\mathcal{B}_i) = \mathrm{FPP}(\mathfrak{N})$, which is absurd. □

We can extend the main result of the previous section to finite unions of forbidden patterns problems and, in particular, to disconnected representations. We first deal with the case when the normal set corresponds to a finite union of constraint satisfaction problems.

THEOREM 49. *Let $\mathfrak{N}$ be a normal set of the form $\{(\emptyset, \mathcal{T}_1), (\emptyset, \mathcal{T}_2), \dots, (\emptyset, \mathcal{T}_n)\}$. Then* $\mathrm{FPP}(\mathfrak{N}) = \bigcup_{1 \leq i \leq n} \mathrm{CSP}(\mathcal{T}_i)$. *Moreover, if*

$$\bigcup_{1\leq i\leq n} \mathrm{CSP}(\mathcal{T}_i) = \bigcup_{1\leq i\leq m} \mathrm{CSP}(\mathcal{T}_i'),$$

then the following hold:

(i) *for every $1 \leq i \leq m$, there exists $1 \leq j \leq n$ such that $\mathcal{T}_i' \rightarrow \mathcal{T}_j$;*

(ii) *for every $1 \leq i \leq n$, there exists $1 \leq j \leq m$ such that $\mathcal{T}_i$ is the core of $\mathcal{T}_j'$;*

(iii) *$m \geq n$.*

Proof. Property (i) follows directly from the fact that $\mathcal{T}_i' \in \mathrm{CSP}(\mathcal{T}_i')$. We now prove (ii). Using a similar argument as above, there exists $\mathcal{T}_j'$ such that $\mathcal{T}_i \rightarrow \mathcal{T}_j'$. By (i), there exists some $\mathcal{T}_k$ such that $\mathcal{T}_j' \rightarrow \mathcal{T}_k$. By composition, $\mathcal{T}_i \rightarrow \mathcal{T}_k$. Recall that, by definition of the normal set, there is no homomorphism between any $\mathcal{T}_i$ and $\mathcal{T}_k$ for any i such that $1 \leq i < k \leq n$. Moreover, every $\mathcal{T}_i$ is automorphic. Thus, $i = k$, and it follows that $\mathcal{T}_i$ is homomorphically equivalent to $\mathcal{T}_j'$. This proves that $\mathcal{T}_i$ is the core of $\mathcal{T}_j'$. Property (iii) follows from (ii) since $\mathcal{T}_i$ and $\mathcal{T}_k$, for any i, k such that $i \neq k$, cannot be the core of the same $\mathcal{T}_j'$; otherwise, they would be isomorphic (by uniqueness of the core). This concludes the proof. □

We can now precisely characterize when a normal set does not give rise to a finite union of constraint satisfaction problems.

THEOREM 50. *The following are equivalent:*

(i) *the normal set $\mathfrak{N}$ contains a representation $(\mathscr{F}', \mathcal{T}')$, with $\mathscr{F}' \neq \emptyset$;*

(ii) *the problem* $\mathrm{FPP}(\mathfrak{N})$ *is not a finite union of constraint satisfaction problems;*

(iii) *there exists a strong witness family for $\mathfrak{N}$.*

Proof. The implication (ii) $\Longrightarrow$ (i) is the contrapositive of the (trivial statement in the) previous theorem. The implication (iii) $\Longrightarrow$ (ii) holds by Lemma 48. We now prove that (i) $\Longrightarrow$ (iii).

The case when $\mathfrak{N}$ is a singleton is a direct corollary of the proof of Theorem 38, as the construction of a witness family can be easily adapted to obtain a strong witness family. Indeed, as is pointed out just before the statement of Proposition 45, the construction of $\mathcal{W}_{\mathcal{B}}$ depends only on the size of $\mathcal{B}$. So, for a set of structures $\mathscr{B} = \{\mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_n\}$, we build $\mathcal{W}_{\mathcal{B}_i}$, where $\mathcal{B}_i$ is a structure with the largest domain within this set. Now, for any structure $\mathcal{C}$ such that $|\mathcal{C}| \leq |\mathcal{B}_i|$, if $\mathcal{W}_{\mathcal{B}_i} \xrightarrow{h} \mathcal{C}$, then $h(\mathcal{W}_{\mathcal{B}_i})$ contains a homomorphic image of the structure chosen as a basis for our gadget, namely, $\mathcal{T}'$ (see ($\star$) in the proof of Proposition 45), which is not a yes instance of $\mathrm{FPP}(\mathfrak{N})$ (otherwise, $\mathcal{T}'$ would also be a yes instance of $\mathrm{FPP}(\mathfrak{N})$, which would contradict Theorem 35). This means that $\mathcal{W}_{\mathcal{B}_i}$ is a strong witness for $\mathscr{B}$.

Suppose now that $\mathfrak{N}$ is not a singleton. By Proposition 46, $\mathcal{T}'$ is not valid w.r.t. $(\mathscr{F}'', \mathcal{T}'')$ for any $(\mathscr{F}'', \mathcal{T}'') \in \mathfrak{N}$. Thus, we may choose $\mathcal{T}'$ as the basis of our gadget and proceed as in the case of a singleton in order to get a strong witness family for $\mathfrak{N}$. □

6.3. The main result. We need a last definition before we can state the main result of this paper. Let Φ be a sentence of MMSNP. We call a *normal set of* Φ the normal set of the set of representations obtained from Φ as follows: First, Φ is logically equivalent to a finite set of primitive sentences, which we can build effectively as in the proof of Proposition 11; second, each such primitive sentence captures precisely a forbidden patterns problem (again, this is effective; see Theorem 12); finally, we compute the normal set of this set of representations. The main result of this paper is an exact characterization of the strict inclusion of MMSNP in CSP.

THEOREM 51. *Let Φ be a sentence of MMSNP. The problem defined by Φ is in CSP if and only if its normal set consists of a singleton $(\emptyset, \mathcal{T})$.*

Proof. The result follows from the definition of the normal set of Φ and from Theorems 49 and 50. □

7. Concluding remarks. Building upon a previous attempt by Feder and Vardi to provide a logical characterization of constraint satisfaction problems, we have introduced a new class of combinatorial problems, the forbidden patterns problems, and shown that they provide a combinatorial characterization of the logic MMSNP. Furthermore, we have provided a complete classification as to when forbidden patterns problems are in CSP, and there exists an effective procedure to decide whether a given forbidden patterns problem (or problem described by a sentence of MMSNP) is in CSP or not.

We end by describing two directions for further research. Tardif and Nešetřil [31] have characterized *duality pairs*, which correspond essentially to forbidden patterns problems with a single color (the target as only one element) that are also constraint satisfaction problems. Their elegant proof relies on a correspondence between these duality pairs and the notion of *density* (with respect to the partial order given by the existence of a homomorphism). This correspondence exists essentially because one can define the notion of the *exponential of a structure* (in graph theory, this notion plays an important role in relation with Hedetniemi's conjecture [29]). It turns out that a notion of the exponential of a representation can also be defined [24]. In a forthcoming paper, we will elaborate on this and delineate the relationship between the two approaches.

Another direction for further research relates to the containment problem and is as follows. A homomorphism problem is given by its template; hence, given two homomorphism problems CSP($\mathcal{A}$) and CSP($\mathcal{B}$) over the same signature, it is decidable whether CSP($\mathcal{A}$) $\subseteq$ $CSP(\mathcal{B})$. As a matter of fact, the containment problem for homomorphism problems is nothing other than the uniform homomorphism problem, known to be NP-complete (as we noted in Remark 5). We would like to extend this result to the more general containment problem for forbidden patterns problems (given by their representations). Indeed, Feder and Vardi proved in [15] that the containment problem for MMSNP is decidable; hence by Theorem 13, it follows that the containment problem for forbidden patterns problems is decidable. However, to the best of our knowledge, nothing has been proved about the *complexity* of the containment problem for MMSNP.

We know that the existence of a recoloring implies the containment of the corresponding problems, and this provokes the following question: "*Does the existence of a recoloring correspond to the containment of the corresponding problems*?" However, we can answer this question negatively. Indeed, the major inconvenience of forbidden patterns problems, in comparison with homomorphism problems, is that the inclusion of two problems does not necessarily reduce to the question of the existence of a recoloring; for, in [24], an example is given where a representation is transformed into an equivalent representation, using Feder–Vardi reductions, but such that the representations are not equivalent with respect to recolorings. However, we think that the right notion of a morphism for representations should constitute a *finite sequence of recol-*

orings and Feder–Vardi reductions. More precisely, we believe that the following question can be answered affirmatively: "*Does the existence of a recoloring correspond to the containment of the corresponding problems in the case of normal (connected) representations*?" In [24], a few restricted cases for which an affirmative answer to the above question is obtained, and this leads us to propose the following conjecture (where for any representation $\mathscr{R}$, $\mathbf{normal}(\mathscr{R})$ is a normal representation equivalent to $\mathscr{R}$).

CONJECTURE 52. *Let* $\mathscr{R}_1$ *and* $\mathscr{R}_2$ *be two nontrivial connected representations.* $\mathrm{FPP}(\mathscr{R}_1) \subseteq \mathrm{FPP}(\mathscr{R}_2)$ *if and only if* $\mathbf{normal}(\mathscr{R}_1) \rightarrow \mathbf{normal}(\mathscr{R}_2)$.

Acknowledgment. The first author thanks Etienne Grandjean for his continual support.

REFERENCES

[1] M. Benedikt and L. Segoufin, *Regular tree languages definable in FO*, in Proceedings of the 22nd International Symposium on Theoretical Aspects of Computer Science (STACS'05), V. Diekert and B. Durand, eds., Lect. Notes Comput. Sci. 3404, Springer-Verlag, Berlin, 2006, pp. 327–339.

[2] M. Bodirsky and V. Dalmau, *Datalog and constraint satisfaction with infinite templates*, in Proceedings of the 23rd International Symposium on Theoretical Aspects of Computer Science (STACS'06), B. Durand and W. Thomas, eds., Lect. Notes Comput. Sci. 3884, Springer-Verlag, Berlin, 2006, pp. 646–659.

[3] A. A. Bulatov, *A dichotomy theorem for constraints on a three-element set*, in Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science (FOCS'02), 2002, pp. 649–658.

[4] A. A. Bulatov, *Tractable conservative constraint satisfaction problems*, in Proceedings of the 18th IEEE Symposium on Logic in Computer Science (LICS'03), 2003, pp. 321–330.

[5] A. A. Bulatov and V. Dalmau, *Towards a dichotomy theorem for the counting constraint satisfaction problem*, in Proceedings of the 44th IEEE Symposium on Foundations of Computer Science (FOCS'03), 2003, pp. 562–573.

[6] A. A. Bulatov, A. A. Krokhin, and P. Jeavons, *Constraint satisfaction problems and finite algebras*, in Proceedings of the 27th International Colloquium on Automata, Languages and Programming (ICALP'00), U. Montanari, J. D. P. Rolim and E. Welzl, eds., Lect. Notes Comput. Sci. 1853, Springer-Verlag, Berlin, 2000, pp. 272–282.

[7] A. K. Chandra and P. M. Merlin, *Optimal implementation of conjunctive queries in relational databases*, in Proceedings of the 9th ACM Symposium on Theory of Computing (STOC'77), 1977, pp. 77–90.

[8] J. Diaz, M. Serna, and D. M. Thilikos, *The complexity of restrictive H-coloring*, in Proceedings of the 28th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'02), L. Kucera, ed., Lect. Notes Comput. Sci. 2573, Springer-Varlag, Berlin, 2002, pp. 126–137.

[9] M. Dyer and C. Greenhill, *The complexity of counting graph homomorphisms*, Random Structures Algorithms, 17 (2000), pp. 260–289.

[10] R. Fagin, *Generalized first-order spectra and polynomial-time recognizable sets*, in Complexity of Computation, R. M. Karp, ed., SIAM-AMS Proc. 7, 1974, pp. 43–73.

[11] T. Feder, *Classification of homomorphisms to oriented cycles and of k-partite satisfiability*, SIAM J. Discrete Math., 14 (2001), pp. 471–480.

[12] T. Feder and P. Hell, *List constraint satisfaction and list partition*, manuscript.

[13] T. Feder, P. Hell, and J. Huang, *List homomorphisms and circular arc graphs*, Combinatorica, 19 (1999), pp. 487–505.

[14] T. Feder, P. Hell, and J. Huang, *Bi-arc graphs and the complexity of list homomorphisms*, J. Graph Theory, 42 (1999), pp. 61–80.

[15] T. Feder and M. Y. Vardi, *The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory*, SIAM J. Comput., 28 (1998), pp. 57–104.

[16] H. Gaifman, H. Mairson, Y. Sagiv, and M. Y. Vardi, *Undecidable optimization problems for database logic programs*, J. ACM, 40 (1993), pp. 683–713.

[17] G. Hahn and C. Tardif, *Graph homomorphisms: Structure and symmetry*, in NATO Sci. Ser. C Math. Phys. Sci. 497, Kluwer, 1997, pp. 107–166.

[18] P. Hell and J. Nešetřil, *On the complexity of H-coloring*, J. Combin. Theory Ser. B, 48 (1990), pp. 92–110.

[19] P. Hell and J. Nešetřil, *Graphs and Homomorphisms*, Oxford Lecture Ser. Math. Appl., Oxford University Press, London, 2004.

[20] P. Jeavons, D. A. Cohen, and M. Gyssens, *Closure properties of constraints*, J. ACM, 44 (1997), pp. 527–548.

[21] P. G. Kolaitis and M. Y. Vardi, *Conjunctive-query containment and constraint satisfaction*, J. Comput. System Sci., 61 (2000), pp. 302–332.

[22] R. E. Ladner, *On the structure of polynomial time reducibility*, J. ACM, 22 (1975), pp. 155–171.

[23] B. Larose, C. Loten, and C. Tardif, *A characterisation of first-order constraint satisfaction problems*, in Proceedings of the 21st IEEE Symposium on Logic in Computer Science (LICS'06), IEEE, 2006, pp. 201–210.

[24] F. R. Madelaine, *Constraint satisfaction problems and related logic*, Ph.D. thesis, University of Leicester, Leicester, 2003.

[25] F. R. Madelaine and I. A. Stewart, *Some problems not definable using structure homomorphisms*, Ars Combin., 67 (2003), pp. 153–159.

[26] C. H. Papadimitriou, *Computational Complexity*, Addison-Wesley, Reading, MA, 1994.

[27] A. Puricella and I. A. Stewart, *A generic greedy algorithm, partially-ordered graphs and NP-completeness*, in Proceedings of the 27th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'01), A. Brandstaedt and V. B. Le, eds., Lect. Notes Comput. Sci. 2204, Springer-Verlag, Berlin, 2001, pp. 306–316.

[28] A. Puricella and I. A. Stewart, *Greedy algorithms, H-colourings and a complexity-theoretic dichotomy*, Theoret. Comput. Sci., 290 (2003), pp. 1897–1913.

[29] N. Sauer, *Hedetniemi's conjecture – a survey*, Discrete Math., 229 (2001), pp. 261–292.

[30] T. J. Schaefer, *The complexity of satisfiability problems*, in Proceedings of the 10th ACM Symposium on Theory of Computing (STOC'78), 1978, pp. 216–226.

[31] C. Tardif and J. Nešetřil, *Duality theorems for finite structures (characterizing gaps and good characterizations)*, J. Combin. Theory Ser. B, 80 (2000), pp. 80–97.

SPECIAL SECTION ON FOUNDATIONS OF COMPUTER SCIENCE

This volume comprises the polished and fully refereed versions of a selection of papers presented at the Forty-Fifth Annual IEEE Symposium on Foundations of Computer Science (FOCS 2004), held in Rome, Italy, October 17–19, 2004. Unrefereed preliminary versions of the papers presented at the symposium appeared in the proceedings of the meeting, published by IEEE.

The FOCS 2004 Program Committee consisted of Dimitris Achlioptas, Micah Adler, Eli Ben-Sasson, Faith Fich, Oded Goldreich, Martin Grohe, Sean Hallgren, Johan Håstad, Giuseppe F. Italiano, Vladlen Koltun, Yuval Rabani, Miklos Santha, Leonard Schulman, Rocco Servedio, D. Sivakumar, Eli Upfal, and David Williamson.

Out of 272 "Extended Abstracts" submitted to the FOCS 2004 Program Committee, 64 were selected for presentation at the symposium. Eight of those 64 papers are included in this volume. This collection encompasses a wide variety of questions and methods in theoretical computer science, often shedding new light on entire areas with a fresh approach. The topics include fundamental questions of complexity theory and algorithms as well as foundational mathematical problems. All papers were refereed in accordance with SICOMP's stringent standards, and most were substantially updated in the process.

We take this opportunity to thank all the referees whose anonymous work has significantly contributed to the value of this volume. It was an honor to edit this special section in *SIAM Journal on Computing*.

Dimitris Achlioptas and Vladlen Koltun
Guest Editors

SIAM J. COMPUT.
Vol. 37, No. 1, pp. 166–194

ADIABATIC QUANTUM COMPUTATION IS EQUIVALENT TO STANDARD QUANTUM COMPUTATION[*]

DORIT AHARONOV[†], WIM VAN DAM[‡], JULIA KEMPE[§], ZEPH LANDAU[¶], SETH LLOYD[‖], AND ODED REGEV[**]

Abstract. Adiabatic quantum computation has recently attracted attention in the physics and computer science communities, but its computational power was unknown. We describe an efficient adiabatic simulation of any given quantum algorithm, which implies that the adiabatic computation model and the conventional quantum computation model are polynomially equivalent. Our result can be extended to the physically realistic setting of particles arranged on a two-dimensional grid with nearest neighbor interactions. The equivalence between the models allows stating the main open problems in quantum computation using well-studied mathematical objects such as eigenvectors and spectral gaps of sparse matrices.

Key words. quantum computation, adiabatic computation, nearest neighbor interactions

AMS subject classifications. 81P68, 68Q05

DOI. 10.1137/S0097539705447323

1. Introduction.

The study of adiabatic quantum computation was initiated several years ago by Farhi, Goldstone, Gutmann, and Sipser [14], who suggested a novel quantum algorithm for solving classical optimization problems such as SATISFIABILITY (SAT). Their algorithm is based on a celebrated theorem in quantum mechanics known as *the adiabatic theorem* [19,25]. Several simulations (see, e.g., [13]) on random instances of up to 20 quantum bits led to various optimistic speculations. The bad news is that there is now mounting evidence [9, 10, 29] that the algorithm of [14] takes exponential time in the worst-case for NP-complete problems. Nevertheless, adiabatic computation was since shown to be promising in other less ambitious directions: it possesses several interesting algorithmic capabilities, as we will soon review, and in addition it exhibits inherent robustness against certain types of quantum errors [8]. We note that a small scale adiabatic algorithm has already been implemented experimentally, using a nuclear magnetic resonance (NMR) system [36].

[*] Received by the editors March 21, 2005; accepted for publication (in revised form) August 14, 2005; published electronically May 14, 2007. The first author's work is supported in part by ARO grant DAAD19-03-1-0082, NSF ITR grant CCR-0121555, ISF grant 032-9738, and an Alon fellowship. The second author's work was supported in part by the U.S. Department of Energy (DOE) and cooperative research agreement DF-FC02-94ER40818, a CMI postdoctoral fellowship, and an HP/MSRI fellowship. The third author's work was partly sponsored by DARPA and AFL, Air Force Material Command, USAF, under agreement F30602-01-2-0524 and FDN00014-01-1-0826 and by ACI Sécurité Informatique, 2003-n24, project "Réseaux Quantiques," ACI-CR 2002-40, and EU 5th framework program RESQ IST-2001-37559. The sixth author's effort is supported by an Alon fellowship, the Israeli Science Foundation, ARO grant DAAD19-03-1-0082, and NSF grant CCR-9987845.

http://www.siam.org/journals/sicomp/37-1/44732.html

[†] School of Computer Science and Engineering, Hebrew University, Jerusalem, Israel (doria@cs.huji.il).

[‡] Department of Computer Science, UC Santa Barbara, Santa Barbara, CA 93106.

[§] CNRS-LRI UMR 8623, Université de Paris-Sud, Orsay, France.

[¶] Department of Mathematics, City College of New York, New York, NY 10031 (landau@sci.ccny.cuny.edu).

[‖] Department of Mechanical Engineering, MIT, Cambridge, MA 02139 (slloyd@mit.edu).

[**] Computer Science Department, Tel Aviv University, Israel.

We briefly describe the model of adiabatic computation (a more precise description appears in section 2.2). A computation in this model is specified by two Hamiltonians named H_{init} and H_{final} (a Hamiltonian is simply a Hermitian matrix). The eigenvector with smallest eigenvalue (also known as the *ground state*) of H_{init} is required to be an easy to prepare state, such as a tensor product state. The output of the adiabatic computation is the ground state of the final Hamiltonian H_{final}. Hence, we choose an H_{final} whose ground state represents the solution to our problem. We require the Hamiltonians to be *local*, i.e., we require them to only involve interactions between a constant number of particles (this can be seen as the equivalent of allowing gates operating on a constant number of qubits in the standard model). This, in particular, makes sure that the Hamiltonians have a short classical description, by simply listing the matrix entries of each local term. The running time of the adiabatic computation is determined by the minimal spectral gap[1] of all the Hamiltonians on the straight line connecting H_{init} and H_{final}: $H(s) = (1-s)H_{\text{init}} + sH_{\text{final}}$ for $s \in [0, 1]$. More precisely, the adiabatic computation is in polynomial time if this minimal spectral gap is at least inverse polynomial.

The motivation for the above definition is physical. The Hamiltonian operator corresponds to the energy of the quantum system, and for it to be physically realistic and implementable it must be local. Its ground state is the state of lowest energy. We can set up a quantum system in the ground state of H_{init} (which is supposed to be easy to generate) and apply the Hamiltonian H_{init} to the system. We then slowly modify the Hamiltonian along the straight line from H_{init} towards H_{final}. It follows from the adiabatic theorem that if this transformation is performed slowly enough (how slow is determined by the minimal spectral gap), the final state of the system will be in the ground state of H_{final}, as required.

What is the computational power of this model? In order to refer to the adiabatic model as a computational model that computes classical functions (rather than quantum states), we consider the result of the adiabatic computation to be the outcome of a measurement of one or more of the qubits, performed on the final ground state. It is known that adiabatic computation can be efficiently simulated by standard quantum computers [9,13]. Hence, its computational power is not greater than that of standard quantum computers. Several positive results are also known. In [9, 30] it was shown that Grover's quadratic speed-up for an unsorted search [16] can be realized as an adiabatic computation. Moreover, [11,29,32] showed that adiabatic computation can "tunnel" through wide energy barriers and thus outperform simulated annealing, a classical counterpart of the adiabatic model. However, whether or not adiabatic computation can achieve the full power of quantum computation was not known. In fact, it was even unknown whether adiabatic computation can simulate general *classical* computations efficiently. The focus of this paper is the exact characterization of the computational power of adiabatic computation.

Before we describe our results, let us clarify one subtle point. Most of the previous work on the subject focused on a restricted class of adiabatic algorithms that can be referred to as adiabatic *optimization* algorithms. In these algorithms, H_{final} is chosen to be a diagonal matrix, corresponding to a combinatorial optimization problem. In particular, this implies that the ground state of H_{final} (which is the output of the computation) is a classical state, i.e., a state in the computational basis. In this paper, however, we associate the term *adiabatic computation* with the more general class of adiabatic algorithms, where the only restriction on H_{final} is that it is a local

[1] The spectral gap is the difference between the lowest and second lowest eigenvalue.

Hamiltonian. We do this because, from a physical point of view, there is no reason to force the physical process described above to have a diagonal H_{final}, when all other Hamiltonians are not restricted this way. Thus, our definition of adiabatic computation seems to be the natural one to use. See subsection 1.6 for a further discussion of adiabatic optimization.

1.1. Results—Computational complexity of the adiabatic model. Our main result clarifies the question of the computational power of adiabatic algorithms.

Theorem 1.1. *The model of adiabatic computation is polynomially equivalent to the standard model of quantum computation.*

As mentioned above, one direction of the equivalence is already known [9, 13]. Our contribution is to show that standard quantum computation can be efficiently simulated by adiabatic computation. We do this by using adiabatic computation with 3-local Hamiltonians. We note that [4] made a preliminary step in the direction of Theorem 1.1 but the model that they considered was quite different.[2]

One corollary of our main theorem is the following. We can consider the model of adiabatic computation with a more general set of Hamiltonians known as *explicit sparse* Hamiltonians. These are Hermitian matrices that have at most polynomially many nonzero elements in each row and column, and, moreover, for which there is an efficient Turing machine that can generate a list of all nonzero entries in a given row or column. Clearly, local Hamiltonians are a special case of explicit sparse Hamiltonians. It was shown in [4] that adiabatic computation with explicit sparse Hamiltonians can still be simulated by standard quantum computation (this extends the result of [9, 14] in a nontrivial way). Hence, we obtain the following result.

Corollary 1.2. *The model of adiabatic computation with explicit sparse Hamiltonians is polynomially equivalent to the standard model of quantum computation.* Explicit sparse matrices are pervasive in computer science and combinatorics, and hence this corollary might be more useful than Theorem 1.1 in the context of the design of quantum algorithms and the study of quantum complexity.

To summarize, our results show that questions about quantum computation can be equivalently considered in the model of adiabatic computation, a model that is quite different from the more common circuit-based models. In particular, the results imply that we do not lose in computational power by trying to design quantum algorithms in the adiabatic framework. There are two reasons why taking this approach seems worthwhile. First, there are several known powerful techniques to analyze spectral gaps of matrices, including expander theory [15] and rapidly mixing Markov chains [24, 34]. Indeed, probability theory is often used in mathematical physics to analyze spectral gaps of Hamiltonians (see, e.g., [35]), and our proofs also make extensive use of Markov chain tools. Second, it is known that many interesting algorithmic problems in quantum computation can be cast as quantum state generation problems [4]. The problem of generating special quantum states seems more natural in the adiabatic model than in the standard model. Finally, we mention here that the results also give a different perspective on lower bounds on quantum computational complexity.

1.2. Results—Towards experimental implications. Theorem 1.1 uses 3-local Hamiltonians that act on particles that may be arbitrarily far apart. From a

[2] Namely, [4] showed that adiabatic computation using *simulatable* Hamiltonians is as powerful as standard quantum computation. Simulatable Hamiltonians are Hamiltonians that can be simulated efficiently by a quantum circuit. They are very different from local Hamiltonians, and cannot even be written explicitly. Instead, such Hamiltonians are specified using products of local unitary matrices.

practical point of view, it is often difficult to create controlled interactions between particles located far away from each other. Moreover, three-particle Hamiltonians are technologically very difficult to realize. If one wants to physically realize the adiabatic algorithms, it would be much better to have only 2-local interactions between nearest neighbor particles. To this end we prove the following theorem. This puts our result in a slightly more physically realistic context.

Theorem 1.3. *Any quantum computation can be efficiently simulated by an adiabatic computation with two-local nearest neighbor Hamiltonians operating on six-state particles set on a two-dimensional grid.*

The need for six-state particles arises from our construction. It is an open question whether this can be improved.

Theorems 1.1 and 1.3 open up the possibility of physically realizing universal quantum computation using adiabatically evolving quantum systems. As mentioned before, there are possible advantages to this approach: adiabatic quantum computation is resilient to certain types of noise [8]. An important component of this resilience is the existence of a spectral gap in the Hamiltonian. It is well known in physics that such a gap plays an important role in the context of protecting quantum systems from noise. However, it remains to be further studied, both experimentally and theoretically, what the right model for noisy adiabatic computation is, and whether fault tolerant adiabatic computation can be achieved. We refer the reader to further discussion in subsection 1.6.

1.3. Proof of Theorem 1.1: Overview. Given an arbitrary quantum circuit [26], our goal is to design an adiabatic computation whose output is the same as that of the quantum circuit. Some similarities between the models are obvious: one model involves unitary gates on a constant number of qubits, while the other involves local Hamiltonians. However, after some thought, one eventually arrives at the following difficulty. The output state of the adiabatic computation is the ground state of H_{final}. The output state of the quantum circuit is its final state, which is unknown to us. How can we specify H_{final} without knowing the output state of the quantum circuit? Notice that this state can be some complicated quantum superposition. One might wonder why our task is not trivial, since this state does have an efficient local classical description, namely the quantum circuit. However, local quantum gates, which operate in sequence to generate a nonlocal overall action, are very different from local Hamiltonians, which correspond to simultaneous local constraints. To explain the solution, we first set some notations.

Without loss of generality we assume that the input to the quantum circuit consists of n qubits all initialized to $|0\rangle$'s.[3] Then, a sequence of L unitary gates, $U_1, \ldots, U_L$, each operating on one or two qubits, is applied to the state. The system's state after the ℓth gate is $|\alpha(\ell)\rangle$. The output of the quantum circuit is in general a complicated quantum state $|\alpha(L)\rangle$ of n qubits, which is then measured in the standard basis. We now want to associate with it a corresponding adiabatic computation.

A first natural attempt would be to define H_{final} as a local Hamiltonian with $|\alpha(L)\rangle$ as its ground state. However, this attempt encounters the difficulty mentioned above: not knowing $|\alpha(L)\rangle$, it seems impossible to explicitly specify H_{final}. The key to resolve this difficulty is the observation that the ground state of H_{final} need not necessarily be the state $|\alpha(L)\rangle$. It is sufficient (under some mild restrictions) that the ground state has a nonnegligible inner product with $|\alpha(L)\rangle$. This gives us significant

[3] Otherwise, the first n gates can be used to flip the qubits to the desired input.

flexibility in designing H_{final}. Our idea is to use an ingenious, seemingly unrelated, result of Kitaev [22], in which he provides the first complete problem for the class QMA, a quantum analogue of NP. The problem he describes is known as the *local Hamiltonian* problem. His result can be viewed as the quantum analogue of the Cook–Levin theorem [28], which states that 3-SAT is NP-complete. For his proof, Kitaev defined a local Hamiltonian that checks the time propagation of a quantum circuit. Kitaev's local Hamiltonian has as its ground state the entire *history* of the quantum computation, in *superposition*:

$$|\eta\rangle := \frac{1}{\sqrt{L+1}} \sum_{\ell=0}^{L} |\alpha(\ell)\rangle \otimes |1^{\ell} 0^{L-\ell}\rangle^{c}. \tag{1}$$

The right (L qubits) register is a clock that counts the steps by adding 1s from left to right. The superscript c denotes clock qubits. We note that this state has a nonnegligible projection on our desired state $|\alpha(L)\rangle$. Hence, instead of designing a Hamiltonian that has the final unknown state of the circuit as its ground state, a task that seems impossible, we can define H_{final} to be Kitaev's local Hamiltonian. Why is it possible to define a local Hamiltonian whose ground state is $|\eta\rangle$, whereas the same task seems impossible with $|\alpha(L)\rangle$? The idea is that the unary representation of the clock enables a local verification of correct propagation of the computation from one step to the next, which cannot be done without the intermediate computational steps.

We thus choose Kitaev's Hamiltonian [22] to be our H_{final}. This Hamiltonian involves five body interactions (three clock particles and two computation particles). For the initial Hamiltonian H_{init} we require that it has $|\alpha(0)\rangle \otimes |0^L\rangle^c$, the first term in the history state, as its unique ground state. It is easy to define such a local Hamiltonian, because $|\alpha(0)\rangle \otimes |0^L\rangle^c$ is a tensor product state. Crucially, H_{init} and H_{final} can be constructed efficiently from the given quantum circuit; no knowledge of $|\alpha(L)\rangle$ is required for the construction.

A technical problem lies in showing that the spectral gap of the intermediate Hamiltonian $H(s)$ is lower-bounded by some inverse polynomial (more specifically, we show it is larger than $1/L^2$). To do this, we use a mapping of the Hamiltonian to a Markov chain corresponding to a random walk on the $L+1$ time steps. We then apply the conductance bound from the theory of rapidly mixing Markov chains [34] to bound the spectral gap of this chain. We note that, in general, applying the conductance bound requires knowing the limiting distribution of the chain, which in our case is hard since it corresponds to knowing the coefficients of the ground state for all the Hamiltonians $H(s)$. We circumvent this problem by noticing that it is actually sufficient in our case to know very little about the limiting distribution of the Markov chain, namely that it is monotone (in a certain sense to be defined later). This allows us to apply the conductance bound, and deduce that the spectral gap is $\Omega(1/L^2)$. From this it follows that the running time of the adiabatic computation is polynomial. Extracting the output of the quantum circuit from the history state efficiently is easy: Measure all the qubits of the clock and if the clock is in the state $|1^\ell\rangle$, the computational qubits carry the result of the circuit. Otherwise, start from scratch.[4]

The above scheme gives a proof of Theorem 1.1 that uses 5-local Hamiltonians, and runs in time roughly $O(L^5)$. The improvement to 3-locality is based on a simple

[4] This gives an overhead factor of L which can be avoided by adding $O(\frac{1}{\epsilon}L)$ identity gates to the quantum circuit at the end, which has the effect that most of the history state $|\eta\rangle$ is concentrated on the final state $|\alpha(L)\rangle$. See subsection 3.3 for more details.

idea (used in [20] to prove that the 3-local Hamiltonian problem is QMA-complete) but obtaining a lower bound on the spectral gap involves additional technical issues. We postpone its explanation to the body of the paper. The running time we achieve in this case is roughly $O(L^{14})$.

1.4. Proof of Theorem 1.3: Overview. The idea underlying the proof of Theorem 1.1 by itself does not suffice to prove Theorem 1.3. The basic problem lies in arranging sufficient interaction between the computational and clock particles, since if the particles are set on a grid, each clock particle can only interact with four neighbors. We circumvent this problem as follows. Instead of having separate clock and computational particles, we now assign to each particle both clock and computational degrees of freedom (this is what makes our particles six-state). We then construct a computation that propagates locally over the entire set of particles, snaking up and down each column of the lattice. The adiabatic evolution now ends up in the history state of this snake-like sequence of states.

The lower bound on the spectral gap is obtained in an essentially identical way as in the 3-local Hamiltonian case.

1.5. Related work. After the preliminary version of this paper appeared [2], the results regarding QMA-completeness were tightened by [21] to show that the 2-local Hamiltonian problem is QMA-complete. Following the ideas presented in the current paper, [21] used their result to show that Theorem 1.1 holds when the Hamiltonians are 2-local.

The idea to use an inverse polynomial spectral gap for fault tolerance is certainly not new. It is a crucial ingredient in topological (and later, geometrical) quantum computation [18, 23, 27]. Note, however, that in those models the spectral gap has no effect on the running time or on any other algorithmic aspects, and it is used only to separate the computational subspace from the "noisy" subspace. In contrast, the spectral gap in adiabatic computation is crucial from the algorithmic point of view, since it determines the time complexity of the computation.

1.6. Open questions. This paper demonstrates that quantum computation can be studied and implemented entirely within the adiabatic computation model, without losing its computational power. This result raises many open questions in various directions. First, it would be interesting to determine if the parameters presented in this work can be improved. For example, it might be possible to shorten the running time of our adiabatic simulation. Decreasing the dimensionality of the particles used in Theorem 1.3 from six to two or three might be important for implementation applications. An interesting question is whether Theorem 1.3 can be achieved using a one-dimensional instead of a two-dimensional grid.

Second, the possibility of fault tolerant adiabatic computation deserves to be studied both experimentally and theoretically. Since the publication of the preliminary version of the current paper [2], several researchers have begun to study adiabatic computation in the presence of noise [1, 31, 33]. However, it is still unclear whether adiabatic evolution might be helpful for the physical implementation of quantum computers.

Our results imply the equivalence between standard quantum computation and various other variants of adiabatic computation that have been considered in the literature and are more general than our model. These include adiabatic computation with a general path between H_{init} and H_{final}, rather than a straight line (see [4, 12] for a rigorous definition), and adiabatic computation with explicit sparse Hamiltonians [4] (see Corollary 1.2). A problem we leave open is to characterize the computational power of the adiabatic *optimization* model, the model in which the final Hamiltonian is assumed to be diagonal. The situation here is unclear. In fact, we believe that proving that this model can simulate arbitrary quantum computation is a very difficult task. The reason for this is that the output of an adiabatic optimization algorithm is the solution of a classical optimization problem and can thus be computed by a classical polynomial-time Turing machine with the help of an NP oracle. Hence, such a simulation would place BQP in the polynomial time hierarchy, thereby resolving a long-standing open question.

Finally, we hope that the adiabatic framework might lead to the discovery of new quantum algorithms (see [4] in this context). As shown in this paper, as well as in [4], tools from probability theory, mathematical physics, and spectral gap analysis might turn out to be relevant and useful. In order to improve our understanding of the benefits of the adiabatic paradigm, it might be insightful to see adiabatic versions of known quantum algorithms, presented in a meaningful way.

Organization. In section 2 we describe the model of adiabatic computation and state some relevant facts about Markov chains. Section 3 shows how adiabatic systems with five-local Hamiltonians can efficiently simulate standard quantum computations. Section 4 extends and improves this result, and in particular proves it using only three-local Hamiltonians. Section 5 shows how to adapt the construction to a two-dimensional grid.

2. Preliminaries.

2.1. Hamiltonians of n-particle systems.

For background on n-qubit systems, quantum circuits, and Hamiltonians, see [26]. An n-particle system is described by a state in Hilbert space of dimension d^n, the tensor product of n d-dimensional Hilbert spaces. For simplicity, we restrict our discussion in this subsection to quantum systems composed of 2-dimensional particles, i.e., qubits; a similar discussion holds for higher dimensional particles (such as the six-dimensional case we consider later).

In the standard model of quantum computation, the state of n qubits evolves in discrete time steps by unitary operations. In fact, the underlying physical description of this evolution is continuous, and is governed by Schrödinger's equation: $-i\frac{d}{dt}|\psi(t)\rangle = H(t)|\psi(t)\rangle$. Here $|\psi(t)\rangle$ is the state of the n qubits at time t, and $H(t)$ is a Hermitian $2^n \times 2^n$ matrix operating on the space of n qubits. This $H(t)$ is the *Hamiltonian* operating on a system; it governs the dynamics of the system. Given that the state of the system at time $t = 0$ is equal to $|\psi(0)\rangle$, one can in principle solve Schrödinger's equation with this initial condition to get $|\psi(T)\rangle$, the state of the system at a later time $t = T$. The fact that the Hamiltonian is Hermitian corresponds to the familiar fact that the discrete time evolution of the quantum state from time t_1 to a later time t_2 is unitary.

We sometimes refer to eigenvalues of Hamiltonians as *energies.* The *ground energy* of a Hamiltonian is its lowest eigenvalue and the corresponding eigenvector(s) are called *ground state*(s). We define $\Delta(H)$, the *spectral gap* of a Hamiltonian H, to be the difference between the lowest eigenvalue of H and its second lowest eigenvalue. ($\Delta(H) = 0$ if the lowest eigenvalue is degenerate, that is, has more than one eigenvector associated with it.) We define the *restriction* of H to some subspace $\mathcal{S}$, denoted $H_{\mathcal{S}}$, as $\Pi_{\mathcal{S}} H \Pi_{\mathcal{S}}$, where $\Pi_{\mathcal{S}}$ is the orthogonal projection on $\mathcal{S}$.

A Hamiltonian on an n-particle system represents a certain physical operation that one can, in principle, apply to an n-particle system. However, it is clear that one cannot efficiently apply any arbitrary Hamiltonian (even describing a Hamiltonian on

n qubits requires exponential space in the worst case). We say that a Hamiltonian H is *k-local* if H can be written as $\sum_A H^A$, where A runs over all subsets of k particles, and H^A operates trivially on all but the particles in A (i.e., it is a tensor product of a Hamiltonian on A with identity on the particles outside of A). Notice that for any constant k, a k-local Hamiltonian on n-qubits can be described by $2^{2k}n^k = \text{poly}(n)$ numbers. We say that H is local if H is k-local for some constant k.

In this paper we restrict our attention to k-local Hamiltonians. This requirement corresponds to the fact that all known interactions in nature involve a constant number of particles. We attempt to make k as small as possible to make the Hamiltonian presumably easier to implement.

2.2. The model of adiabatic computation. The cornerstone of the adiabatic model of computation is the celebrated adiabatic theorem [19, 25]. Consider a time-dependent Hamiltonian $H(s)$, $s \in [0,1]$, and a system initialized at time $t = 0$ in the ground state of $H(0)$ (here and in the following we assume that for all $s \in [0,1]$, $H(s)$ has a unique ground state). Let the system evolve according to the Hamiltonian $H(t/T)$ from time $t = 0$ to time T. We refer to such a process as an *adiabatic evolution according to H for time T*. The adiabatic theorem affirms that for large enough T the final state of the system is very close to the ground state of $H(1)$. Just how large T should be for this to happen is determined by the spectral gap of the Hamiltonians $H(s)$. Such an upper bound on T is given in the following theorem, adapted from [29] (whose proof in turn is based on [6]; see also [5] for a recent elementary proof of a slightly weaker version).

Theorem 2.1 (the adiabatic theorem (adapted from [29])). *Let H_{init} and H_{final} be two Hamiltonians acting on a quantum system and consider the time-dependent Hamiltonian $H(s) := (1-s)H_{\text{init}} + sH_{\text{final}}$. Assume that for all s, $H(s)$ has a unique ground state. Fix $\delta > 0$, and let*

$$T \geq \Omega\left(\frac{\|H_{\text{final}} - H_{\text{init}}\|^{1+\delta}}{\epsilon^{\delta}\min_{s\in[0,1]}\{\Delta^{2+\delta}(H(s))\}}\right). \tag{2}$$

Let $|\psi(T)\rangle$ be the solution at time $t = T$ for Schrödinger's equation $-i\frac{d}{dt}|\psi(t)\rangle = H(t/T)|\psi(t)\rangle$ with the initial state $|\psi(0)\rangle$ being the ground state of H_{init}. Then $|\psi(T)\rangle$ is ϵ-close in ℓ_2-norm to the ground state of H_{final} (with an appropriate setting of global phase). The matrix norm is the spectral norm $\|H\| := \max_{w\neq 0}\|Hw\|_2/\|w\|_2$.

One should think of δ as being some fixed constant, say 0.1. We cannot take $\delta = 0$ because of the constant hidden in the Ω notation, which goes to infinity as δ goes to 0.

Let us now describe the model of adiabatic computation. In this paper we use the following definition of adiabatic computation that slightly generalizes that of Farhi et al. [14]. The adiabatic "circuit" is determined by H_{init} and H_{final} and the output of the computation is (close to) the ground state of H_{final}.

Definition 2.2. *A k-local adiabatic computation $AC(n, d, H_{\text{init}}, H_{\text{final}}, \epsilon)$ is specified by two k-local Hamiltonians, H_{init} and H_{final} acting on n d-dimensional particles, such that both Hamiltonians have unique ground states. The ground state of H_{init} is a tensor product state. The output is a state that is ϵ-close in ℓ_2-norm to the ground state of H_{final}. Let T be the smallest time such that the final state of an adiabatic evolution according to $H(s) := (1-s)H_{\text{init}} + sH_{\text{final}}$ for time T is ϵ-close in ℓ_2-norm to the ground state of H_{final}. The running time of the adiabatic algorithm is defined to be $T \cdot \max_s \|H(s)\|$.*

Observe that we have chosen our definition of running time to be $T \cdot \max_s \|H(s)\|$ and not T. To see why this is the right notion of running time, notice that according to Schrödinger's equation, for any $c > 0$, the final state of a system that evolves according to some Hamiltonian $H(s)$ for time T is identical to that of a system that evolves according to some Hamiltonian $cH(s)$ for time T/c. In fact, this is a basic physical trade-off between energy and time. This trade-off can also be seen in Theorem 2.1: if both H_{init} and H_{final} are multiplied by some factor $c > 0$, the resulting bound on T gets divided by the same factor. Hence, one can achieve an arbitrarily small value of T by multiplying the Hamiltonians by some large factor. This clearly shows that T is not a meaningful notion of running time. On the other hand, our notion of running time is invariant under multiplication by a constant factor.

The right-hand side of (2) can be used to provide an upper bound on the running time of an adiabatic computation. Hence, in order to show that an adiabatic algorithm is efficient, it is enough to use Hamiltonians of at most poly(n) norm, and show that for all $s \in [0, 1]$ the spectral gap $\Delta(H(s))$ is at least inverse polynomial in n.

We note that in certain cases, it is possible to obtain a stronger upper bound on the running time. Indeed, assume there exists a subspace $\mathcal{S}$ such that for all $s \in [0, 1]$, $H(s)$ leaves $\mathcal{S}$ invariant, i.e., $H(s)(\mathcal{S}) \subseteq \mathcal{S}$. Equivalently, $H(s)$ is block diagonal in $\mathcal{S}$ and its orthogonal space $\mathcal{S}^{\perp}$. Consider $H_{\mathcal{S}}(s)$, the restriction of $H(s)$ to $\mathcal{S}$. Then, starting from a state inside $\mathcal{S}$, an adiabatic evolution according to H is *identical* to an adiabatic evolution according to $H_{\mathcal{S}}$ (this follows from Schrödinger's equation). Hence, we can potentially obtain a stronger upper bound by replacing $\Delta(H(s))$ with $\Delta(H_{\mathcal{S}}(s))$ in (2). This stronger upper bound will be used in our first adiabatic algorithm.

Finally, let us mention that one can define more general models of adiabatic computation. For example, one might consider nonlocal Hamiltonians (see [4]). Another possible extension is to consider more general paths between H_{init} and H_{final} (see, e.g., [4, 8, 12]). Obviously, our main results, such as Theorem 1.1, hold also for these more general models.

2.3. Markov chains and Hermitian matrices. Under certain conditions, there exists a standard mapping of Hamiltonians to Markov chains (for background on Markov chains, see [24]). The following fact is useful to show that this mapping applies in the case we analyze.

FACT 2.3 (adapted from Perron's theorem, Theorem 8.2.11 in [17]). *Let G be a Hermitian matrix with real nonnegative entries. If there exists a finite k such that all entries of G^k are positive, then G's largest eigenvalue is positive, and all other eigenvalues are strictly smaller in absolute value. Moreover, the corresponding eigenvector is unique, and all its entries are positive.*

We define the mapping for G, a Hermitian matrix operating on an $L+1$-dimensional Hilbert space. Suppose that all the entries of G are real and nonnegative, that its eigenvector $(\alpha_0, \dots, \alpha_L)$ with largest eigenvalue μ satisfies $\alpha_i > 0$ for all $0 \leq i \leq L$, and that $\mu > 0$. Define P by

$$P_{ij} := \frac{\alpha_j}{\mu \alpha_i} G_{ij}. \tag{3}$$

The matrix P is well defined, and is stochastic because all its entries are nonnegative and each of its rows sums up to one. It is easy to verify the following fact.

FACT 2.4. *The vector $(v_0, \dots, v_L)$ is an eigenvector of G with eigenvalue δ if and only if $(\alpha_0 v_0, \dots, \alpha_L v_L)$ is a left eigenvector of P with eigenvalue δ/μ.*

We will consider G of the form $G = I - H$ for some Hamiltonian H. The above fact implies that if $(\alpha_0, \ldots, \alpha_L)$ is the ground state of H with eigenvalue λ, then $(\alpha_0^2, \ldots, \alpha_L^2)$ is a left eigenvector of P with maximal eigenvalue 1. By normalizing, we obtain that $\pi := (\alpha_0^2/Z, \ldots, \alpha_L^2/Z)$ is the limiting distribution of P, where $Z = \sum \alpha_i^2$. Moreover, the gap between P's largest and second largest eigenvalues is equal to $\Delta(H)/(1-\lambda)$.

2.4. Spectral gaps of Markov chains. Given a stochastic matrix P with limiting distribution π, and a subset $B \subseteq \{0, \ldots, L\}$, the *flow* from B is given by $F(B) := \sum_{i \in B, j \notin B} \pi_i P_{ij}$. Define the π-weight of B as $\pi(B) := \sum_{i \in B} \pi_i$. The *conductance* of P is defined by $\varphi(P) := \min_B F(B)/\pi(B)$, where we minimize over all nonempty subsets $B \subseteq \{0, \ldots, L\}$ with $\pi(B) \leq \frac{1}{2}$.

THEOREM 2.5 (the conductance bound [34]). *The eigenvalue gap of P is at least $\frac{1}{2}\varphi(P)^2$.*

3. Equivalence of adiabatic and quantum computation. Here we prove Theorem 1.1 by showing how to simulate a quantum circuit consisting of L two-qubit gates on n qubits by an adiabatic computation on $n + L$ qubits (the other direction was shown in [9,14]). We allow five-qubit interactions; this will be improved to three-qubit interactions in the next section. Theorem 1.1 thus follows as a corollary from the following theorem.

THEOREM 3.1. *Given a quantum circuit on n qubits with L two-qubit gates implementing a unitary U, and $\epsilon > 0$, there exists a 5-local adiabatic computation $AC(n+L, 2, H_{\text{init}}, H_{\text{final}}, \epsilon)$ whose running time is* $\text{poly}(L, \frac{1}{\epsilon})$ *and whose output (after tracing out some ancilla qubits) is ϵ-close (in trace distance) to $U|0^n\rangle$. Moreover, H_{init} and H_{final} can be computed by a polynomial time Turing machine.*

The running time we obtain here is $O(\epsilon^{-(5+3\delta)} L^{5+2\delta})$ for any fixed $\delta > 0$.

3.1. The Hamiltonian. For our construction we use the Hamiltonian defined in [22]. Denote $|\gamma_\ell\rangle := |\alpha(\ell)\rangle \otimes |1^\ell 0^{L-\ell}\rangle^c$, where $|\alpha(\ell)\rangle$ denotes the state of the circuit after the ℓth gate and the superscript c denotes the clock qubits. We would like to define a local Hamiltonian H_{init} with ground state $|\gamma_0\rangle = |0^n\rangle \otimes |0^L\rangle^c$, and a local Hamiltonian H_{final} with ground state $|\eta\rangle = \frac{1}{\sqrt{L+1}} \sum_{\ell=0}^{L} |\gamma_\ell\rangle$ as in (1). To do this, we write H_{init} and H_{final} as a sum of terms:

$$H_{\text{init}} := H_{\text{clockinit}} + H_{\text{input}} + H_{\text{clock}},$$
$$H_{\text{final}} := \frac{1}{2} \sum_{\ell=1}^{L} H_\ell + H_{\text{input}} + H_{\text{clock}}.$$

The terms in H_{final} (and likewise in H_{init}) are defined such that the only state whose energy (i.e., eigenvalue) is 0 is the desired ground state. This is done by assigning an *energy penalty* to any state that does not satisfy the required properties of the ground state. The different terms, which correspond to different properties of the ground states, are described in the following paragraphs. The adiabatic evolution then follows the time-dependent Hamiltonian

$$H(s) = (1-s)H_{\text{init}} + sH_{\text{final}}. \quad (4)$$

Notice that as s goes from 0 to 1, $H_{\text{clockinit}}$ is slowly replaced by $\frac{1}{2}\sum_{\ell=1}^{L} H_\ell$ while H_{input} and H_{clock} are held constant.

We now describe each of the terms. First, H_{clock} checks that the clock's state is of the form $|1^\ell 0^{L-\ell}\rangle^c$ for some $0 \le \ell \le L$. This is achieved by assigning an energy penalty to any basis state on the clock qubits that contains the sequence 01,

$$H_{\text{clock}} := \sum_{\ell=1}^{L-1} |01\rangle\langle 01|^c_{\ell,\ell+1},$$

where the subscript indicates which clock qubits the projection operates on. Note that illegal clock states are eigenstates of H_{clock} with an eigenvalue of at least 1; legal clock states have an eigenvalue of 0.

Next, H_{input} checks that if the clock is $|0^L\rangle^c$, the computation qubits must be in the state $|0^n\rangle$,

$$H_{\text{input}} := \sum_{i=1}^{n} |1\rangle\langle 1|_i \otimes |0\rangle\langle 0|^c_1.$$

We complete the description of H_{init} with $H_{\text{clockinit}}$ whose goal is to check that the clock's state is $|0^L\rangle^c$,

$$H_{\text{clockinit}} := |1\rangle\langle 1|^c_1.$$

CLAIM 3.2. *The state $|\gamma_0\rangle$ is a ground state of H_{init} with an eigenvalue of* 0.[5]

Proof. It is easy to verify that $H_{\text{init}}|\gamma_0\rangle = 0$. As a sum of projectors, H_{init} is positive semidefinite and hence $|\gamma_0\rangle$ is a ground state of H_{init}. □

We now proceed to the first term in H_{final}. The Hamiltonian H_ℓ checks that the propagation from step $\ell - 1$ to ℓ is correct, i.e., that it corresponds to the application of the gate U_ℓ. For $1 < \ell < L$, it is defined as

$$\begin{aligned} H_\ell := I \otimes |100\rangle\langle 100|^c_{\ell-1,\ell,\ell+1} - U_\ell \otimes |110\rangle\langle 100|^c_{\ell-1,\ell,\ell+1} \\ -U_\ell^\dagger \otimes |100\rangle\langle 110|^c_{\ell-1,\ell,\ell+1} + I \otimes |110\rangle\langle 110|^c_{\ell-1,\ell,\ell+1}. \end{aligned} \tag{5}$$

Intuitively, the three-qubit terms above move the state of the clock one step forward, one step backward, or leave it unchanged. The accompanying matrices $U_\ell, U_\ell^\dagger$ describe the associated time evolution. For the boundary cases $\ell = 1, L$, we omit one clock qubit from these terms and define

$$\begin{aligned} H_1 &:= I \otimes |00\rangle\langle 00|_{1,2} - U_1 \otimes |10\rangle\langle 00|_{1,2} - U_1^\dagger \otimes |00\rangle\langle 10|_{1,2} + I \otimes |10\rangle\langle 10|_{1,2}, \\ H_L &:= I \otimes |10\rangle\langle 10|_{L-1,L} - U_L \otimes |11\rangle\langle 10|_{L-1,L} - U_L^\dagger \otimes |10\rangle\langle 11|_{L-1,L} + I \otimes |11\rangle\langle 11|_{L-1,L}. \end{aligned} \tag{6}$$

CLAIM 3.3. *The history state $|\eta\rangle$ is a ground state of H_{final} with an eigenvalue of* 0.

Proof. It is easy to verify that $H_{\text{final}}|\eta\rangle = 0$. It remains to be noticed that for all $1 \le \ell \le L$, H_ℓ is positive semidefinite and hence so is H_{final}. □

[5] The state $|\gamma_0\rangle$ is in fact the *unique* ground state of H_{init} as will become apparent from the proof of the spectral gap. A similar statement holds for Claim 3.3.

3.2. Spectral gap in a subspace. Let $\mathcal{S}_0$ be the $L+1$-dimensional subspace spanned by $|\gamma_0\rangle, \dots, |\gamma_L\rangle$. It is easy to verify the following claim.

CLAIM 3.4. *The subspace $\mathcal{S}_0$ is invariant under $H(s)$, i.e., $H(s)(\mathcal{S}_0) \subseteq \mathcal{S}_0$.*

In this subsection, we show that the spectral gap of $H_{\mathcal{S}_0}(s)$, the restriction of H to $\mathcal{S}_0$, is inverse polynomial in L. As mentioned in subsection 2.2, this, together with Claim 3.4, is enough to obtain a bound on the running time of the adiabatic algorithm.

Remark. Notice that both H_{clock} and H_{input} are 0 on the invariant subspace $\mathcal{S}_0$. This means that Theorem 3.1 holds even if we remove the terms H_{clock} and H_{input} from both H_{init} and H_{final}. We include these terms in the Hamiltonian for the sake of the presentation, and for consistency with the rest of the paper.

LEMMA 3.5. *The spectral gap of the restriction of $H(s)$ to $\mathcal{S}_0$ satisfies $\Delta(H_{\mathcal{S}_0}(s)) = \Omega(L^{-2})$ for all $s \in [0,1]$.*

Proof. Let us write the Hamiltonians $H_{\mathcal{S}_0,\text{init}}$ and $H_{\mathcal{S}_0,\text{final}}$ in the basis $|\gamma_0\rangle, \dots, |\gamma_L\rangle$ of $\mathcal{S}_0$. Both H_{clock} and H_{input} are 0 on $\mathcal{S}_0$ and can thus be ignored. We have the following $(L+1) \times (L+1)$ matrices:

$$H_{\mathcal{S}_0,\text{init}} = \begin{pmatrix} 0 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}, \tag{7}$$

$$\begin{aligned} H_{\mathcal{S}_0,\text{final}} &= \tfrac{1}{2}|\gamma_0\rangle\langle\gamma_0| - \tfrac{1}{2}|\gamma_0\rangle\langle\gamma_1| - \tfrac{1}{2}|\gamma_L\rangle\langle\gamma_{L\text{-}1}| + \tfrac{1}{2}|\gamma_L\rangle\langle\gamma_L| \\ &\quad + \sum_{\ell=1}^{L-1}(-\tfrac{1}{2}|\gamma_\ell\rangle\langle\gamma_{\ell-1}| + |\gamma_\ell\rangle\langle\gamma_\ell| - \tfrac{1}{2}|\gamma_\ell\rangle\langle\gamma_{\ell+1}|) \\ &= \begin{pmatrix} \frac{1}{2} & \text{-}\frac{1}{2} & 0 & & \cdots & & 0 \\ \text{-}\frac{1}{2} & 1 & \text{-}\frac{1}{2} & 0 & \ddots & & \vdots \\ 0 & \text{-}\frac{1}{2} & 1 & \text{-}\frac{1}{2} & 0 & \ddots & \vdots \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ \vdots & & 0 & \text{-}\frac{1}{2} & 1 & \text{-}\frac{1}{2} & 0 \\ & & & 0 & \text{-}\frac{1}{2} & 1 & \text{-}\frac{1}{2} \\ 0 & & \cdots & & 0 & \text{-}\frac{1}{2} & \frac{1}{2} \end{pmatrix}. \end{aligned} \tag{8}$$

We now lower bound $\Delta(H_{\mathcal{S}_0}(s))$. We consider two cases.

The case $s < 1/3$. Here, $H_{\mathcal{S}_0}(s)$ is sufficiently close to $H_{\mathcal{S}_0,\text{init}}$ (whose spectral gap is 1) so we can apply the following standard lemma (see, e.g., [7, page 244]).

LEMMA 3.6 (Gerschgorin's circle theorem). *Let A be any matrix with entries a_{ij}. Consider the discs in the complex plane given by*

$$D_i = \Big\{ z \mid |z - a_{ii}| \le \sum_{j \ne i} |a_{ij}| \Big\}, \quad 1 \le i \le n.$$

Then the eigenvalues of A are contained in $\cup D_i$ and any connected component of $\cup D_i$ contains as many eigenvalues of A as the number of discs that form this component.

For $s < 1/3$, we have that $H_{\mathcal{S}_0}(s)_{1,1} < 1/6$, and $\sum_{j \ne 1} H_{\mathcal{S}_0}(s)_{1,j} < 1/6$. Moreover, for any $i \ne 1$, we have that $H_{\mathcal{S}_0}(s)_{i,i} > 5/6$, and $\sum_{j \ne i} H_{\mathcal{S}_0}(s)_{i,j} < 1/6$. By the above lemma, we obtain that there is one eigenvalue smaller than 1/3 while all other eigenvalues are larger than 2/3. Hence, the spectral gap is at least 1/3.

The case $s \geq 1/3$. We note that $H_{\mathcal{S}_0,\text{final}}$ is the Laplacian of the simple random walk [24] of a particle on a line of length $L+1$. A standard result in Markov chain theory implies $\Delta(H_{\mathcal{S}_0,\text{final}}) = \Omega(1/L^2)$ [24]. For $s \geq 1/3$, $H_{\mathcal{S}_0}(s)$ is sufficiently close to $H_{\mathcal{S}_0,\text{final}}$ to apply Markov chain techniques, as we show next.

Let $(\alpha_0, \ldots, \alpha_L)^\dagger$ be the ground state of $H_{\mathcal{S}_0}(s)$ with eigenvalue λ. Define the Hermitian matrix $G(s) = I - H_{\mathcal{S}_0}(s)$. It is easy to see that $G(s)$ satisfies the conditions of Fact 2.3 for all $s > 0$. We obtain that the largest eigenvalue $\mu = 1 - \lambda$ of $G(s)$ is positive and nondegenerate and the corresponding eigenvector $(\alpha_0, \ldots, \alpha_L)^\dagger$ has positive entries. We can now map the matrix $G(s)$ to a stochastic matrix $P(s)$ as described in subsection 2.3. The transition matrix $P(s)$ describes a random walk on the line of $L+1$ sites (see Figure 1). Fact 2.4 implies that the limiting distribution of $P(s)$ is given by $\pi = (\alpha_0^2/Z, \ldots, \alpha_L^2/Z)$, where $Z = \sum_i \alpha_i^2$.

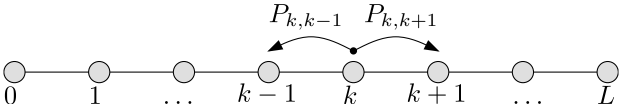


FIG. 1. *The random walk of* $P(s)$.

We bound the spectral gap of $P(s)$ using the conductance bound (see subsection 2.4). To do this we need to know that π is monotone. We first show the following claim.

CLAIM 3.7. *For all* $0 \leq s \leq 1$, *the ground state of* $H_{\mathcal{S}_0}(s)$ *is monotone, namely* $\alpha_0 \geq \alpha_1 \geq \cdots \geq \alpha_L \geq 0$.

Proof. The case $s = 0$ is obvious, so assume $s > 0$. We first claim that the ground state $(\alpha_0, \ldots, \alpha_L)^\dagger$ of $H_{\mathcal{S}_0}(s) = I - G(s)$ can be written as the limit

$$\frac{1}{c_0} \lim_{\ell \to \infty} (G(s)/\mu)^\ell (1, \ldots, 1)^\dagger$$

for some constant $c_0 > 0$. To see this, let $|v_0\rangle, \ldots, |v_L\rangle$ be an orthonormal set of eigenvectors of $G(s)$, with corresponding eigenvalues $\mu_0 \geq \mu_1 \geq \cdots \geq \mu_L$. By Fact 2.3, the largest eigenvalue corresponds to a unique eigenvector, and hence we have $|v_0\rangle = (\alpha_0, \ldots, \alpha_L)^\dagger$, and $\mu_0 = \mu$.

The set of eigenvectors $|v_i\rangle$ forms an orthonormal basis, and we can write $(1, \ldots, 1)^\dagger$ in terms of this basis: $(1, \ldots, 1)^\dagger = \sum_i c_i |v_i\rangle$. Now, we have that $(G(s)/\mu)^\ell (1, \ldots, 1)^\dagger = \sum_i c_i (\frac{\mu_i}{\mu})^\ell |v_i\rangle$. By Fact 2.3 we have $|\mu_i| < \mu$ for all $i \neq 0$, and $\mu > 0$. We thus have that $\lim_{\ell \to \infty} (G(s)/\mu)^\ell (1, \ldots, 1)^\dagger = c_0 |v_0\rangle$.

It is easy to check that $G(s)$ preserves monotonicity, namely, if $G(s)$ is applied to a monotone vector, the result is a monotone vector. Hence, when $G(s)/\mu$ is applied to the monotone vector $(1, \ldots, 1)^\dagger$, the result is a monotone vector. Thus, $c_0 |v_0\rangle$ is monotone. Finally, we observe that $c_0 > 0$. This is because c_0 is the inner product between the all 1 vector, and $|v_0\rangle$, whose entries are all positive by Fact 2.3. This implies that $|v_0\rangle$ is also monotone, as desired. □

It follows that π is also monotone. We use this and simple combinatorial arguments to prove the following claim.

CLAIM 3.8. *For all* $1/3 \leq s \leq 1$, $\varphi(P(s)) \geq \frac{1}{6L}$.

Proof. We show that for any nonempty $B \subseteq \{0,\dots,L\}$, $F(B)/\pi(B) \geq \frac{1}{6L}$. We consider two cases. First, assume that $0 \in B$. Let k be the smallest such that $k \in B$ but $k+1 \notin B$. Then,

$$F(B) \geq \pi_k P(s)_{k,k+1} = \pi_k \cdot \frac{\sqrt{\pi_{k+1}}}{\mu\sqrt{\pi_k}} G(s)_{k,k+1} = \frac{\sqrt{\pi_k \pi_{k+1}}}{1-\lambda} G(s)_{k,k+1} \geq \frac{\pi_{k+1}}{1-\lambda} G(s)_{k,k+1},$$

where the last inequality follows from the monotonicity of π. Using the definition of G and the assumption that $s \geq 1/3$ we get that $G(s)_{k,k+1} \geq 1/6$. We also have $0 < 1-\lambda \leq 1$, where the second inequality follows from the fact that $H_{\mathcal{S}_0}(s)$ is positive semidefinite, and the first follows from $\mu > 0$ which we previously deduced from Fact 2.3. Hence,

$$\frac{F(B)}{\pi(B)} \geq \frac{\pi_{k+1}}{6\pi(B)}. \tag{9}$$

By $\pi(B) \leq 1/2$, we have $\pi(\{k+1,\dots,L\}) \geq 1/2$. Together with $\pi(\{k+1,\dots,L\}) \leq L\pi_{k+1}$ we obtain $\pi_{k+1} \geq 1/(2L)$. This yields the desired bound $F(B)/\pi(B) \geq 1/(6L)$.

Now assume that $0 \notin B$ and let k be the smallest such that $k \notin B$ and $k+1 \in B$. It is easy to see that $\pi_k P(s)_{k,k+1} = \pi_{k+1} P(s)_{k+1,k}$. Hence, using the same argument as before we can see that (9) holds in this case too. Since $B \subseteq \{k+1,\dots,L\}$, we have $\pi(\{k+1,\dots,L\}) \geq \pi(B)$. Hence, $\pi_{k+1} \geq \pi(B)/L$. Again, this yields the bound $F(B)/\pi(B) \geq 1/(6L)$. □

By Theorem 2.5, we have that the spectral gap of $P(s)$ is larger than $1/(2\cdot(6)^2\cdot L^2)$. By subsection 2.3, we have that $\Delta(H_{\mathcal{S}_0}) \geq \mu/(2\cdot(6)^2 L^2)$. Finally, notice that $\mu = 1-\lambda \geq \frac{1}{2}$, because $\lambda \leq \langle\gamma_0|H_{\mathcal{S}_0}(s)|\gamma_0\rangle = \frac{s}{2} \leq \frac{1}{2}$. □

3.3. Running time. We now complete the proof of Theorem 3.1. Note that we have already proved something which is very close to Theorem 3.1.

CLAIM 3.9. *Given a quantum circuit on n qubits with L gates, the adiabatic algorithm with H_{init} and H_{final} as defined in the previous section, with $T = O(\epsilon^{-\delta}L^{4+2\delta})$ for some fixed $\delta > 0$, outputs a final state that is within ℓ_2-distance ϵ of the history state of the circuit, $|\eta\rangle$. The running time of the algorithm is $O(T \cdot L)$.*

Proof. Claim 3.4 shows that $\mathcal{S}_0$ is invariant under H. Hence, as mentioned in subsection 2.2, an adiabatic evolution according to H is identical to an adiabatic evolution according to $H_{\mathcal{S}_0}$. Using Lemma 3.5 and Theorem 2.1 (with $\|H_{\text{init}} - H_{\text{final}}\| = O(1)$), we obtain that for T as above the final state (with global phase adjusted appropriately) is indeed ϵ-close in ℓ_2-norm to $|\eta\rangle$. By our definition, the running time of the adiabatic algorithm is $O(T\cdot L)$ since $\|H(s)\| \leq (1-s)\|H_{\text{init}}\| + s\|H_{\text{final}}\| = O(L+n) = O(L)$. The last equality follows from $n = O(L)$, because each qubit is assumed to participate in the computation (otherwise we can omit it). □

In fact, one might be satisfied with this claim, which enables generating adiabatically a state which is very close to $|\eta\rangle$, instead of our desired $|\alpha(L)\rangle$. To see why this might be sufficient to simulate quantum circuits, suppose for a moment that ϵ is 0, and the final state is exactly $|\eta\rangle$. As mentioned in the introduction, we can now measure the clock qubits of the history state, and with probability $1/L$ the outcome is $\ell = L$, which means that the state of the first register is the desired state $|\alpha(L)\rangle$. If the measurement yields another value, we repeat the adiabatic algorithm from scratch. To get $\ell = L$ with sufficiently high probability, we repeat the process $O(L)$ times, which introduces an overhead factor of L. The above discussion is also true with $\epsilon > 0$, as long as it is much smaller than $1/L$, the weight of $|\alpha(L)\rangle$ in $|\eta\rangle$.

However, strictly speaking, this is not sufficient to complete the proof of Theorem 3.1. Indeed, the theorem as stated follows our definition of the model of adiabatic computation, which allows one to perform one adiabatic evolution and then measure (and possibly trace out some qubits). Classical postprocessing such as conditioning on ℓ being equal to L, and repeating the computation if it is not, are not allowed. Hence, we need to adiabatically generate a state that is close to the final state of the circuit, $|\alpha(L)\rangle$.

This technical issue can be resolved with the following simple trick, which at the same time allows us to avoid the overhead factor of L introduced before. We simply add another $O(\frac{1}{\epsilon}L)$ identity gates to the original quantum circuit at the end of its computation. This modification ensures that the history state (after tracing out the clock qubits) is close to $|\alpha(L)\rangle$. We then apply the adiabatic simulation to this modified circuit. The following easy lemma makes this precise.

Lemma 3.10. *Assume we can transform any given quantum circuit with L two-qubit gates on n qubits into a k-local adiabatic computation on $n+L$ d-dimensional particles whose output is ϵ close in ℓ_2-norm to the* history state *of the quantum circuit and whose running time is $f(L,\epsilon)$ for some function f. Then, we can transform any given quantum circuit with L two-qubit gates on n qubits into a k-local adiabatic computation on $n+2L/\epsilon$ d-dimensional particles whose output (after tracing out some ancilla qubits) is ϵ close in trace distance to the* final state *of the circuit and whose running time is $f(2L/\epsilon, \epsilon/2)$.*

Proof. Given a quantum circuit on n qubits with L gates, consider the circuit obtained by appending to it $(\frac{2}{\epsilon}-1)L$ identity gates. Let $L' = 2L/\epsilon$ be the number of gates in the modified circuit and let $|\eta\rangle$ denote its history state. By our assumption, we can transform this modified circuit into an adiabatic computation whose output is $\epsilon/2$ close in ℓ_2-norm to $|\eta\rangle$ and whose running time is $f(L', \epsilon/2)$. Since the trace distance between two pure states is bounded from above by the ℓ_2-distance (see, e.g., [3]), we obtain that the output of the adiabatic computation is also $\epsilon/2$ close in trace distance to $|\eta\rangle\langle\eta|$. In addition, it is easy to check that after we trace out the clock qubits from $|\eta\rangle$, we are left with a state that is $\epsilon/2$ close in trace distance to the final state of the circuit. We complete the proof by applying the triangle inequality. □

We can now apply this lemma on the result of Claim 3.9. This completes the proof of Theorem 3.1, with the running time being $O(\epsilon^{-(5+3\delta)}L^{5+2\delta})$.

4. Improvements and extensions. In this section we present two extensions of the result of the previous section. We note that the techniques developed here will be used again in section 5, where we impose additional geometrical constraints on the system.

The first result is presented in subsection 4.1. It is related to the spectral gap of the Hamiltonian $H(s)$ used in section 3. We showed there that inside a preserved subspace $\mathcal{S}_0$ this Hamiltonian has a nonnegligible spectral gap. This was enough for the proof of Theorem 1.1 since the entire adiabatic evolution is performed inside this subspace. Here, we extend this result and show that the spectral gap of $H(s)$ in the *entire* Hilbert space is also nonnegligible. The existence of a nonnegligible spectral gap in the entire Hilbert space might have some relevance when dealing with adiabatic computation in the presence of noise (see, e.g., [8]).

The second result is given in subsection 4.2. There, we show that Theorem 1.1 holds with 3-local Hamiltonians (rather than 5). The proof of this result uses techniques developed in the above spectral gap proof together with some new tools.

4.1. Spectral gap.

LEMMA 4.1. *For all* $0 \leq s \leq 1$, $\Delta(H(s)) = \Omega(L^{-3})$.

Proof. Let $\mathcal{S}$ be the subspace of dimension $(L+1) \cdot 2^n$ spanned by all legal clock states. Observe that $\mathcal{S}$ is preserved by $H(s)$, i.e., $H(s)(\mathcal{S}) \subseteq \mathcal{S}$. Hence, the eigenstates of $H(s)$ belong either to $\mathcal{S}$ or to its orthogonal subspace $\mathcal{S}^{\perp}$. We can therefore analyze the spectrum of $H_{\mathcal{S}}(s)$ and of $H_{\mathcal{S}^{\perp}}(s)$ separately.

First, due to the term H_{clock} and the fact that all other terms are positive semidefinite, the ground energy of $H_{\mathcal{S}^{\perp}}(s)$ is at least 1. Second, as we will show next using Lemma 4.2, the spectral gap of $H_{\mathcal{S}}(s)$ is $\Omega(L^{-3})$. To establish the same spectral gap for $H(s)$, it is enough to show that the ground energy of $H_{\mathcal{S}}(s)$ is smaller than $\frac{1}{2}$, which would mean that the spectral gap of $H(s)$ is exactly that of $H_{\mathcal{S}}(s)$. Indeed, observe that

$$\langle \gamma_0 | H_{\mathcal{S}}(s) | \gamma_0 \rangle = \langle \gamma_0 | H_{\mathcal{S}_0}(s) | \gamma_0 \rangle = s/2 \leq 1/2,$$

where the first equality holds because $|\gamma_0\rangle \in \mathcal{S}_0$ and the second follows from (7) and (8). Therefore, the smallest eigenvalue of $H_{\mathcal{S}}(s)$ is bounded from above by $1/2$. □

LEMMA 4.2. *Let* $\mathcal{S}$ *denote the subspace spanned by all legal clock states. Then the ground state of* $H_{\mathcal{S}}(0)$ *is* $|\gamma_0\rangle$, *and that of* $H_{\mathcal{S}}(1)$ *is* $|\eta\rangle$. *Moreover, for all* $0 \leq s \leq 1$, $\Delta(H_{\mathcal{S}}(s)) = \Omega(L^{-3})$.

Proof. We can write $\mathcal{S}$ as the direct sum of 2^n orthogonal subspaces $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_{2^n-1}$, defined as follows. For $0 \leq j \leq 2^n - 1$ and $0 \leq \ell \leq L$, define $|\gamma_\ell^j\rangle := |\alpha^j(\ell)\rangle \otimes |1^\ell 0^{L-\ell}\rangle$, where $|\alpha^j(\ell)\rangle$ is the state of the quantum circuit at time ℓ if the input state corresponds to the binary representation j. Note that $|\gamma_\ell^0\rangle = |\gamma_\ell\rangle$. The space $\mathcal{S}_j$ is spanned by $\{|\gamma_0^j\rangle, \ldots, |\gamma_L^j\rangle\}$. It is easy to check the following claim (see Figure 2).

CLAIM 4.3. *The Hamiltonian* $H_{\mathcal{S}}(s)$ *is block diagonal in the* $\mathcal{S}_j$*'s.*

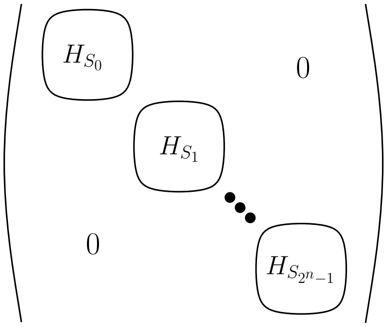


FIG. 2. *$H_{\mathcal{S}}(s)$ is block diagonal.*

By Claims 3.2, 3.3, and 4.3, and Lemma 3.5, it suffices to argue that the ground energy of $H_{\mathcal{S}_j}(s)$ for any $j \neq 0$ is larger than the ground energy of $H_{\mathcal{S}_0}(s)$ by at least $\Omega(1/L^3)$. Essentially, this follows from the penalty given by the term H_{input} to nonzero input states. The proof, however, is slightly subtle since H_{input} assigns a penalty only to states $|\gamma_\ell^j\rangle$ with $\ell = 0$.

Notice that

$$H_{\mathcal{S}_j}(s) = H_{\mathcal{S}_0}(s) + H_{\mathcal{S}_j,\text{input}}.$$

Moreover, for $1 \leq j \leq 2^n - 1$, $H_{\mathcal{S}_j,\text{input}}$ is diagonal, with its top-left element at least 1 (it actually equals the number of 1s in the binary representation of j) and all other diagonal elements zero. Hence, if we define M as

$$M := \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix},$$

then $H_{\mathcal{S}_j,\text{input}} - M$ is positive definite and therefore we can lower bound the ground energy of $H_{\mathcal{S}_j}(s)$ with the ground energy of $H_{\mathcal{S}_0}(s) + M$. For this, we apply the following geometrical lemma by Kitaev (Lemma 14.4 in [22]).

Lemma 4.4. *Let H_1, H_2 be two Hamiltonians with ground energies a_1, a_2, respectively. Suppose that for both Hamiltonians the difference between the energy of the (possibly degenerate) ground space and the next highest eigenvalue is larger than Λ, and that the angle between the two ground spaces is θ. Then the ground energy of $H_1 + H_2$ is at least $a_1 + a_2 + 2\Lambda \sin^2(\theta/2)$.*

We now apply this lemma to $H_{\mathcal{S}_0}(s)$ and M. By Lemma 3.5, the spectral gap of $H_{\mathcal{S}_0}(s)$ is $\Omega(1/L^2)$. The spectral gap of M is clearly 1. Moreover, using Claim 3.7, we obtain that the angle between the two ground spaces satisfies $\cos(\theta) \leq 1 - 1/L$ by the monotonicity property of the ground state of $H_{\mathcal{S}_0}(s)$ (see Claim 3.7). It follows that the ground energy of $H_{\mathcal{S}_j}(s)$ is higher by at least $\Omega(1/L^3)$ than that of $H_{\mathcal{S}_0}(s)$. □

Remark. Notice that we only used the following properties of H_{input}: its restriction to $\mathcal{S}_0$ is 0 and its restriction to $\mathcal{S}_j$ for any $j \neq 0$ is a diagonal matrix in the basis $|\gamma_0^j\rangle, \dots, |\gamma_L^j\rangle$ whose top-left entry is at least 1 and all other entries are nonnegative. This observation will be useful in section 5.

4.2. Three-local Hamiltonian.

4.2. Three-local Hamiltonian. We now show that adiabatic computation with 3-local Hamiltonians is sufficient to simulate standard quantum computations.

Theorem 4.5. *Given a quantum circuit on n qubits with L two-qubit gates implementing a unitary U, and $\epsilon > 0$, there exists a 3-local adiabatic computation $AC(n + L, 2, H_{\text{init}}, H_{\text{final}}, \epsilon)$ whose running time is $\text{poly}(L, \frac{1}{\epsilon})$ and whose output state is ϵ-close (in trace distance) to $U|0^n\rangle$. Moreover, H_{init} and H_{final} can be computed by a polynomial time Turing machine.*

The proof of this theorem builds on techniques developed in previous subsections.

4.2.1. The Hamiltonian. Consider the Hamiltonian constructed in subsection 3.1. Notice that all terms except H_ℓ are already 3-local (some are even 2-local or 1-local). In order to obtain a 3-local Hamiltonian, we remove two clock qubits from the 5-local terms in H_ℓ and leave only the ℓth clock qubit. More precisely, for $1 < \ell < L$ define

$$H'_\ell := I \otimes |100\rangle\langle 100|^c_{\ell-1,\ell,\ell+1} - U_\ell \otimes |1\rangle\langle 0|^c_\ell - U_\ell^\dagger \otimes |0\rangle\langle 1|^c_\ell + I \otimes |110\rangle\langle 110|^c_{\ell-1,\ell,\ell+1}.$$

For the boundary cases $l = 1, L$ we define

$$H'_1 := I \otimes |00\rangle\langle 00|^c_{1,2} - U_1 \otimes |1\rangle\langle 0|^c_1 - U_1^\dagger \otimes |0\rangle\langle 1|^c_1 + I \otimes |10\rangle\langle 10|^c_{1,2},$$
$$H'_L := I \otimes |10\rangle\langle 10|^c_{L-1,L} - U_\ell \otimes |1\rangle\langle 0|^c_L - U_L^\dagger \otimes |0\rangle\langle 1|^c_L + I \otimes |11\rangle\langle 11|^c_{L-1,L}.$$

Note that because of the terms $|1\rangle\langle 0|^c$ and $|0\rangle\langle 1|^c$, these Hamiltonians no longer leave the subspace $\mathcal{S}$ invariant. To mend this, we assign a much larger energy penalty to illegal clock states. As we will see soon, this makes the lower part of the spectrum of our Hamiltonians behave essentially like in their restriction to $\mathcal{S}$. Set $J = \epsilon^{-2}L^6$ and define

$$H'_{\text{init}} := H_{\text{clockinit}} + H_{\text{input}} + J \cdot H_{\text{clock}},$$
$$H'_{\text{final}} := \frac{1}{2}\sum_{\ell=1}^{L} H'_\ell + H_{\text{input}} + J \cdot H_{\text{clock}}.$$

The Hamiltonian we use here is thus

$$H'(s) = (1-s)H'_{\text{init}} + sH'_{\text{final}}.$$

Essentially the same proof as that of Claim 3.2 shows that $|\gamma_0\rangle$ is a ground state of H'_{init}. However, it turns out that $|\eta\rangle$ is no longer a ground state of H'_{final} (the proof of Claim 3.3 does not apply since H'_ℓ is no longer positive semidefinite). However, as we shall see later, $|\eta\rangle$ is very close to the ground state of H'_{final}.

4.2.2. The spectral gap. Our first claim is that, when restricted to $\mathcal{S}$, H' and H are identical.

CLAIM 4.6. *For any* $0 \le s \le 1$, $H_{\mathcal{S}}(s) = H'_{\mathcal{S}}(s)$.

Proof. Let $\Pi_{\mathcal{S}}$ be the orthogonal projection on $\mathcal{S}$. Then our goal is to show that $\Pi_{\mathcal{S}}H(s)\Pi_{\mathcal{S}} = \Pi_{\mathcal{S}}H'(s)\Pi_{\mathcal{S}}$. The only difference between $H(s)$ and $H'(s)$ is the factor of J in H_{clock}, and that the H_ℓ terms are replaced by H'_ℓ. We note that $H_{\mathcal{S},\text{clock}}$ is zero. Hence, it suffices to show that for all $1 \le \ell \le L$,

$$\Pi_{\mathcal{S}}H_\ell\Pi_{\mathcal{S}} = \Pi_{\mathcal{S}}H'_\ell\Pi_{\mathcal{S}}.$$

For this, observe that for any $1 < \ell < L$,

$$\Pi_{\mathcal{S}}|1\rangle\langle 0|^c_\ell\Pi_{\mathcal{S}} = |1^\ell 0^{L-\ell}\rangle\langle 1^{\ell-1}0^{L-(\ell-1)}|^c = \Pi_{\mathcal{S}}|110\rangle\langle 100|^c_{\ell-1,\ell,\ell+1}\Pi_{\mathcal{S}}$$

and similarly for $|0\rangle\langle 1|^c_\ell$. A similar statement holds for $\ell = 1, L$ with the right-hand term modified appropriately. □

Lemma 4.2 and Claim 4.6 imply that $\Delta(H'_{\mathcal{S}}(s)) = \Omega(L^{-3})$. We now want to deduce from this a lower bound on $\Delta(H'(s))$, without the restriction to $\mathcal{S}$. For this we use the following claim. Essentially, it says that if J is large enough, then the lower part of the spectrum of $H'(s)$ is similar to that of $H'_{\mathcal{S}}(s)$. More precisely, it shows that the lowest eigenvalues, the second lowest eigenvalues, and the ground states of the two Hamiltonians are close. Intuitively, this holds since the energy penalty given to states in $\mathcal{S}^\perp$, the orthogonal space to $\mathcal{S}$, is very high and hence any eigenvector with low eigenvalue must be almost orthogonal to $\mathcal{S}^\perp$ (and hence almost inside $\mathcal{S}$). We note that a similar lemma was used in [20] in the context of QMA-complete problems.

LEMMA 4.7. *Let* $H = H_1 + H_2$ *be the sum of two Hamiltonians operating on some Hilbert space* $\mathcal{H} = \mathcal{S} + \mathcal{S}^\perp$. *The Hamiltonian* H_2 *is such that* $\mathcal{S}$ *is a zero eigenspace and the eigenvectors in* $\mathcal{S}^\perp$ *have an eigenvalue of at least* $J > 2K$ *where* $K = \|H_1\|$. *Let* a *and* b *be the lowest and second lowest eigenvalues of* $H_{\mathcal{S}}$, *and let* a' *and* b' *be the corresponding quantities for* H. *Then the lowest eigenvalue of* H *satisfies* $a - \frac{K^2}{J-2K} \le a' \le a$ *and the second lowest eigenvalue of* H *satisfies* $b' \ge b - \frac{K^2}{J-2K}$. *If, moreover,* $b > a$, *then the ground states* $|\xi\rangle, |\xi'\rangle$ *of* $H_{\mathcal{S}}, H$, *respectively, satisfy*

$$|\langle\xi|\xi'\rangle|^2 \ge 1 - \frac{K^2}{(b-a)(J-2K)}.$$

Proof. First, we show that $a' \leq a$. Using $H_2|\xi\rangle = 0$,

$$\langle\xi|H|\xi\rangle = \langle\xi|H_1|\xi\rangle + \langle\xi|H_2|\xi\rangle \ = \ a$$

and hence H must have an eigenvector of eigenvalue at most a.

We now show the lower bound on a'. We can write any unit vector $|v\rangle \in \mathcal{H}$ as $|v\rangle = \alpha_1|v_1\rangle + \alpha_2|v_2\rangle$, where $|v_1\rangle \in \mathcal{S}$ and $|v_2\rangle \in \mathcal{S}^\perp$ are two unit vectors and α_1, α_2 are two nonnegative reals satisfying $\alpha_1^2 + \alpha_2^2 = 1$. Then we have,

$$\begin{aligned}\langle v|H|v\rangle &\geq \langle v|H_1|v\rangle + J\alpha_2^2\\ &= (1-\alpha_2^2)\langle v_1|H_1|v_1\rangle + 2\alpha_1\alpha_2\mathrm{Re}\langle v_1|H_1|v_2\rangle + \alpha_2^2\langle v_2|H_1|v_2\rangle + J\alpha_2^2\\ &\geq \langle v_1|H_1|v_1\rangle - K\alpha_2^2 - 2K\alpha_2 - K\alpha_2^2 + J\alpha_2^2\\ &= \langle v_1|H_1|v_1\rangle + (J-2K)\alpha_2^2 - 2K\alpha_2,\end{aligned}$$

where we used $\alpha_1^2 = 1 - \alpha_2^2$ and $\alpha_1 \leq 1$. Since $(J-2K)\alpha_2^2 - 2K\alpha_2$ is minimized for $\alpha_2 = K/(J-2K)$, we have

$$\langle v|H|v\rangle \geq \langle v_1|H_1|v_1\rangle - \frac{K^2}{J-2K}. \tag{10}$$

We obtain the required lower bound by noting that $\langle v_1|H_1|v_1\rangle \geq a$.

Consider now the two-dimensional space $\mathcal{L}$ spanned by the two eigenvectors of H corresponding to a' and b'. For any unit vector $|v\rangle \in \mathcal{L}$ we have $\langle v|H|v\rangle \leq b'$. Hence, if $\mathcal{L}$ contains a vector $|v\rangle$ orthogonal to $\mathcal{S}$, then we have $b' \geq \langle v|H|v\rangle \geq J - K > K \geq b$ and we are done. Otherwise, the projection of $\mathcal{L}$ on $\mathcal{S}$ must be a two-dimensional space. Being two-dimensional, this space must contain a vector orthogonal to $|\xi\rangle$. Let $|v\rangle$ be a vector in $\mathcal{L}$ whose projection on $\mathcal{S}$ is orthogonal to $|\xi\rangle$. By (10), $b' \geq \langle v|H|v\rangle \geq b - \frac{K^2}{J-2K}$, as required.

Finally, let $\beta = |\langle\xi|\xi'\rangle|^2$. Then we can write $|\xi\rangle = \sqrt{\beta}|\xi'\rangle + \sqrt{1-\beta}|{\xi'}^\perp\rangle$ for some unit vector $|{\xi'}^\perp\rangle$ orthogonal to $|\xi'\rangle$. Since $|\xi'\rangle$ is an eigenvector of H,

$$\begin{aligned}a \ = \ \langle\xi|H|\xi\rangle &= \beta\langle\xi'|H|\xi'\rangle + (1-\beta)\langle{\xi'}^\perp|H|{\xi'}^\perp\rangle\\ &\geq \beta a' + (1-\beta)b'\\ &\geq \beta\Big(a - \frac{K^2}{J-2K}\Big) + (1-\beta)\Big(b - \frac{K^2}{J-2K}\Big)\\ &= a + (1-\beta)(b-a) - \frac{K^2}{J-2K}.\end{aligned}$$

Rearranging, we obtain the required bound. □

We can now bound the spectral gap of $H'(s)$.

LEMMA 4.8. *For all* $0 \leq s \leq 1$, $\Delta(H'(s)) = \Omega(L^{-3})$.

Proof. We apply Lemma 4.7 by setting $H_2 = J \cdot H_{clock}$ and H_1 to be the remaining terms such that $H'(s) = H_1 + H_2$. Note that Lemma 4.7 implies that the spectral gap of $H'(s)$ is smaller than that of $H'_{\mathcal{S}}(s)$ (which is $\Omega(1/L^3)$ by Lemma 4.2) by at most $K^2/(J-2K)$. But it is easy to see that $K = O(L)$, due to the fact that H_1 consists of $O(L)$ terms, each of constant norm. The result follows since $J = \epsilon^{-2}L^6$. □

This shows the desired bound on the spectral gap. Before we complete the proof, we must show that the final ground state is close to the history state.

LEMMA 4.9. *The ground state of* $H'(1)$ *is* ϵ*-close to* $|\eta\rangle$.

Proof. Apply Lemma 4.7 as in the proof of Lemma 4.8, for the case $s = 1$. We obtain that the inner product squared between the ground state of $H'(1)$ and $|\eta\rangle$, is at least $1 - \delta$, with $\delta = \frac{K^2}{(b-a)(J-2K)} = O(L^{-1}\epsilon^2)$, where we have used $K = O(L)$, $J = \epsilon^{-2}L^6$, and $b - a = \Omega(1/L^3)$ by Lemma 4.2. This implies that the ℓ_2-distance between the ground state of $H'(1)$ and $|\eta\rangle$ is $O(\epsilon/\sqrt{L}) \leq \epsilon$. □

We now complete the proof of Theorem 4.5. The adiabatic algorithm starts with $|\gamma_0\rangle$ and evolves according to $H'(s)$ for $T = \theta(\epsilon^{-\delta}L^{7+3\delta})$. Such a T satisfies the adiabatic condition (see (2)), using $\|H'_{\text{final}} - H'_{\text{init}}\| = O(L)$. By Theorem 2.1 the final state is ϵ-close in ℓ_2-distance to the ground state of H'_{final}. Lemma 4.9 implies that this state is ϵ-close in ℓ_2-distance to $|\eta\rangle$. Using the triangle inequality we note that the output of the adiabatic computation is 2ϵ-close to $|\eta\rangle$. The running time of this algorithm is $O(T \cdot J \cdot L) = O(T \cdot \epsilon^{-2}L^7) = O(\epsilon^{-(2+\delta)}L^{14+3\delta})$.

We can now apply Lemma 3.10 to obtain a modified adiabatic computation whose output state after tracing out the clock qubits is ϵ-close in trace distance to $U|0^n\rangle$. The running time is $O(\epsilon^{-(16+4\delta)}L^{14+3\delta})$ for any fixed $\delta > 0$.

5. Two-local Hamiltonians on a two-dimensional lattice.

In this section we prove Theorem 1.3. We simulate a given quantum circuit by an adiabatic evolution of a system of 6-dimensional quantum particles arranged on a two-dimensional grid. More precisely, we prove the following theorem.

THEOREM 5.1. *Given a quantum circuit on n qubits with L two-qubit gates implementing a unitary U, and $\epsilon > 0$, there exists a 2-local adiabatic computation $AC(\text{poly}(n, L), 6, H_{\text{init}}, H_{\text{final}}, \epsilon)$ such that H_{init} and H_{final} involve only nearest neighbors on a 2-dimensional grid. Moreover, the running time of this algorithm is $\text{poly}(L, \frac{1}{\epsilon})$, and its output (after performing a partial measurement on each particle) is ϵ-close (in trace distance) to $U|0^n\rangle$. Finally, H_{init} and H_{final} can be computed by a polynomial time Turing machine.*

As mentioned in the introduction, the main problem in proving this theorem, and more precisely, in moving to a two-dimensional grid, is the notion of a clock. In the constructions of the previous section, the clock is represented by an additional register that counts the clock steps in unary representation. The terms H_ℓ, which check the correct propagation in the ℓth time step, interact between the ℓth qubit of the clock and the corresponding qubits on which U_ℓ operates. If we want to restrict the interaction to nearest neighbors in two dimensions using this idea, then no matter how the clock qubits are arranged on the grid, we run into problems interacting the qubits with the corresponding clock qubits in a local way. The solution to this problem lies in the way we represent the clock. Instead of using an extra register, we embed the clock into the same particles that perform the computation by defining the notion of a *shape* of a state, to be defined later. We then create a sequence of legal shapes, and show how states can evolve from one legal shape to another.

Although the construction of this section is more involved than the ones of the previous section, its analysis follows almost immediately from the analysis carried out in Theorem 4.5. To achieve this, we make sure that the Hamiltonians and some relevant subspaces are as similar as possible to those in the previous section.

5.1. Assumptions on the input circuit.

To simplify the construction of our adiabatic evolution, we first assume without loss of generality that the quantum circuit we wish to simulate has a particular layout of its gates. Namely, it consists of R rounds, where each round is composed of n nearest neighbor gates (some can be the identity gate), followed by n identity gates, as in Figure 3. More specifically, the first gate in each round is a one-qubit gate applied to the first qubit. For $i = 2, \dots, n$, the ith gate is a two-qubit gate applied to qubits $i-1$ and i. For $i = n+1, \dots, 2n$ the ith gate is an identity gate applied to the $(2n+1-i)$th qubit. These identity gates are included for convenience of notation. Any circuit can be transformed to such a form by introducing extra identity and swap gates. Let $L = 2nR$ be the total number of gates in the circuit so obtained. Clearly, L is at most polynomially larger than the number of gates in the original circuit.

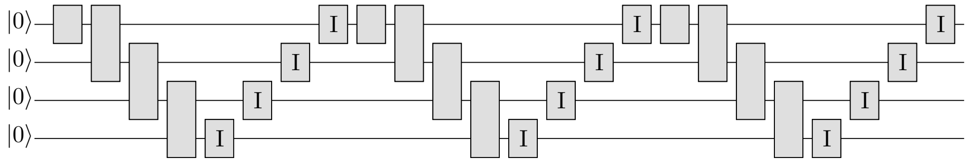


Fig. 3. *The modified circuit with $R = 3$.*

5.2. The particles of the adiabatic quantum system. The adiabatic computation is performed on 6-dimensional particles, arranged on a two-dimensional square lattice with n rows and $R+1$ columns. We number the rows from 1 (top) to n (bottom) and the columns from 0 (left) to R (right). Columns number 0 and 1 are used to simulate the first round of the circuit. Column numbers 1 and 2 are used for the second round of computation, and so on. We denote the six internal states of a particle by $|\bigcirc\rangle, |{\uparrow}\!\!\!\!\bigcirc\rangle, |{\downarrow}\!\!\!\!\bigcirc\rangle, |{\Uparrow}\!\!\!\!\bigcirc\rangle, |{\Downarrow}\!\!\!\!\bigcirc\rangle$, and $|\otimes\rangle$. These six states are divided into four *phases*: the *unborn phase* $|\bigcirc\rangle$, the *first phase* $|{\uparrow}\!\!\!\!\bigcirc\rangle, |{\downarrow}\!\!\!\!\bigcirc\rangle$, the *second phase* $|{\Uparrow}\!\!\!\!\bigcirc\rangle, |{\Downarrow}\!\!\!\!\bigcirc\rangle$, and the *dead phase* $|\otimes\rangle$. The two states in the first phase and the two states in the second phase correspond to computational degrees of freedom, namely to the $|0\rangle$ and $|1\rangle$ states of a qubit. We write $|{\updownarrow}\!\!\!\!\bigcirc\rangle$ to denote an arbitrary state in the subspace spanned by $|{\uparrow}\!\!\!\!\bigcirc\rangle$ and $|{\downarrow}\!\!\!\!\bigcirc\rangle$. Similarly, $|{\Updownarrow}\!\!\!\!\bigcirc\rangle$ denotes a state in the space spanned by $|{\Uparrow}\!\!\!\!\bigcirc\rangle$ and $|{\Downarrow}\!\!\!\!\bigcirc\rangle$. The phases are used to define the *shape* of the basis states. A shape of a basis state is simply an assignment of one of the four phases to each particle, ignoring the computational degrees of freedom inside the first and second phase. These shapes will be used instead of the clock states of the previous section.

5.3. Geometrical clock. We now describe the way we represent a clock using shapes. In the previous constructions, the space $\mathcal{S}$ of dimension $2^n(L+1)$ was the ground space of the clock, i.e., the space spanned by legal clock states. Inside the clock register there were $L+1$ legal clock states. Note that each such clock state can be described, essentially, in a geometric way by the "shape" of the clock particles: how many 1's precede how many 0's.

We now describe the corresponding subspaces involved in our construction for the two-dimensional case. For each $0 \le \ell \le L$, we have a 2^n-dimensional subspace corresponding to that clock state. Each of these $L+1$ subspaces can be described by its *shape*, that is, a setting of one of the four phases to each particle. See Figure 4 for an illustration with $n = 6, R = 6$. The six shapes shown correspond to clock states $\ell = 0, \ell = 4n, \ell = 4n+3, \ell = 5n+2, \ell = 6n$, and $\ell = 2nR$, respectively. Notice that each shape has exactly n particles in the first or second phase. Hence, the dimension of the subspace induced by each shape is 2^n. As ℓ goes from 0 to L, the shape changes from that shown in Figure 4a to that shown in Figure 4f. The locations at which the changes occur form a snake-like pattern winding down and up the lattice, following the layout of the gates in the input circuit (see Figure 3 for an example with $R = 3$).

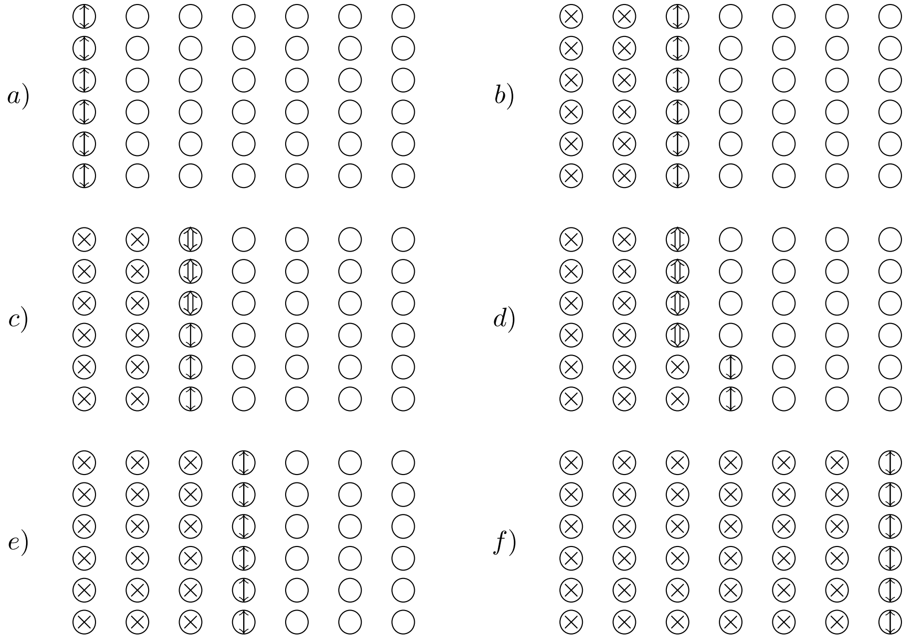


FIG. 4. *Legal shapes of clock states for* $n = 6$*,* $R = 6$ *at different stages of the computation (see subsection* 5.3*).*

We now describe the legal shapes more formally.

1. The shape corresponding to clock state $\ell = 2nr + k$ for $0 \leq k \leq n$ has its r leftmost columns in the dead phase. The top k particles in the $r+1$st column are in their second phase while the bottom $n-k$ are in the first phase. Particles in the remaining $R-r$ columns are all in the unborn phase.
2. The shape corresponding to clock state $\ell = 2nr+n+k$ for $1 \leq k \leq n-1$ has, as before, its r leftmost columns in the dead phase. The $r+1$st column has its $n-k$ topmost particles in the second phase, and its remaining k particles in the dead phase. The $r+2$nd column has its $n-k$ topmost particles in the unborn phase and its remaining k particles in the first phase. All remaining particles are in the unborn phase.

The subspace $\mathcal{S}$ is defined as the $(L+1)2^n$-dimensional space spanned by all legal shapes. As in previous sections we partition $\mathcal{S}$ into 2^n subspaces $\mathcal{S}_j$. Each subspace $\mathcal{S}_j$ is spanned by $L+1$ orthogonal states $|\gamma_0^j\rangle, \ldots, |\gamma_L^j\rangle$, defined as follows. For each $0 \leq \ell \leq L$ and $0 \leq j \leq 2^n - 1$, the shape of $|\gamma_\ell^j\rangle$ corresponds to ℓ. The state of the n active particles (i.e., those in either the first or second phase), when read from top to bottom, corresponds to the state of the circuit after the first ℓ gates are applied to an initial state corresponding to the binary representation of j; i.e., it corresponds to the state $U_\ell \cdot U_{\ell-1} \cdots U_1 |j\rangle$. More precisely, these particles are in a superposition obtained by mapping this state to the state of the n active particles in the following way: $|0\rangle$ to ↑⃝ (or ⇑⃝ for a second phase particle) and $|1\rangle$ to ↓⃝ (or ⇓⃝ for a second phase particle). We often denote $|\gamma_\ell^0\rangle$, which corresponds to the all 0 input, by $|\gamma_\ell\rangle$. For example, $|\gamma_0\rangle$ is shown in Figure 5.

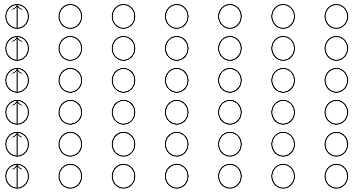


FIG. 5. *The initial state.*

With the risk of being somewhat redundant, let us now give an alternative description of the states $|\gamma_0^j\rangle, \ldots, |\gamma_L^j\rangle$. This description is more helpful in understanding the Hamiltonians H''_ℓ which we will define shortly. Consider a state $|\gamma_\ell^j\rangle$ for some $\ell = 2rn$. The n particles in the rth column are in their first phase and their computational degrees of freedom correspond to the state of the circuit's qubits at the beginning of the rth round. Particles to the left of this column are dead, those to the right of this column are unborn. The state $|\gamma_{\ell+1}^j\rangle$ is obtained from $|\gamma_\ell^j\rangle$ by changing the topmost particle in the rth column to a second phase particle and applying the first gate in the rth round (a one-qubit gate) to its computational degrees of freedom. Next, the state $|\gamma_{\ell+2}^j\rangle$ is obtained from $|\gamma_{\ell+1}^j\rangle$ by changing the second particle from above in the rth column to a second phase particle and applying the second gate in the rth round (a two-qubit gate) to both this particle and the one on top of it. We continue in a similar fashion until we reach $|\gamma_{\ell+n}^j\rangle$, in which the entire rth column is in the second phase. We refer to these steps as the downward stage.

Next, let us describe the upward stage. The state $|\gamma_{\ell+n+1}^j\rangle$ is obtained from $|\gamma_{\ell+n}^j\rangle$ by "moving" the bottommost particle in the rth column one location to the right. More precisely, the bottommost particle changes to the dead phase and the one to the right of it changes to the first phase. The computational degrees of freedom are the same in both states. This corresponds to the fact that the $n+1$st gate in a round of the circuit is the identity gate.[6] Continuing in a similar fashion, we see that the upwards stage ends in the state $|\gamma_{\ell+n+n}^j\rangle = |\gamma_{2(r+1)n}^j\rangle$ that matches the above description of the first state in a round.

5.4. The Hamiltonian. We now construct a two-local Hamiltonian that guarantees correct propagation from one γ state to the next. In other words, the Hamiltonian has the history state, namely the superposition over all the γ states, as its ground state. The construction of this Hamiltonian should be more or less clear by now, where the only subtleties are due to edge cases.

The initial and final Hamiltonians are defined as

$$H''_{\text{init}} := H''_{\text{clockinit}} \;+\; H''_{\text{input}} \;+\; J \cdot H''_{\text{clock}},$$

$$H''_{\text{final}} := \frac{1}{2}\sum_{\ell=1}^{L} H''_\ell + H''_{\text{input}} + J \cdot H''_{\text{clock}},$$

where $J = \epsilon^{-2} \cdot L^6$. These Hamiltonians are chosen to be as similar as possible to the corresponding Hamiltonians in previous sections. For example, H''_{clock} has as its

[6] We could allow arbitrary one-qubit gates here instead of identity gates. This leads to a slightly more efficient construction but also to more cumbersome Hamiltonians.

ground space the space of legal clock states, $\mathcal{S}$. As before, it allows us to essentially project all other Hamiltonians on $\mathcal{S}$, by assigning a large energy penalty to states with illegal shape. Also, the Hamiltonians H''_ℓ (once projected to $\mathcal{S}$) check correct propagation from one step to the next. Other terms also serve similar roles as before.

Let us start with the simplest terms. Define

$$H''_{\text{input}} := \sum_{i=1}^{n} (|\textcircled{\downarrow}\rangle\langle\textcircled{\downarrow}|)_{i,1}.$$

The indices indicate the row and column of the particle on which the Hamiltonian operates. This Hamiltonian checks that none of the particles in the leftmost column are in $|\textcircled{\downarrow}\rangle$. Then, define

$$H''_{\text{clockinit}} = (I - |\textcircled{\uparrow}\rangle\langle\textcircled{\uparrow}| - |\textcircled{\downarrow}\rangle\langle\textcircled{\downarrow}|)_{1,1}.$$

This Hamiltonian checks that the top-left particle is in a $|\textcircled{\updownarrow}\rangle$ state. The remaining terms are described in the following subsections.

5.4.1. The clock Hamiltonian. The shapes we define satisfy the following important property: there exist *two-local* conditions that guarantee that a shape is legal. This allows us to define a two-local clock Hamiltonian, H''_{clock}, whose ground space is exactly $\mathcal{S}$, the $(L+1)2^n$-dimensional space spanned by all legal shapes.

TABLE 1
Local rules for basis state to be in $\mathcal{S}$.

Forbidden	Guarantees that
$\bigcirc\textcircled{\updownarrow}, \bigcirc\textcircled{\Updownarrow}, \bigcirc\otimes$	$\bigcirc$ is to the right of all other qubits
$\bigcirc\otimes, \textcircled{\updownarrow}\otimes, \textcircled{\Updownarrow}\otimes$	$\otimes$ is to the left of all other qubits
$\bigcirc\otimes, \otimes\bigcirc$	$\bigcirc$ and $\otimes$ are not horizontally adjacent
$\textcircled{\updownarrow}\textcircled{\updownarrow},\ \textcircled{\updownarrow}\textcircled{\Updownarrow},$ $\textcircled{\Updownarrow}\textcircled{\updownarrow},\ \textcircled{\Updownarrow}\textcircled{\Updownarrow}$	only one of $\textcircled{\updownarrow}$, $\textcircled{\Updownarrow}$ per row
$\begin{matrix}\bigcirc\\ \textcircled{\Updownarrow}\end{matrix}, \begin{matrix}\textcircled{\updownarrow}\\ \textcircled{\Updownarrow}\end{matrix}, \begin{matrix}\otimes\\ \textcircled{\Updownarrow}\end{matrix}$	only $\textcircled{\Updownarrow}$ above $\textcircled{\Updownarrow}$
$\begin{matrix}\textcircled{\updownarrow}\\ \bigcirc\end{matrix}, \begin{matrix}\textcircled{\updownarrow}\\ \textcircled{\Updownarrow}\end{matrix}, \begin{matrix}\textcircled{\updownarrow}\\ \otimes\end{matrix}$	only $\textcircled{\updownarrow}$ below $\textcircled{\updownarrow}$
$\begin{matrix}\bigcirc\\ \otimes\end{matrix}, \begin{matrix}\otimes\\ \bigcirc\end{matrix}$	$\bigcirc$ and $\otimes$ are not vertically adjacent
$\begin{matrix}\textcircled{\Updownarrow}\\ \bigcirc\end{matrix}, \begin{matrix}\otimes\\ \textcircled{\updownarrow}\end{matrix}$	no $\bigcirc$ below $\textcircled{\Updownarrow}$ and no $\textcircled{\updownarrow}$ below $\otimes$

CLAIM 5.2. *A shape is legal if and only if it contains none of the forbidden configurations of Table* 1.

Proof. It is easy to check that any legal shape contains none of the forbidden configurations. For the other direction, consider any shape that contains none of these configurations. Observe that each row must be of the form $\otimes^*[\textcircled{\updownarrow}, \textcircled{\Updownarrow}]\bigcirc^*$, that is, it starts with a sequence of zero or more $\otimes$, it then contains either $\textcircled{\updownarrow}$ or $\textcircled{\Updownarrow}$, and then ends with a sequence of zero or more $\bigcirc$. Columns can be of three different forms. Read from top to bottom, it is either $\textcircled{\Updownarrow}^*\textcircled{\updownarrow}^*$, $\textcircled{\Updownarrow}^*\otimes^*$, or $\bigcirc^*\textcircled{\updownarrow}^*$. It is now easy to verify that such a shape must be one of the legal shapes. □

Using this claim, we can define a two-local nearest-neighbor Hamiltonian that guarantees a legal shape. For example, if the rule forbids a particle at location (i,j) in state $\bigcirc$ to the left of a particle at location $(i,j+1)$ in state $\otimes$, then the corresponding term in the Hamiltonian is $(|\bigcirc, \otimes\rangle\langle\bigcirc, \otimes|)_{(i,j),(i,j+1)}$. Summing over all the forbidden configurations of Table 1 and over all relevant pairs of particles, we have

$$H''_{\rm clock} := \sum_{r \in \rm rules} H_r.$$

Note that the ground space of $H''_{\rm clock}$ is the $(L+1)2^n$-dimensional space $\mathcal{S}$.

5.4.2. The propagation Hamiltonian. The choice of legal shapes has the following important property: the shape of ℓ and that of $\ell+1$ differ in at most two locations. This means that for any ℓ and j, the shape of $|\gamma^j_{\ell-1}\rangle$ and that of $|\gamma^j_\ell\rangle$ differ in at most two locations. Moreover, if we consider the state of the n active particles in both states we see that these differ on at most two particles, namely, those on which the ℓth gate in the circuit acts. Crucially, and this is where we use our assumption on the form of the circuit (Figure 3), the particle(s) on which the ℓth gate acts are at the same location as the particle(s) whose phase changes. It is this structure that allows us to define the Hamiltonians H''_ℓ. These Hamiltonians act on two particles and "simultaneously" advance the clock (by changing the shape) and advance the computational state (by modifying the state of the active particles). Since $|\gamma_\ell\rangle$ differs from $|\gamma_{\ell-1}\rangle$ in at most two adjacent lattice sites, this can be done using a two-body nearest neighbor Hamiltonian.

The definition of H''_ℓ depends on whether ℓ is in the downward phase (i.e., is of the form $2rn+k$ for $1 \le k \le n$) or in the upward phase (i.e., is of the form $2rn+n+k$ for $1 \le k \le n$). We first define H''_ℓ for the upward phase. Assume $\ell = 2rn+n+k$ for some $0 \le r < R, 1 < k < n$ and let $i = n-k+1$ be the row in which $|\gamma_{\ell-1}\rangle$ and $|\gamma_\ell\rangle$ differ. Then,

$$\begin{aligned} H''_\ell := {} & \left|\begin{matrix}\text{\textcircled{$\Uparrow$}}\\ \otimes\end{matrix}\right\rangle\left\langle\begin{matrix}\text{\textcircled{$\Uparrow$}}\\ \otimes\end{matrix}\right|^{\rm i,r}_{\rm i+1,r} + \left|\begin{matrix}\bigcirc\\ \text{\textcircled{$\uparrow$}}\end{matrix}\right\rangle\left\langle\begin{matrix}\bigcirc\\ \text{\textcircled{$\uparrow$}}\end{matrix}\right|^{\rm i-1,r+1}_{\rm i,r+1} - (|\text{\textcircled{$\Uparrow$}}, \bigcirc\rangle\langle\otimes, \text{\textcircled{$\uparrow$}}| + |\otimes, \text{\textcircled{$\uparrow$}}\rangle\langle\text{\textcircled{$\Uparrow$}}, \bigcirc|)_{\rm (i,r)(i,r+1)} \\ & + \left|\begin{matrix}\text{\textcircled{$\Downarrow$}}\\ \otimes\end{matrix}\right\rangle\left\langle\begin{matrix}\text{\textcircled{$\Downarrow$}}\\ \otimes\end{matrix}\right|^{\rm i,r}_{\rm i+1,r} + \left|\begin{matrix}\bigcirc\\ \text{\textcircled{$\downarrow$}}\end{matrix}\right\rangle\left\langle\begin{matrix}\bigcirc\\ \text{\textcircled{$\downarrow$}}\end{matrix}\right|^{\rm i-1,r+1}_{\rm i,r+1} - (|\text{\textcircled{$\Downarrow$}}, \bigcirc\rangle\langle\otimes, \text{\textcircled{$\downarrow$}}| + |\otimes, \text{\textcircled{$\downarrow$}}\rangle\langle\text{\textcircled{$\Downarrow$}}, \bigcirc|)_{\rm (i,r)(i,r+1)}. \end{aligned}$$

The first line corresponds to changing the state $|\text{\textcircled{$\Uparrow$}}, \bigcirc\rangle$ into $|\otimes, \text{\textcircled{$\uparrow$}}\rangle$. The second line is similar for $|\text{\textcircled{$\Downarrow$}}, \bigcirc\rangle$ and $|\otimes, \text{\textcircled{$\downarrow$}}\rangle$. The purpose of the first two terms in each line is the same as that of $|100\rangle\langle100|^c$ and $|110\rangle\langle110|^c$ in H_ℓ from previous sections.[7] The difference is that here, to uniquely identify the current clock state, we need to consider particles on top of each other. The remaining terms in each line correspond to $|100\rangle\langle110|^c$ and $|100\rangle\langle110|^c$ in H_ℓ.

For the case $k = 1, n$, the definition is

$$\begin{aligned} H''_{2rn+n+1} := {} & |\text{\textcircled{$\Uparrow$}}\rangle\langle\text{\textcircled{$\Uparrow$}}|_{n,r} + \left|\begin{matrix}\bigcirc\\ \text{\textcircled{$\uparrow$}}\end{matrix}\right\rangle\left\langle\begin{matrix}\bigcirc\\ \text{\textcircled{$\uparrow$}}\end{matrix}\right|^{\rm n-1,r+1}_{\rm n,r+1} - (|\text{\textcircled{$\Uparrow$}}, \bigcirc\rangle\langle\otimes, \text{\textcircled{$\uparrow$}}| + |\otimes, \text{\textcircled{$\uparrow$}}\rangle\langle\text{\textcircled{$\Uparrow$}}, \bigcirc|)_{\rm (n,r)(n,r+1)} \\ & + |\text{\textcircled{$\Downarrow$}}\rangle\langle\text{\textcircled{$\Downarrow$}}|_{n,r} + \left|\begin{matrix}\bigcirc\\ \text{\textcircled{$\downarrow$}}\end{matrix}\right\rangle\left\langle\begin{matrix}\bigcirc\\ \text{\textcircled{$\downarrow$}}\end{matrix}\right|^{\rm n-1,r+1}_{\rm n,r+1} - (|\text{\textcircled{$\Downarrow$}}, \bigcirc\rangle\langle\otimes, \text{\textcircled{$\downarrow$}}| + |\otimes, \text{\textcircled{$\downarrow$}}\rangle\langle\text{\textcircled{$\Downarrow$}}, \bigcirc|)_{\rm (n,r)(n,r+1)}, \\ H''_{2rn+2n} := {} & \left|\begin{matrix}\text{\textcircled{$\Uparrow$}}\\ \otimes\end{matrix}\right\rangle\left\langle\begin{matrix}\text{\textcircled{$\Uparrow$}}\\ \otimes\end{matrix}\right|^{\rm 1,r}_{\rm 2,r} + |\text{\textcircled{$\uparrow$}}\rangle\langle\text{\textcircled{$\uparrow$}}|_{1,r+1} - (|\text{\textcircled{$\Uparrow$}}, \bigcirc\rangle\langle\otimes, \text{\textcircled{$\uparrow$}}| + |\otimes, \text{\textcircled{$\uparrow$}}\rangle\langle\text{\textcircled{$\Uparrow$}}, \bigcirc|)_{\rm (1,r)(1,r+1)} \\ & + \left|\begin{matrix}\text{\textcircled{$\Downarrow$}}\\ \otimes\end{matrix}\right\rangle\left\langle\begin{matrix}\text{\textcircled{$\Downarrow$}}\\ \otimes\end{matrix}\right|^{\rm 1,r}_{\rm 2,r} + |\text{\textcircled{$\downarrow$}}\rangle\langle\text{\textcircled{$\downarrow$}}|_{1,r+1} - (|\text{\textcircled{$\Downarrow$}}, \bigcirc\rangle\langle\otimes, \text{\textcircled{$\downarrow$}}| + |\otimes, \text{\textcircled{$\downarrow$}}\rangle\langle\text{\textcircled{$\Downarrow$}}, \bigcirc|)_{\rm (1,r)(1,r+1)}. \end{aligned}$$

[7] There are other (equally good) ways to define these terms. For example, it is possible to define them so that they both act on the rth column.

For the downward stage, H''_ℓ checks that a gate is applied correctly. For $\ell = 2nr+k$ and $1 < k < n$ we define

$$H''_\ell := \begin{pmatrix} 0 & -U_\ell \\ -U_\ell^\dagger & 0 \end{pmatrix} + \left(\left|\begin{matrix}\textcircled{\Uparrow}\\ \textcircled{\uparrow}\end{matrix}\right\rangle\left\langle\begin{matrix}\textcircled{\Uparrow}\\ \textcircled{\uparrow}\end{matrix}\right| + \left|\begin{matrix}\textcircled{\Uparrow}\\ \textcircled{\downarrow}\end{matrix}\right\rangle\left\langle\begin{matrix}\textcircled{\Uparrow}\\ \textcircled{\downarrow}\end{matrix}\right| + \left|\begin{matrix}\textcircled{\Downarrow}\\ \textcircled{\uparrow}\end{matrix}\right\rangle\left\langle\begin{matrix}\textcircled{\Downarrow}\\ \textcircled{\uparrow}\end{matrix}\right| + \left|\begin{matrix}\textcircled{\Downarrow}\\ \textcircled{\downarrow}\end{matrix}\right\rangle\left\langle\begin{matrix}\textcircled{\Downarrow}\\ \textcircled{\downarrow}\end{matrix}\right| \right)\begin{matrix}\scriptstyle \text{k}-1,\text{r}\\ \scriptstyle \text{k,r}\end{matrix}$$

$$+ \left(\left|\begin{matrix}\textcircled{\Uparrow}\\ \textcircled{\uparrow}\end{matrix}\right\rangle\left\langle\begin{matrix}\textcircled{\Uparrow}\\ \textcircled{\uparrow}\end{matrix}\right| + \left|\begin{matrix}\textcircled{\Uparrow}\\ \textcircled{\downarrow}\end{matrix}\right\rangle\left\langle\begin{matrix}\textcircled{\Uparrow}\\ \textcircled{\downarrow}\end{matrix}\right| + \left|\begin{matrix}\textcircled{\Downarrow}\\ \textcircled{\uparrow}\end{matrix}\right\rangle\left\langle\begin{matrix}\textcircled{\Downarrow}\\ \textcircled{\uparrow}\end{matrix}\right| + \left|\begin{matrix}\textcircled{\Downarrow}\\ \textcircled{\downarrow}\end{matrix}\right\rangle\left\langle\begin{matrix}\textcircled{\Downarrow}\\ \textcircled{\downarrow}\end{matrix}\right| \right)\begin{matrix}\scriptstyle \text{k,r}\\ \scriptstyle \text{k}+1,\text{r}\end{matrix}.$$

The last two terms are meant, as before, to replace the terms $|110\rangle\langle 110|^c$ and $|100\rangle\langle 100|^c$. Once again, to uniquely identify the current clock state, we need to consider particles on top of each other. The first term represents a Hamiltonian that acts on the two particles in positions (k, r) and $(k+1, r)$. These particles span a 36-dimensional space. The matrix shown above is in fact the restriction of this Hamiltonian to the 8-dimensional space spanned by

$$\left|\begin{matrix}\textcircled{\Uparrow}\\ \textcircled{\uparrow}\end{matrix}\right\rangle \left|\begin{matrix}\textcircled{\Uparrow}\\ \textcircled{\downarrow}\end{matrix}\right\rangle \left|\begin{matrix}\textcircled{\Downarrow}\\ \textcircled{\uparrow}\end{matrix}\right\rangle \left|\begin{matrix}\textcircled{\Downarrow}\\ \textcircled{\downarrow}\end{matrix}\right\rangle \quad \left|\begin{matrix}\textcircled{\Uparrow}\\ \textcircled{\Uparrow}\end{matrix}\right\rangle \left|\begin{matrix}\textcircled{\Uparrow}\\ \textcircled{\Downarrow}\end{matrix}\right\rangle \left|\begin{matrix}\textcircled{\Downarrow}\\ \textcircled{\Uparrow}\end{matrix}\right\rangle \left|\begin{matrix}\textcircled{\Downarrow}\\ \textcircled{\Downarrow}\end{matrix}\right\rangle$$

(recall that U_ℓ acts on two qubits and is therefore a 4×4 matrix). Everywhere else in this 36-dimensional subspace, this Hamiltonian acts trivially, i.e., is 0.

For the case $k = n$ we slightly modify the terms that identify the clock states

$$H''_{2nr+n} := \begin{pmatrix} 0 & -U_{2nr+n} \\ -U_{2nr+n}^\dagger & 0 \end{pmatrix} + \left(\left|\begin{matrix}\textcircled{\Uparrow}\\ \textcircled{\uparrow}\end{matrix}\right\rangle\left\langle\begin{matrix}\textcircled{\Uparrow}\\ \textcircled{\uparrow}\end{matrix}\right| + \left|\begin{matrix}\textcircled{\Uparrow}\\ \textcircled{\downarrow}\end{matrix}\right\rangle\left\langle\begin{matrix}\textcircled{\Uparrow}\\ \textcircled{\downarrow}\end{matrix}\right| + \left|\begin{matrix}\textcircled{\Downarrow}\\ \textcircled{\uparrow}\end{matrix}\right\rangle\left\langle\begin{matrix}\textcircled{\Downarrow}\\ \textcircled{\uparrow}\end{matrix}\right| \right.$$

$$\left. + \left|\begin{matrix}\textcircled{\Downarrow}\\ \textcircled{\downarrow}\end{matrix}\right\rangle\left\langle\begin{matrix}\textcircled{\Downarrow}\\ \textcircled{\downarrow}\end{matrix}\right| \right)\begin{matrix}\scriptstyle \text{n}-1,\text{r}\\ \scriptstyle \text{n,r}\end{matrix}$$

$$+ \left(|\textcircled{\Uparrow}\rangle\langle\textcircled{\Uparrow}| + |\textcircled{\Downarrow}\rangle\langle\textcircled{\Downarrow}|\right)_{n,r}.$$

For the case $k = 1$ we have

$$H''_{2nr+1} := \begin{pmatrix} 0 & -U_{2nr+1} \\ -U_{2nr+1}^\dagger & 0 \end{pmatrix} + \left(|\textcircled{\uparrow}\rangle\langle\textcircled{\uparrow}| + |\textcircled{\downarrow}\rangle\langle\textcircled{\downarrow}|\right)_{1,r}$$

$$+ \left(\left|\begin{matrix}\textcircled{\Uparrow}\\ \textcircled{\uparrow}\end{matrix}\right\rangle\left\langle\begin{matrix}\textcircled{\Uparrow}\\ \textcircled{\uparrow}\end{matrix}\right| + \left|\begin{matrix}\textcircled{\Uparrow}\\ \textcircled{\downarrow}\end{matrix}\right\rangle\left\langle\begin{matrix}\textcircled{\Uparrow}\\ \textcircled{\downarrow}\end{matrix}\right| + \left|\begin{matrix}\textcircled{\Downarrow}\\ \textcircled{\uparrow}\end{matrix}\right\rangle\left\langle\begin{matrix}\textcircled{\Downarrow}\\ \textcircled{\uparrow}\end{matrix}\right| \right.$$

$$\left. + \left|\begin{matrix}\textcircled{\Downarrow}\\ \textcircled{\downarrow}\end{matrix}\right\rangle\left\langle\begin{matrix}\textcircled{\Downarrow}\\ \textcircled{\downarrow}\end{matrix}\right| \right)\begin{matrix}\scriptstyle \text{1,r}\\ \scriptstyle \text{2,r}\end{matrix},$$

where the first term shows the restriction an operator acting on the particle $(1, r)$ to the four-dimensional space spanned by $|\textcircled{\uparrow}\rangle, |\textcircled{\downarrow}\rangle, |\textcircled{\Uparrow}\rangle, |\textcircled{\Downarrow}\rangle$ (recall that U_{2nr+1} is a one-qubit gate).

5.5. Spectral gap. The analysis of the spectral gap follows almost immediately from that in subsection 4.2.2. The main effort is in verifying that the restriction of each of our Hamiltonians to $\mathcal{S}$ is identical to the restriction of the corresponding Hamiltonian in previous sections to $\mathcal{S}$, when both are constructed according to the modified quantum circuit of subsection 5.1. This, in fact, does not hold for H''_{input}, whose projection is not quite the same as that of H_{input}; still, it is similar enough for the analysis in subsection 4.2.2 to hold.

CLAIM 5.3. $H''_{\mathcal{S},\text{clockinit}} = H_{\mathcal{S},\text{clockinit}}$.

Proof. Both Hamiltonians are diagonal in the basis $|\gamma_\ell^j\rangle$ with eigenvalue 0 for $\ell = 0$ and eigenvalue 1 for any $\ell > 0$. □

CLAIM 5.4. *For any* $1 \leq \ell \leq L$, $H''_{\mathcal{S},\ell} = H_{\mathcal{S},\ell}$.

Proof. It is straightforward to verify that both Hamiltonians, when restricted to $\mathcal{S}$, are equal to

$$\sum_{j=0}^{2^n-1} [|\gamma_\ell^j\rangle\langle\gamma_\ell^j| + |\gamma_{\ell-1}^j\rangle\langle\gamma_{\ell-1}^j| - |\gamma_\ell^j\rangle\langle\gamma_{\ell-1}^j| - |\gamma_{\ell-1}^j\rangle\langle\gamma_\ell^j|]. \qquad \square$$

For H''_{input} the situation is similar, although in this case the restriction to $\mathcal{S}$ is not exactly the same. Still, the resemblance is enough for the same analysis to hold.

CLAIM 5.5. *Both* $H_{\mathcal{S},\text{input}}$ *and* $H''_{\mathcal{S},\text{input}}$ *are diagonal in the basis* $|\gamma_\ell^j\rangle$. *Moreover, the eigenvalue in both Hamiltonians corresponding to* $|\gamma_\ell^j\rangle$ *for* $\ell = 0$ *is exactly the number of* 1*'s in the binary representation of* j.

Proof. Easy to verify. $\square$

The similarity between the two Hamiltonians breaks down as follows. While the eigenvalues corresponding to $|\gamma_\ell^j\rangle$ for $\ell > 0$ are 0 in $H_{\mathcal{S},\text{input}}$, those in $H''_{\mathcal{S},\text{input}}$ might be positive (namely, for $0 \leq \ell \leq n$, the eigenvalue of $|\gamma_\ell^j\rangle$ is the number of 1's in the last $n - \ell$ digits in the binary representation of j). Nevertheless, due to the remark at the end of subsection 4.1, Lemma 4.2 holds here as well. We then get the following lemma.

LEMMA 5.6. *For any* $0 \leq s \leq 1$, $H''_{\mathcal{S}}(s)$ *has a spectral gap of* $\Omega(L^{-3})$. *Moreover, the ground state of* $H''_{\mathcal{S},\text{final}}$ *is* $|\eta\rangle$.

The rest of the proof of Theorem 1.3 is essentially the same as in subsection 4.2.2. By applying Lemma 4.7, we obtain the following lemma.

LEMMA 5.7. *For all* $0 \leq s \leq 1, \Delta(H''(s)) = \Omega(L^{-3})$. *Moreover, the ground state of* $H''(1)$ *is* ϵ*-close to* $|\eta\rangle$.

The proof is similar to that of Lemmas 4.8 and 4.9. This enables us to adiabatically generate the history state with exactly the same running time as in the three-local case (when the number of gates is that of the modified circuit of subsection 5.1).

Finally, we would like to apply Lemma 3.10 as before. However, we cannot quite do this due to a technical issue: our Hilbert space is no longer a tensor product of computation qubits and clock qubits and tracing out the clock qubits is meaningless. Nevertheless, a minor modification of that lemma still applies. We first add, say, L/ϵ identity gates to the end of the (modified) circuit. Now, the adiabatic computation produces a state close to the history state. We then measure the shape of the system without measuring the inner computational degrees of freedom. Due to the additional identity gates, with all but ϵ probability, the outcome of the measurement is a shape ℓ for $\ell \geq L$. If this is the case, then the state of the system is such that the active particles are in the final state of the circuit, as desired. This completes the proof of Theorem 5.1.

Acknowledgments. We wish to thank Dave Bacon, Ed Farhi, Leonid Gurvitz, and Umesh Vazirani for inspiring discussions. Part of this work was done while Dorit Aharonov, Wim van Dam, Julia Kempe, and Zeph Landau were members of the MSRI, Berkeley, CA.

REFERENCES

[1] J. ABERG, D. KULT, AND E. SJOQVIST, *Robustness of adiabatic quantum search*, arXiv:quant-ph/0412124, 2004.

[2] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev, *Adiabatic quantum computation is equivalent to standard quantum computation*, in Proceedings of the 45th Foundations of Comput. Sci., Rome, Italy, 2004, pp. 42–51.

[3] D. Aharonov, A. Kitaev, and N. Nisan, *Quantum circuits with mixed states*, in Proceedings of the 30th Annual ACM STOC, Dallas, TX, 1998, pp. 20–30.

[4] D. Aharonov and A. Ta-Shma, *Adiabatic quantum state generation and statistical zero-knowledge*, in Proceedings of the 35th Annual ACM STOC, San Diego, CA, 2003, pp. 20–29.

[5] A. Ambainis and O. Regev, *An elementary proof of the quantum adiabatic theorem*, arXiv:quant-ph/0411152, 2004.

[6] Y. Avron and A. Elgart, *Adiabatic theorem without a gap condition*, Comm. Math. Phys., 203, 1999, pp. 445–463.

[7] R. Bhatia, *Matrix Analysis,* Grad. Texts in Math. 169, Springer-Verlag, New York, 1997.

[8] A. Childs, E. Farhi, and J. Preskill, *Robustness of adiabatic quantum computation*, Phys. Review A, 65:012322, 2002.

[9] W. van Dam, M. Mosca, and U. Vazirani, *How powerful is adiabatic quantum computation?*, in Proceedings of the 42nd IEEE Symposium on Foundation of Comput. Sci., Las Vegas, NV, IEEE Computer Society, Los Alamitos, CA, 2001, pp. 279–287.

[10] W. van Dam and U. Vazirani, *More on the Power of Adiabatic Computation*, unpublished, 2001.

[11] E. Farhi, J. Goldstone, and S. Gutmann, *Quantum adiabatic evolution algorithms versus simulated annealing*, arXiv:quant-ph/0201031, 2002.

[12] E. Farhi, J. Goldstone, and S. Gutmann, *Quantum adiabatic evolution algorithms with different paths*, arXiv:quant-ph/0208135, 2002.

[13] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, *A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem*, Science, 292 (2001), pp. 472–476.

[14] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, *Quantum computation by adiabatic evolution*, arXiv:quant-ph/0001106, 2000.

[15] J. Friedman, *Expanding Graphs,* DIMACS Series in Discrete Mathematics and Theoretical Computer Science, AMS, Providence, RI, 1993.

[16] L. Grover, *Quantum mechanics helps in searching for a needle in a haystack*, Phys. Rev. Lett., 79:325, 1997.

[17] R. A. Horn and C. R. Johnson, *Matrix Analysis,* Cambridge University Press, Cambridge, UK, 1985.

[18] J. A. Jones, V. Vedral, A. Ekert, and G. Castagnoli, *Geometric quantum computation with NMR*, Nature, 403 (2000), pp. 869–871.

[19] T. Kato, *On the adiabatic theorem of quantum mechanics*, J. Phys. Soc. Jap., 5 (1951), pp. 435–439.

[20] J. Kempe and O. Regev, 3*-local Hamiltonian is QMA-complete*, Quantum Inf. Comput., 3 (2003), pp. 258–264.

[21] J. Kempe, A. Kitaev, and O. Regev, *The complexity of the local Hamiltonian problem*, in Proceedings of the 24th Annual FSTTCS, Lecture Notes in Comput. Sci. 3328, Springer-Verlag, Berlin, 2004, pp. 372–383.

[22] A. Kitaev, A. Shen, and M. Vyalyi, *Classical and Quantum Computation,* Grad. Series in Math. 47, AMS, Providence, RI, 2002.

[23] A. Y. Kitaev, *Fault-tolerant quantum computation by anyons*, Ann. Physics, 303 (2003), pp. 2–30.

[24] L. Lovász, *Random walks on graphs: A survey*, in Combinatorics, Paul Erdős is Eighty, Vol. 2 (Keszthely, 1993), Bolyai Soc. Math. Stud., vol. 2 János Bolyai Math. Soc., Budapest, 1996, pp. 353–397.

[25] A. Messiah, *Quantum Mechanics,* John Wiley and Sons, New York, 1958.

[26] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information,* Cambridge University Press, Cambridge, UK, 2000.

[27] J. Pachos and P. Zanardi, *Quantum holonomies for quantum computing*, Int. J. Mod. Phys. B, 15 (2001), pp. 1257–1286.

[28] C. Papadimitriou, *Computational Complexity,* Addison Wesley, Reading, MA, 1994.

[29] B. Reichardt, *The quantum adiabatic optimization algorithm and local minima*, in Proceedings of the 36th Annual ACM STOC, Chicago, IL, 2004, pp. 502–510.

[30] J. Roland and N. Cerf, *Quantum search by local adiabatic evolution*, Phys. Rev. A, 65:042308, 2002.

[31] J. Roland and N. Cerf, *Noise resistance of adiabatic quantum computation using random*

matrix theory, Phys. Rev. A, arXiv:quant-ph/0409127, 2004, to appear.

[32] G. Santoro, R. Martonak, E. Tosatti, and R. Car, *Theory of quantum annealing of an Ising spin glass*, Science, 295 (2002), pp. 2427–2430.

[33] M. S. Sarandy and D. Lidar, *Adiabatic quantum computation in open systems*, arXiv:quant-ph/0502014, 2005.

[34] A. Sinclair and M. Jerrum, *Approximate counting, uniform generation, and rapidly mixing Markov chains* (extended abstract), in Graph-Theoretic Concepts in Computer Science (Staffelstein, 1987), Lecture Notes in Comput. Sci. 314, Springer-Verlag, Berlin, 1988, pp. 134–148.

[35] W. Spitzer and S. Starr, *Improved bounds on the spectral gap above frustration-free ground states of quantum spin chains*, Lett. Math. Phys., 63 (2003), pp. 165–177.

[36] M. Steffen, W. van Dam, T. Hogg, G. Breyta, and I. Chuang, *Experimental implementation of an adiabatic quantum optimization algorithm*, Phys. Rev. Lett., 90, 067903, 2003.

SIAM J. COMPUT.
Vol. 37, No. 1, pp. 195–209

ON THE LIST AND BOUNDED DISTANCE DECODABILITY OF REED–SOLOMON CODES*

QI CHENG[†] AND DAQING WAN[‡]

Abstract. For an error-correcting code and a distance bound, the *list decoding problem* is to compute all the codewords within a given distance to a received message. The *bounded distance decoding* problem is to find one codeword if there is at least one codeword within the given distance, or to output the empty set if there is not. Obviously the bounded distance decoding problem is not as hard as the list decoding problem. For a Reed–Solomon code $[n, k]_q$, a simple counting argument shows that for any integer $0 < g < n$, there exists at least one Hamming ball of radius $n - g$, which contains at least $\binom{n}{g}/q^{g-k}$ many codewords. Let $\hat{g}(n, k, q)$ be the smallest positive integer g such that $\binom{n}{g}/q^{g-k} \leq 1$. One knows that

$$k - 1 \leq \hat{g}(n, k, q) \leq \sqrt{n(k-1)} \leq n.$$

For the distance bound up to $n - \sqrt{n(k-1)}$, it is known that both the list and bounded distance decoding can be solved efficiently. For the distance bound between $n - \sqrt{n(k-1)}$ and $n - \hat{g}(n, k, q)$, we do not know whether the Reed–Solomon code is list or bounded distance decodable; nor do we know whether there are polynomially many codewords in all balls of the radius. It is generally believed that the answer to both questions is no. In this paper, we prove the following: (1) List decoding cannot be done for radius $n - \hat{g}(n, k, q)$ or larger, unless the discrete logarithm over $\mathbf{F}_{q^{\hat{g}(n,k,q)-k}}$ is easy. (2) Let h and g be positive integers satisfying $q \geq \max(g^2, (h-1)^{2+\epsilon})$ and $g \geq (\frac{4}{\epsilon}+2)(h+1)$ for a constant $\epsilon > 0$. We show that the discrete logarithm problem over $\mathbf{F}_{q^h}$ can be efficiently reduced by a randomized algorithm to the bounded distance decoding problem of the Reed–Solomon code $[q, g-h]_q$ with radius $q - g$. These results show that the decoding problems for the Reed–Solomon code are at least as hard as the discrete logarithm problem over certain finite fields. For the list decoding problem of Reed–Solomon codes, although the infeasible radius that we obtain is much larger than the radius, which is known to be feasible, it is the first nontrivial bound. Our result on the bounded distance decodability of Reed–Solomon codes is also the first of its kind. The main tools for obtaining these results are an interesting connection between the problem of list decoding of Reed–Solomon code, the problem of a discrete logarithm over finite fields, and a generalization of Katz's theorem on representations of elements in an extension finite field by products of distinct linear factors.

Key words. list decoding algorithm, bounded distance decoding algorithm, Reed–Solomon codes, discrete logarithm problem

AMS subject classifications. 11Y16, 94B35, 94B65

DOI. 10.1137/S0097539705447335

1. Introduction.

An error-correcting code C over a finite alphabet Σ is an injective map $\phi : \Sigma^k \to \Sigma^n$. When we need to transmit a message of k letters over a noisy channel, we apply the map on the message first (i.e., encode the message) and send its image (i.e., the codeword) of n letters over the channel. The Hamming distance between two sequences of letters of the same length is the number of positions

*Received by the editors March 23, 2005; accepted for publication (in revised form) August 10, 2005; published electronically May 14, 2007. A preliminary version of this paper appeared in *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science* (FOCS), IEEE, Piscataway, NJ, 2004, pp. 335–341.

http://www.siam.org/journals/sicomp/37-1/44733.html

[†] School of Computer Science, the University of Oklahoma, Norman, OK 73019 (qcheng@cs.ou.edu). This author's work was partially supported by NSF Career Award CCR-0237845.

[‡] Department of Mathematics, University of California, Irvine, CA 92697 and Institute of Mathematics, Chinese Academy of Sciences, Beijing, People's Republic of China (dwan@math.uci.edu). The work of this author was partially supported by the NSF.

where two sequences differ. A good error-correcting code should have a large *minimum distance* d, which is defined to be the minimum Hamming distance between two distinct codewords in $\phi(\Sigma^k)$. A received message, possibly corrupted, but with no more than $(d-1)/2$ errors, corresponds to a unique codeword and thus may be decoded into the original message despite errors occuring during the communication.

Error-correcting codes are widely used in practice. They are mathematically interesting and intriguing. This subject has attracted the attention of the theoretical computer science community recently. Several major achievements in theoretical computer science, notably the original proof of the probabilistically checkable proof (PCP) theorem and derandomization techniques, rely heavily on the techniques found in error-correcting codes. We refer to the survey [19] for details.

For the purpose of efficient encoding and decoding, Σ is usually set to be the finite field $\mathbf{F}_q$ of q elements, and the map ϕ is $\mathbf{F}_q$-linear. Numerous error-correcting codes have been proposed; among them, the Reed–Solomon codes are particularly important. They are deployed to transmit information to and from spacecraft and to store information in optical media [22].

Notation. For a polynomial $f(x)$ and a set $S = \{x_1, \dots, x_n\}$, we use $(f(x))_{x\in S}$ to denote the vector obtained by evaluating $f(x)$ at the elements in S, that is,

$$(f(x))_{x\in S} = (f(x_1), \dots, f(x_n)).$$

Let S be a subset of $\mathbf{F}_q$ with $|S| = n$. The Reed–Solomon code $[n, k]_q$ is the map from $(a_0, a_1, \dots, a_{k-1}) \in \mathbf{F}_q^k$ to

$$(a_0 + a_1 x + \cdots + a_{k-1}x^{k-1})_{x\in S} \in \mathbf{F}_q^n.$$

The choice of S will not affect our results in this paper. Since any two different polynomials with degree $k-1$ can share at most $k-1$ points, the minimum distance of the Reed–Solomon code is $n-k+1$.

1.1. Related works. If the radius of a Hamming ball is less than half of the minimum distance, there is at most one codeword in the Hamming ball. Finding the codeword is called *unambiguous decoding*. It can be efficiently solved (see [2]) for a simple algorithm. If we gradually increase the radius, there may be two or more codewords lying in some Hamming balls. Can we efficiently enumerate all the codewords in any Hamming ball of a certain radius? This is the so-called list decoding problem. The notion was first introduced by Elias [6]. There was virtually no progress on this problem for radius slightly larger than half of the minimum distance until Sudan published his influential paper [18]. His result was subsequently improved; the current best algorithm [11] solves the list decoding problem for a radius as large as $n - \sqrt{n(k-1)}$. The work [11] sheds new light on the list decodability of Reed–Solomon codes. To the other extreme, if the radius is greater than or equal to the minimum distance, there are exponentially many codewords in some Hamming balls.

The decoding problem of Reed–Solomon codes can be reformulated into the problem of *curve fitting* or *polynomial reconstruction*. In this problem, we are given n points

$$(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$$

in $\mathbf{F}_q^2$. The goal is to find polynomials of degree $k-1$ that pass at least g points. In this paper, we consider only the case when the n given points have distinct x-coordinates. If we allow multiple occurrences of x-coordinates, the problem is NP-hard [7, Theorem 6.1], and it is not relevant to the Reed–Solomon decoding problem. If $g \geq (n+k)/2$, it corresponds to the unambiguous decoding of Reed–Solomon codes. If $g > \sqrt{n(k-1)}$, and the radius is less than $n - \sqrt{n(k-1)}$, which is essentially the Johnson radius [12, 10], the problem can be solved by the Guruswami–Sudan algorithm [11]. If $g \leq k$, it is possible that there are exponentially many solutions, but finding one is very easy.

It is known that any Hamming ball of the Johnson radius contains only polynomially many codewords. In this paper, we study the following question: By how much can we increase the radius before the list decoding problem or the bounded distance decoding problem becomes infeasible? This question has been intensively investigated for Reed–Solomon codes and other error-correcting codes. The case of general nonlinear codes has been solved in [7], where it was proved that there exist codes with exponentially many codewords in Hamming balls of radius slightly larger than the Johnson radius. The case for linear codes is much harder. Some partial results have been obtained in [9, 8], where it was proved that there exist linear codes with superpolynomially many codewords in Hamming balls of radius close to the Johnson radius. However, none of them apply to Reed–Solomon codes. No negative result is known about the list decoding of Reed–Solomon codes, except for a simple combinatorial bound given by Justesen and Hoholdt [13], which states that for any positive integer $g < n$, there exists at least one Hamming ball of radius $n - g$, which contains at least $\binom{n}{g}/q^{g-k}$ many codewords. This bound matches the intuition well. Consider an imaginary algorithm as follows: Randomly select g points from the n input points, and use polynomial interpolation to get a polynomial of degree at most $g - 1$ which passes these g points. Then with probability $1/q^{g-k}$, for a random word in $\mathbf{F}_q^n$, the resulting polynomial has degree $k - 1$. The sample space has size $\binom{n}{g}$. Thus heuristically, the number of codewords in Hamming balls of radius $n - g$ is at least $\binom{n}{g}/q^{g-k}$ on the average. In the same paper, Justesen and Hoholdt also gave an upper bound for the radius of the Hamming balls containing a constant or fewer number of codewords.

1.2. Our results. If we gradually increase g, starting from k and going toward n, then $\binom{n}{g}/q^{g-k}$ will fall below 1 at some point. However, g is still very far away from $\sqrt{n(k-1)}$. Let $\hat{g}(n,k,q)$ be the smallest positive integer such that $\binom{n}{g}/q^{g-k}$ is no greater than 1. Roughly speaking, a Hamming ball with a random center and the radius $n - \hat{g}(n,k,q)$ contains on average about one codeword. The following lemma shows that there is a gap between $\hat{g}(n,k,q)$ and $\sqrt{n(k-1)}$.

LEMMA 1. *For positive integers $k < g < n$, if $g > \sqrt{nk}$, then $q^{g-k} \geq n^{g-k} > \binom{n}{g}$. This implies that $\hat{g}(n,k,q) \leq \sqrt{nk}$.*

For a fixed rate k/n, the radius $n - \hat{g}(n,k,q)$ has a relative radius approaching the relative distance as n approaches infinity. However, it is not known whether there exist Reed–Solomon codes such that some Hamming balls of radius $n - \hat{g}(n,k,q)$ contain an exponential number of codewords. Then how hard is it to do list decoding for the radius $n - \hat{g}(n,k,q)$?

Instead of trying to find a Hamming ball with a large number of codewords of radius $n - \hat{g}(n,k,q)$, we take another approach. By relating this question to the discrete logarithm over finite fields, we show that even if there is only a small number of codewords in every Hamming ball of this radius, the list decoding problem is still infeasible. The discrete logarithm problem in finite field $\mathbf{F}_{q^m}$ is to compute an integer e such that $t = \gamma^e$, given a generator γ of a subgroup of $\mathbf{F}_{q^m}^*$ and t in the subgroup. The general purpose algorithms for solving the discrete logarithm problem are the number field sieve and the function field sieve (for a survey see [16]). They have a conjectured subexponential time complexity

$$\exp(c(\log q^m)^{1/3}(\log\log q^m)^{2/3})$$

for some constant c when q is small or m is small.

We prove that if the list decoding of the $[n,k]_q$ Reed–Solomon code is feasible for radius $n-\hat{g}(n,k,q)$, then the discrete logarithm over $\mathbf{F}_{q^{\hat{g}(n,k,q)-k}}$ is easy. In other words, we prove that the list decoding is not feasible for radius $n-\hat{g}(n,k,q)$ or larger, assuming that the discrete logarithm over $\mathbf{F}_{q^{\hat{g}(n,k,q)-k}}$ is hard. Note that it does not rule out the possibility that there are only polynomially many codewords in all Hamming balls of radius $n-\hat{g}(n,k,q)$, even assuming the intractability of the discrete logarithm over $\mathbf{F}_{q^{\hat{g}(n,k,q)-k}}$.

THEOREM 1. *If there exists an algorithm solving the list decoding problem of radius $n-\hat{g}(n,k,q)$ for the Reed–Solomon code $[n,k]_q$ in random time $q^{O(1)}$, then a discrete logarithm over the finite field $\mathbf{F}_{q^{\hat{g}(n,k,q)-k}}$ can be computed in random time $q^{O(1)}$.*

Let us consider a numerical example. Set $n=1000$, $k=401$, $q=1201$. The unambiguous decoding algorithm can correct up to $\lfloor(n-k+1)/2\rfloor = 300$ errors. The Guruswami–Sudan algorithm can correct $\lfloor n-\sqrt{n(k-1)}\rfloor = \lfloor 1000-\sqrt{1000*400}\rfloor =$ 368 errors. Can we list decode up to $n-\hat{g}(n,k,q) = 1000-499 = 501$ errors in a reasonable amount of time? The theorem shows that if we can, then the discrete logarithm over $\mathbf{F}_{1201^{98}}$ can be solved efficiently, which is widely regarded as unlikely at present.

When the list decoding problem is hard for certain radius, or a Hamming ball contains too many codewords for us to enumerate all of them, we can ask for an efficient *bounded distance decoding* algorithm, which needs only to output one of the codewords in the ball, or output the empty set in case the ball does not contain any codeword. However, we prove that the bounded distance decoding is hard as well.

THEOREM 2. *Let q be a prime power and h be a positive integer satisfying $q \geq \max(g^2,(h-1)^{2+\epsilon})$ and $g \geq (\frac{4}{\epsilon}+2)(h+1)$ for any constant $\epsilon > 0$. If the bounded distance decoding problem of radius $q-g$ for the Reed–Solomon code $[q,g-h]_q$ can be solved in random time $q^{O(1)}$, the discrete logarithm problem over $\mathbf{F}_{q^h}$ can be solved in random time $q^{O(1)}$.*

For q,g,h satisfying the conditions in the theorem,

$$\frac{\binom{q}{g}}{q^{g-(g-h)}} \geq \frac{(q/g)^g}{q^h} = \frac{q^{g-h}}{g^g} \geq \frac{(g^2)^{g-h}}{g^g} = g^{g-2h} \geq g^{4h/\epsilon}.$$

Hence there is a Hamming ball of radius $q-g$ containing exponentially many codewords. It is infeasible to do list decoding under these parameters. This result has the drawback that it can be applied only to the low rate codes, since $g-h \leq g \leq \sqrt{q}$.

It is generally believed that the list decoding problem and the bounded distance decoding for Reed–Solomon codes are computationally hard if the number of errors is greater than $n-\sqrt{n(k-1)}$ and less than $n-k$. This problem is even used as a hard problem to build public key cryptosystems and pseudorandom generators [15]. A similar problem, noisy polynomial interpolation [3], was proved to be vulnerable to the attack of lattice reduction techniques, and hence is easier than originally thought. This raises concerns about the hardness of the polynomial reconstruction problem. Our results confirm the belief that the polynomial reconstruction problem is hard for certain parameters, under a well-studied hardness assumption in number theory, and hence provide a guideline for selecting parameters for many protocols based on the problem.

1.3. Techniques. We rely on the idea of index calculus to prove these two theorems. Our application of index calculus, however, is different from its usual applications in that we use it to prove a hardness result (a computational lower bound) rather than a computational upper bound. We naturally come across the following question in the proofs: In a finite field $\mathbf{F}_{q^h}$, for any α such that $\mathbf{F}_{q^h} = \mathbf{F}_q[\alpha]$, can $\mathbf{F}_q + \alpha$ generate the multiplicative group $(\mathbf{F}_{q^h})^*$? This interesting problem has many applications in graph theory, and it has been studied by several number theorists. Chung [5] proved that if $q > (h-1)^2$, then $(\mathbf{F}_{q^h})^*$ is generated by $\mathbf{F}_q + \alpha$. Wan [21] showed a negative result that if $q^h - 1$ has a divisor $d > 1$ and $h \geq 2(q \log_q d + \log_q(q+1))$, then $(\mathbf{F}_{q^h})^*$ is not generated by $\mathbf{F}_q + \alpha$ for some α. Katz [14] applied the Lang–Weil method and showed that for every $h \geq 2$ there exists a constant $B(h)$ such that for any finite field $\mathbf{F}_q$ with $q \geq B(h)$, any element in $(\mathbf{F}_{q^h})^*$ can be written as a product of exactly $n = h + 2$ distinct elements from $\mathbf{F}_q + \alpha$. By a simple counting argument, $B(h)$ has to be an exponential function in h. In this paper, we use Weil's character sum estimate and a simple sieving to prove that if $q \geq \max(g^2, (h-1)^{2+\epsilon})$ and $g \geq (\frac{4}{\epsilon} + 2)(h+1)$ for any constant $\epsilon > 0$, then any element in $(\mathbf{F}_{q^h})^*$ can be written as a product of exactly g distinct elements from $\mathbf{F}_q + \alpha$. In comparison to Katz's theorem, we use a bigger n and manage to decrease $B(h)$ to a polynomial function in h.

This paper is organized as follows. In section 2, we show the connection between the decoding problem of Reed–Solomon codes and the discrete logarithm problem over finite fields. In section 3, we present the proof of Theorem 1. In section 4, we present the proof of Theorem 2. In section 5, we show an interesting duality between the size of a group generated by linear factors and the list size in Hamming balls of Reed–Solomon codes. In the appendix, we prove Lemma 1.

2. The decoding problem and the discrete logarithm. Let q be a prime power and let $\mathbf{F}_q$ be the finite field with q elements. Let S be a subset of $\mathbf{F}_q$ of n elements. For a positive integer $g \leq n$, denote

$$\mathcal{P}(S, g) = \{A | A \subseteq S, |A| = g\}.$$

Clearly, the set $\mathcal{P}(S, g)$ has $\binom{n}{g}$ elements. For any $A \in \mathcal{P}(S, g)$, let

$$P_A(x) = \prod_{a \in A}(x - a).$$

This is a monic polynomial of degree g which splits over $\mathbf{F}_q$ as a product of distinct linear factors.

Let $1 < h < g$ be integers. Let $h(x)$ be an irreducible monic polynomial over $\mathbf{F}_q$ of degree h. Define a map

$$\psi : \mathcal{P}(S, g) \to \mathbf{F}_q[x]/(h(x))$$

by

$$\psi(A) = P_A(x) \pmod{h(x)}.$$

For any $f(x)$ in $\mathbf{F}_q[x]/(h(x))$ with degree at most $h - 1$, if $\psi^{-1}(f(x))$ is not empty, then there exists at least one monic polynomial $t(x) \in \mathbf{F}_q[x]$ of degree $g - h$ and one $A \in \mathcal{P}(S, n)$ such that

$$f(x) + t(x)h(x) = P_A(x).$$

For any $a \in A$, $P_A(a) = 0$ and $t(a) = -f(a)/h(a)$. Recall that $h(x)$ is irreducible over $\mathbf{F}_q$, and hence $h(a) \neq 0$ for all $a \in A$. Since $f(x) + t(x)h(x)$ has degree g, there are exactly g elements in S which are the roots of $f(x)+t(x)h(x) = 0$; the curve $y = t(x)$ passes exactly g points in the following set of n points:

$$\{(a, -f(a)/h(a))|a \in S\}.$$

For any polynomial $f \in \mathbf{F}_q[x]$ of degree at most $h - 1$, let $T_{f(x)}$ be the set of monic polynomials $t(x) \in \mathbf{F}_q[x]$ of degree $g - h$ such that $f(x) + t(x)h(x) = P_A(x)$ for some $A \in \mathcal{P}(S, g)$. Let $C_{f(x)}$ be the set of codewords with distance exactly $n - g$ to the received word $(-f(a)/h(a) - a^{g-h})_{a\in S}$ in Reed–Solomon code $[n, g - h]_q$. It is then easy to prove the following lemma.

Lemma 2. *There is a one-to-one correspondence between elements of* $T_{f(x)}$ *and* $C_{f(x)}$*, by sending any* $t(x) \in T_{f(x)}$ *to* $(t(a) - a^{g-h})_{a\in S}$.

Remark 1. According to the pigeonhole principle, there must exist a polynomial $\hat{f}(x)$ such that

$$|\psi^{-1}(\hat{f}(x))| \geq \frac{|\mathcal{P}(S, g)|}{|\mathbf{F}_q[x]/(h(x))|} = \frac{\binom{n}{g}}{q^h}.$$

This provides another proof that there is a Hamming ball of radius $n - g$ with $\frac{\binom{n}{g}}{q^h}$ many codewords.

Suppose that we know $f(x)$ and $h(x)$, but not $t(x)$; can we still find A? Formally we are asking the following question.

Input: A prime power q, an irreducible polynomial $h(x)$ over $\mathbf{F}_q$ of degree h, a polynomial $f(x) \in \mathbf{F}_q[x]$, a positive integer g, and a set $S \subseteq \mathbf{F}_q$.

Problem I: A list of all the subsets $A \in \mathcal{P}(S, g)$ such that

$$f(x) \equiv P_A(x) \pmod{h(x)}.$$

Problem II: One of $A \in \mathcal{P}(S, g)$ such that

$$f(x) \equiv P_A(x) \pmod{h(x)}.$$

Lemma 3. *Problem* I *can be reduced in polynomial time to the list decoding problem of Reed–Solomon code* $[n, g - h]_q$ *at radius* $n - g$*. Problem* II *can be reduced in polynomial time to the bounded distance decoding problem of Reed–Solomon code* $[n, g - h]_q$ *at radius* $n - g$.

Proof. The vector $(-f(a)/h(a) - a^{g-h})_{a\in S}$ can be calculated from the input. Using the list decoding algorithm or the bounded distance decoding algorithm, we can compute $t(x)$ of degree at most $g - h - 1$ such that $t(a) = -f(a)/h(a) - a^{g-h}$ at g many a's. We find A by factoring $f(x) + (t(x) + x^{g-h})h(x)$. □

If A can be found, then in the field $\mathbf{F}_q[x]/(h(x))$, $f(x)$ can be represented as a product of elements from a small set. It is called a *smooth* representation with *factor bases* $x - S$ in computational number theory. The capability of finding a smooth representation constitutes a powerful attack against hard number theory problems like integer factorization and the discrete logarithm over finite fields. The lemma implies that decoding Reed–Solomon codes provides a way to find a smooth representation of any field element. Thus naturally an efficient decoding algorithm produces an attack for the discrete logarithm over finite fields. This idea first appeared in [4], and it provides a general framework for the following proofs.

3. The proof of Theorem 1. Given a Reed–Solomon code $[n,k]_q$, let $h = \hat{g}(n,k,q) - k$. Recall that $\hat{g}(n,k,q)$ is the smallest positive integer such that $\binom{n}{g}/q^{g-k}$ is no greater than 1, and h is the degree of an irreducible polynomial $h(x)$. We show that there is an efficient algorithm for solving the discrete logarithm over $\mathbf{F}_{q^h} = \mathbf{F}_q[x]/(h(x))$ if there is an efficient list decoding algorithm for the Reed–Solomon code $[n,k]_q$ with radius $n - \hat{g}(n,k,q) = n - k - h$.

Let $\alpha = x \pmod{h(x)}$. Suppose that we are given the base $b(\alpha)$ and that we need to find the discrete logarithm of $v(\alpha)$ with respect to the base, where b and v are polynomials over $\mathbf{F}_q$ of degree at most $h-1$. Select any $S \subseteq \mathbf{F}_q$, $|S| = n$. We use the index calculus algorithm with factor bases $(\alpha - a)_{a \in S}$.

ALGORITHM 1.

1. *Initialize an empty set of linear equations.*
2. *Repeat n times:*
   - (a) *Randomly select an integer i between* 0 *and* $q^h - 2$. *Compute* $f(x) = b(x)^i \pmod{h(x)}$.
   - (b) *Apply the list decoding algorithm to find the list of* $A \in \mathcal{P}(S, \hat{g}(n,k,q))$ *such that* $f(x) \equiv P_A(x) \pmod{h(x)}$. *If the list is empty, go back to* 2(a).
   - (c) *Otherwise we have* relations
     $$f(\alpha) = \prod_{a \in A_1} (\alpha - a) = \cdots = \prod_{a \in A_l} (\alpha - a)$$
     for some $A_1, A_2, \ldots, A_l \in \mathcal{P}(S, \hat{g}(n,k,q))$, *where l is the list size. From the relations, we obtain linear equations* mod $(q^h - 1)$:
     $$i \equiv \sum_{a \in A_1} \log_{b(\alpha)}(\alpha - a) \equiv \cdots \equiv \sum_{a \in A_l} \log_{b(\alpha)}(\alpha - a).$$
     Add them to the set of linear equations.
3. *For all $s \in S$ do:*
   - (a) *Randomly select an integer i between* 0 *and* $q^h - 2$. *Compute* $f(x) = b(x)^i/(x - s) \pmod{h(x)}$.
   - (b) *Apply the list decoding algorithm to find the list of* $A \in \mathcal{P}(S, \hat{g}(n,k,q))$ *such that* $f(x) \equiv P_A(x) \pmod{h(x)}$. *If the list is empty, go back to* 3(a).
   - (c) *Otherwise we have* relations
     $$f(\alpha) = \prod_{a \in A_1} (\alpha - a) = \cdots = \prod_{a \in A_l} (\alpha - a)$$
     for some $A_1, A_2, \ldots, A_l \in \mathcal{P}(S, \hat{g}(n,k,q))$, *where l is the list size. From the relations, we obtain linear equations* mod $(q^h - 1)$:
     $$\begin{aligned} i &\equiv \sum_{a \in A_1} \log_{b(\alpha)}(\alpha - a) + \log_{b(\alpha)}(\alpha - s) \equiv \cdots \\ &\equiv \sum_{a \in A_l} \log_{b(\alpha)}(\alpha - a) + \log_{b(\alpha)}(\alpha - s). \end{aligned}$$
     Add them to the set of linear equations.
4. *In these equations,* $\log_{b(\alpha)}(\alpha - a)$, $a \in S$, *are unknowns. If the system has full rank, solve it; otherwise go back to step* 2.

5. *Randomly select an integer i between 0 and $q^h - 2$. Compute $f(x) = b(x)^i v(x) \pmod{h(x)}$.*
6. *Apply the list decoding algorithm to find a list of $A \in \mathcal{P}(S, g)$ such that $f(x) \equiv P_A(x) \pmod{h(x)}$. If the list is empty, go back to step 5.*
7. *Otherwise we have a relation*

$$f(\alpha) = \prod_{a \in A} (\alpha - a).$$

Hence

$$i + \log_{b(\alpha)} v(\alpha) = \sum_{a \in A} \log_{b(\alpha)} (\alpha - a);$$

we can solve $\log_{b(\alpha)} v(\alpha)$ immediately.

Now we analyze the time complexity of the algorithm. An efficient list decoding algorithm implies the following:

1. There are only polynomially many codewords in any Hamming ball of radius $n - \hat{g}(n, k, q)$, which along with Lemma 2 implies that $|\psi^{-1}(f)| \leq q^c$ for any $f \in \mathbf{F}_{q^h}$ and a constant c. Hence

$$|\psi(\mathcal{P}(S, \hat{g}(n, k, q)))| \geq \frac{\binom{n}{\hat{g}(n,k,q)}}{q^c} \geq \frac{q^{\hat{g}(n,k,q)-k}}{q^c} = \frac{q^h}{q^c}.$$

Thus in steps 2(b), 3(b), and 6, since $f(\alpha)$ is a random element in $\mathbf{F}_{q^h}^*$, the list decoding algorithm outputs a nonempty list with probability bigger than $1/q^c$. Note that $\frac{1}{q} \leq \frac{\binom{n}{\hat{g}(n,k,q)}}{q^{\hat{g}(n,k,q)-k}} \leq 1$.

2. These codewords can be found in polynomial time. Each step will take polynomial time. Thus all steps in the algorithm run in polynomial time.

So we only need to show the following.

Lemma 4. *The linear system can yield a unique solution with high probability after polynomially many iterations of the main loop (from step 2 to step 4).*

Informally since i is picked randomly, the probability that a new equation is linearly independent to previous ones is very high at the beginning of the algorithm. It would not be a long time before we have an independent linear system. Solving the system of equations gives us $\log_{b(\alpha)}(\alpha - a)$ for all $a \in S$.

Proof of Lemma 4. The linear system is defined in the ring $\mathbf{Z}/(q^h - 1)$, which is usually not a field. We may proceed with the linear system solver. If the algorithm encounters a zero-divisor in the ring, we can factor $q^h - 1$. We then apply the linear system solver to each modulus. We get the solution for the original system using the Chinese remainder theorem. In the case that a modulus is a prime power, we can solve the linear system modulo the prime first and use Hensel lifting to solve the system modulo the prime power. Since $q^h - 1$ has at most $h \log q$ many distinct prime factors, this issue will slow down the algorithm by only a polynomial factor. Now we may assume that the linear system is defined over a finite field.

Let $w = \lceil 2 \log n \rceil + 3$. Let T be the set of binary vectors with length n and weight g. For the case when the iteration of the main loop is repeated w times, we have selected in step 2(a) nw many integers $i_1, i_2, \ldots, i_{nw}$, and in step 3(a) nw many integers $i_{nw+1}, \ldots, i_{2nw}$, and obtained relations for $b^{i_j}(\alpha)$ $(1 \leq j \leq nw)$ and $b^{i_j}(\alpha)/(\alpha - a)$ $(nw + 1 \leq j \leq 2nw$ and $a \in S)$. This amounts to selecting at least $2nw$

vectors from T independently. Also $\{\log_{b(\alpha)}(\alpha - s) | s \in S\}$ forms a basis for the linear system. According to the following proposition, proved in [17], we get n independent equations with probability more than $1 - \frac{1}{2n}$. Note that it is *not* required that the vectors be selected uniformly. This finishes the proof of Theorem 1. ∎

PROPOSITION 1. *Let V be a vector space over a field $\mathbf{F}$ with $\dim V = n$. Let T be a finite set of vectors in V and let $e_1, \ldots, e_n$ be a basis of V. Let $w = \lceil 2\log n \rceil + 3$. Suppose we choose $2nw$ vectors $u_1, u_2, \ldots, u_{nw}, v_1, v_2, \ldots, v_{nw}$ independently from T. Let V' be the subspace of V spanned by $u_1, u_2, \ldots, u_{nw}$ and the vectors $e_j + v_{(j-1)w+i}$ for $j = 1, 2, \ldots, n$ and $i = 1, \ldots, w$. Then with probability at least $1 - \frac{1}{2n}$, we have that $V = V'$.*

4. The proof of Theorem 2. We first prove the following number theoretic result.

THEOREM 3. *Let h be a positive integer. Assume $q \geq \max(g^2, (h-1)^{2+\epsilon})$ and $g \geq (\frac{4}{\epsilon} + 2)(h+1)$ for a constant $\epsilon > 0$. Then for any α such that $\mathbf{F}_q[\alpha] = \mathbf{F}_{q^h}$, every element in $\mathbf{F}^*_{q^h}$ can be written as a product of exactly g distinct factors from $\{\alpha + a | a \in \mathbf{F}_q\}$.*

Proof. We follow the method used in [21]. Fix an α such that $\mathbf{F}_q[\alpha] = \mathbf{F}_{q^h}$. For $\beta \in \mathbf{F}^*_{q^h}$, let $N_g(\beta)$ denote the number of solutions of the equation

$$\beta = \prod_{i=1}^{g} (\alpha + a_i), \ a_i \in \mathbf{F}_q,$$

where the a_i's are distinct. Permutations of a_i's are counted as different solutions. We need to show that the number $N_g(\beta)$ is always positive if $q \geq \max(g^2, (h-1)^{2+\epsilon})$ and $g \geq (\frac{4}{\epsilon} + 2)(h+1)$.

Let G be the character group of the multiplicative group $\mathbf{F}^*_{q^h}$. That is, G is the set of group homomorphisms from $\mathbf{F}^*_{q^h}$ to $\mathbf{C}^*$. The group G is a cyclic group of order $q^h - 1$, and thus

$$\sum_{\chi \in G} \chi\left(\prod_{i=1}^{g} (\alpha + a_i)/\beta \right) = \begin{cases} q^h - 1 & \text{if } \beta = \prod_i (\alpha + a_i), \\ 0 & \text{otherwise.} \end{cases}$$

From this, we deduce

$$N_g(\beta) = \frac{1}{q^h - 1} \sum_{a_i \in \mathbf{F}_q, \ a_i \text{ distinct}} \sum_{\chi \in G} \chi^{-1}(\beta) \chi\left(\prod_{i=1}^{g} (\alpha + a_i) \right).$$

Since the second summand is always nonnegative, a simple inclusion-exclusion sieving implies that

$$N_g(\beta) \geq \frac{1}{q^h - 1} \left(\sum_{a_i \in \mathbf{F}_q, 1 \leq i \leq g} - \sum_{1 \leq i_1 < i_2 \leq g} \sum_{a_i \in \mathbf{F}_q, a_{i_1} = a_{i_2}} \right)$$
$$\sum_{\chi \in G} \chi^{-1}(\beta) \chi\left(\prod_{i=1}^{g} (\alpha + a_i) \right).$$

For the nontrivial character χ, one has the well-known Weil estimate [21]

$$\left| \sum_{a \in \mathbf{F}_q} \chi(\alpha + a) \right| \leq (h-1)\sqrt{q},$$

and thus

$$\left|\sum_{a_i\in\mathbf{F}_q,1\leq i\leq g}\prod_{i=1}^{g}\chi(\alpha+a_i)\right|\leq(h-1)^g q^{g/2}.$$

If $\chi^2\neq 1$, then for fixed $i_1<i_2$, Weil's estimate implies that

$$\left|\sum_{a_i\in\mathbf{F}_q,a_{i_1}=a_{i_2}}\prod_{i=1}^{g}\chi(\alpha+a_i)\right|\leq(h-1)^{g-1}q^{(g-1)/2}\leq(h-1)^g q^{g/2}.$$

If $\chi^2=1$ but $\chi\neq 1$, then for fixed $i_1<i_2$, Weil's estimate implies that

$$\left|\sum_{a_i\in\mathbf{F}_q,a_{i_1}=a_{i_2}}\prod_{i=1}^{g}\chi(\alpha+a_i)\right|\leq q(h-1)^{g-2}q^{(g-2)/2}\leq(h-1)^g q^{g/2}.$$

Separating the trivial character from the above lower estimate for $N_g(\beta)$, we deduce that

$$N_g(\beta)\geq\frac{q^g-\binom{g}{2}q^{g-1}}{q^h-1}-\left(1+\binom{g}{2}\right)(h-1)^g q^{g/2}.$$

In order for $N_g(\beta)>0$, it suffices to have the inequality

$$\left(q-\binom{g}{2}\right)q^{g/2-1-h}>\left(1+\binom{g}{2}\right)(h-1)^g.$$

This inequality is clearly satisfied if both $q>2\binom{g}{2}+1=g(g-1)+1$ and $q^{g/2-1-h}>(h-1)^g$. These two inequalities are satisfied if we take $q\geq\max(g^2,(h-1)^{2+\epsilon})$ and $g\geq(\frac{4}{\epsilon}+2)(h+1)$. The theorem is proved. □

Remark 2. As we should see immediately, the expression $N_g(\beta)/g!$ is the number of codewords in the Hamming ball with radius $n-g$ and with center at $(-f(a)/h(a)-a^{g-h})_{a\in\mathbf{F}_q}$, where $h(x)$ is the minimum polynomial of α over $\mathbf{F}_q$ and $f(x)$ is the polynomial of degree at most $h-1$ representing β. Adjusting the parameters will give us an exponential lower bound for $N_g(\beta)/g!$. For example, if $q>2\binom{g}{2}+2$ and $g\geq(\frac{4}{\epsilon}+2)(h+1)$, the same proof shows that $N_g(\beta)\geq q^{g-h-1}$, and thus

$$\frac{N_g(\beta)}{g!}\geq\frac{q^{g-h-1}}{g!},$$

which is exponential for certain parameters q, g, and h.

Let $q\geq\max(g^2,(h-1)^{2+\epsilon})$ and $g\geq(\frac{4}{\epsilon}+2)(h+1)$. Let $h(x)$ be an irreducible polynomial over $\mathbf{F}_q$ of degree h. Then $\mathbf{F}_{q^h}=\mathbf{F}_q[x]/(h(x))$. Denote $x \pmod{h(x)}$ by α. Suppose that there exists a polynomial time algorithm for bounded distance decoding of Reed–Solomon code $[q,g-h]_q$ at radius $g-q$. We prove Theorem 2 by describing a polynomial time algorithm for solving the discrete logarithm of $v(\alpha)$ with base $b(\alpha)$ in $\mathbf{F}_{q^h}$, where b and v are polynomials of degree at most $h-1$. We let $S=\mathbf{F}_q$. This algorithm relies on the index calculus idea as in Algorithm 1. It is simpler, as for any polynomial $f(x)\in\mathbf{F}_q[x]$ with degree $g-h-1$ or less, if we run the bounded decoding algorithm with input word $\{(x,-f(x)/h(x))|x\in\mathbf{F}_q\}$ and distance bound $n-g$ (the answer is never empty) according to Theorem 3 and Lemma 2.

ALGORITHM 2.

1. *Initialize an empty set of linear equations.*
2. *Repeat n times:*
   (a) *Randomly select an integer i between 0 and $q^h - 2$.*
   (b) *Compute $b(\alpha)^i$, and let $f(x) = b^i(x) \pmod{h(x)}$.*
   (c) *Run the bounded distance decoding algorithm to find $A = \mathcal{P}(\mathbf{F}_q, g)$ such that*
   $$b(\alpha)^i = \prod_{a \in A} (\alpha - a).$$
   We have
   $$i \equiv \sum_{a \in A} \log_{b(\alpha)}(\alpha - a) \pmod{q^h - 1}.$$
   Add it to the set of linear equations.
3. *For all $s \in S$ do:*
   (a) *Randomly select an integer i between 0 and $q^h - 2$. Let $f(x)$ denote the element $b(x)^i/(x - s) \pmod{h(x)}$.*
   (b) *Run the bounded distance decoding algorithm to find $A \in \mathcal{P}(\mathbf{F}_q, g)$ such that*
   $$b(\alpha)^i/(\alpha - s) = \prod_{a \in A} (\alpha - a).$$
   We get
   $$i \equiv \sum_{a \in A} \log_{b(\alpha)}(\alpha - a) + \log_{b(\alpha)}(\alpha - s) \pmod{q^h - 1}.$$
   Add it to the set of linear equations.
4. *In these equations, $\log_{b(\alpha)}(\alpha - a)$, $a \in S$, are unknowns. If the system has full rank, solve it; otherwise go back to step 2.*
5. *Apply the bounded distance decoding algorithm to find relation*
   $$v(\alpha) = \prod_{a \in A} (\alpha - a).$$
   Hence
   $$\log_{b(\alpha)} v(\alpha) = \sum_{a \in A} \log_{b(\alpha)}(\alpha - a).$$

We can essentially copy the proof of Lemma 4 to prove that we need only try $O(n \log n)$ many i's before we solve the discrete logarithm of $v(\alpha)$ with base $b(\alpha)$ with probability $1 - \frac{1}{2n}$. It is very crucial here that we have step 3, because the system may not have the full rank if we have only step 2. This is the case if all the A_i's in step 2 come from a subset of $\mathbf{F}_q$. An easy consequence of Theorem 2 is as follows. Taking $\epsilon = 2$ and $g = 4h + 4$ in Theorem 2, we get the following corollary.

COROLLARY 1. *Let q be a prime power and let h be a positive integer satisfying $q > \max((4h + 4)^2, (h - 1)^4)$. If the bounded distance decoding problem of radius $q - 4h - 4$ for the Reed–Solomon code $[q, 3h + 4]_q$ can be solved in time $q^{O(1)}$, the discrete logarithm problem over $\mathbf{F}_{q^h}$ can be solved in random time $q^{O(1)}$.*

5. Group size and list size. Let S be a subset of $\mathbf{F}_q$ of n elements. Let α be an element in $\mathbf{F}_{q^h}$ such that $\mathbf{F}_q[\alpha] = \mathbf{F}_{q^h}$ and h is very small compared to q. What is the order of the subgroup generated by $\alpha + S$? This question has an important application in analyzing the performance of the Agrawal–Kayal–Saxena (AKS) primality testing algorithm [1]. Experimental data suggest that the order is greater than $q^{h/c}$ for some absolute constant c for all $|S| \geq h \log q$. If it can be proved, the space complexity of the AKS algorithm can be cut by a factor of $\log p$ (p is the input prime whose primality certificate is sought), which will make (the random variants of) the algorithm comparable to the primality proving algorithm used in practice. However, the best known lower bound is $(c|S|/h)^h$ for some absolute constant c [20]. We present an interesting duality between the group size and the list size in Hamming balls of certain radius.

THEOREM 4. *Let q be a prime power. Let α be an element in the extension of $\mathbf{F}_q$ with degree h. Let $\omega_\alpha(n)$ be the smallest possible order for the group generated by $\alpha + S$ multiplicatively, where $S \in \mathbf{F}_q$ and $|S| = n$. Let $A_q^{RS}(n, d, w)$ be the maximum list size in the Hamming balls of radius w in any Reed–Solomon code with block length n and minimum distance d over $\mathbf{F}_q$. For any integer $k < n - h$, we have*

$$A_q^{RS}(n, n-k, n-k-h) \times \omega_\alpha(n) \geq \binom{n}{k+h}.$$

Proof. Let $\omega_\alpha(n, S)$ be the order of the group generated by $\alpha + S$, where $S \in \mathbf{F}_q$ and $|S| = n$. Let $h(x)$ be the minimum polynomial of α. Consider the mapping

$$\psi : \mathcal{P}(S, k+h) \to \mathbf{F}_q[x]/(h(x)).$$

The range of ψ consists of elements which can be represented as products of $k + h$ distinct elements in $\alpha + S$; thus it has cardinality at most $\omega_\alpha(n, S)$. For any element in $\mathbf{F}_q[x]/(h(x))$, the number of its preimages is at most $A_q^{RS}(n, n-k, n-k-h)$, according to Lemma 2. Hence

$$A_q^{RS}(n, n-k, n-k-h) \times \omega_\alpha(n, S) \geq |\mathcal{P}(S, k+h)| = \binom{n}{k+h}.$$

This implies the theorem. □

COROLLARY 2. *Let k, n be positive integers and q be a prime power. One of the following statements must be true:*

1. *For any constant c_1, there exists a Reed–Solomon code $[n, k]_q$ ($n/3 < k < n/2$) and a Hamming ball of radius $n - \hat{g}(n, k, q)$ containing more than $c_1 1.9^n$ codewords.*
2. *The group generated by $\alpha + S$ has cardinality at least q^{h/c_2} for some absolute constant c_2, where $S \subseteq \mathbf{F}_q$ and $|S| = \lfloor h \log q \rfloor$.*

To prove or disprove the first statement would solve an important open problem about the Reed–Solomon codes. Recall that a Hamming ball with a random center and the radius $n - \hat{g}(n, k, q)$ contains on average one codeword. To prove the second statement would give us a primality proving algorithm much more efficient in terms of space complexity than the original AKS and its random variants, hence making the AKS algorithm not only theoretically interesting, but also practically important. However, at this stage we cannot figure out which one is true. What we can prove, however, is that one of them must be true. Note that it is also possible that both statements are true.

Proof of Corollary 2. Let $k = \lfloor h \log q/2 \rfloor - h$ and $n = \lfloor h \log q \rfloor$. Thus the rate k/n is very close to $1/2$ as q gets large. Since $\binom{n}{\lfloor h \log q/2 \rfloor}$ is about $2^{h \log q} = q^h = q^{h \log q/2 - k}$, $\hat{g}(n,k,q) = h \log q/2 + O(1)$. Assume the first statement is false; this means that there exists a constant c_3 such that for any Reed–Solomon code $[n,k]_q$ with $n/3 < k < n/2$, the number of codewords in any Hamming ball of radius $n - \hat{g}(n,k,q)$ is less than $c_3 1.9^n$. That is,

$$A_q^{RS}(n, n-k-h, n-\hat{g}(n,k,q)) \le c_3 1.9^n.$$

Hence the size of the group generated by $\alpha + S$ is at least

$$\frac{\binom{n}{\hat{g}(n,k,q)}}{c_3 1.9^n} \ge \frac{q^{h+O(1)}}{c_4 1.9^n} = \frac{q^{h - n\log 1.9/\log q + O(1)}}{c_4} \ge q^{h/c_2}. \qquad \square$$

6. Open problems. There is a large gap between $n - \sqrt{n(k-1)}$ and $n - \hat{g}(n,k,q)$, where we do not know whether list decoding for Reed–Solomon codes $[n,k]_q$ is feasible or not. In Theorem 3, the condition $q \ge g^2$ is still quadratic. It would be very interesting to obtain positive results with only linear condition $q \ge cg$ for some positive constant c. Other interesting open questions include whether there exists a reversal reduction which maps the list or bounded distance decoding problem of Reed–Solomon codes for the parameters studied in the paper to discrete logarithm over finite fields, and whether there exists a polynomial time quantum algorithm to solve these decoding problems.

Appendix. Proof of Lemma 1. In this section, we prove Lemma 1 by showing the following statement.

THEOREM 5. *There are no positive integral solutions for the inequalities*

$$\binom{n}{g} > n^h, \tag{1}$$

$$g > \sqrt{n(g-h)}. \tag{2}$$

We first obtain a finite range for h, g, and n.

LEMMA 5. *If (n,g,h) is a positive integral solution, then $h < 88$.*

Proof. Denote g/h by α and n/h by β. From $g > \sqrt{n(g-h)}$, we have $\alpha > \sqrt{\beta(\alpha-1)}$. Hence $\alpha < \beta < \alpha + 1 + \frac{1}{\alpha-1}$.

Recall that for any positive integer i, $\sqrt{2\pi i}(\frac{i}{e})^i \le i! \le \sqrt{2\pi i}(\frac{i}{e})^i(1 + \frac{1}{12i-1})$. We have also

$$\binom{n}{g} = \binom{\beta h}{\alpha h} \le \left(\frac{\beta^\beta}{\alpha^\alpha(\beta-\alpha)^{\beta-\alpha}}\right)^h.$$

Thus $\frac{\beta^\beta}{\alpha^\alpha(\beta-\alpha)^{\beta-\alpha}} \ge \beta h$, which implies

$$h \le \frac{\beta^{\beta-1}}{\alpha^\alpha(\beta-\alpha)^{\beta-\alpha}}.$$

We recall the following additional facts:
1. For $x > 0$, x^x takes the minimum value $0.6922\ldots$ at $x = e^{-1} = 0.36787944\ldots$.
2. For $x > 0$, $1 \le (1 + \frac{1}{x})^x \le e = 2.7182818284\ldots$.
If $\alpha \ge 2$, then $\beta - \alpha \le 1 + \frac{1}{\alpha - 1} \le 2$. We have

$$\begin{aligned}
h &\le \frac{1.45\beta^{\beta-1}}{\alpha^\alpha} \\
&\le \frac{1.45(1+\alpha+\frac{1}{\alpha-1})^{(\alpha+\frac{1}{\alpha-1})}}{\alpha^\alpha} \\
&\le 1.45\left(1+\alpha+\frac{1}{\alpha-1}\right)^{(\frac{1}{\alpha-1})}\left(1+\frac{1}{\alpha}+\frac{1}{\alpha(\alpha-1)}\right)^{\alpha} \\
&\le 1.45 * 4 * \left(1+\frac{1}{\alpha-1}\right)^{\alpha-1}\left(1+\frac{1}{\alpha-1}\right) \\
&\le 1.45 * 4 * e * 2 < 32.
\end{aligned}$$

If $\alpha < 2$, $h \le \frac{1.45\beta^{\beta-1}}{(\beta-\alpha)^{\beta-\alpha}}$. There are two cases. If $\beta \le 3$, then

$$h \le 1.45^2 * 9 < 19.$$

If $\beta > 3$, then

$$\begin{aligned}
h &\le 1.45\left(\frac{\beta}{\beta-\alpha}\right)^{\beta-1}(\beta-\alpha)^{\alpha-1} \\
&\le 1.45\left(\frac{\beta}{\beta-2}\right)^{\beta-1}\left(1+\frac{1}{\alpha-1}\right)^{\alpha-1} \\
&\le 1.45 * \left(1+\frac{2}{\beta-2}\right)^{\beta-2}\left(1+\frac{2}{\beta-2}\right) * e \\
&\le 1.45 * e^2 * 3 * e < 88. \qquad \square
\end{aligned}$$

Since $\alpha = \frac{g}{h}$, $g > h$ are both positive integers, and we easily have the following corollary.

Corollary 3. *$\alpha \ge 88/87$ and $\beta - \alpha < 88$.*

We are ready to prove the main theorem of this section.

Proof of Theorem 5. We claim that $\beta < 178$. If $\alpha < 89$, then $\beta < 178$. If $\alpha \ge 89$, then $\beta - \alpha \le 1 + 1/88$, but $n - g = (\beta - \alpha)h$ is an integer, and $h \le 87$, so $\beta - \alpha \le 1$. This means that $n - g \le h$, and (1) cannot hold.

We verify that there is no solution by exhaustively searching for the solutions in the finite range that $h < 88, n < 178 * 88 = 15664$ and $h < g < n$ in a computer. $\square$

Denote $\frac{n}{g-k}$ by γ and $\frac{g}{g-k}$ by δ. To prove the second part of the lemma, it suffices to see that $\binom{n}{g} = \binom{\gamma(g-k)}{\delta(g-k)} \le c_2^{g-k}$ for some constant c_2 depending only on γ and δ.

Similarly we can show that for any constant c, the inequalities

$$\binom{n}{g} \ge n^{h-c}, \tag{3}$$

$$g > \sqrt{n(g-h)} \tag{4}$$

have only a finite number of positive integral solutions.

Acknowledgments. We thank Professor Chaohua Jia for helpful discussion on the proof of Theorem 3.

REFERENCES

[1] M. Agrawal, N. Kayal, and N. Saxena, *PRIMES is in P*, Ann. of Math. (2), 160 (2004), pp. 781–793.

[2] E. Berlekamp and L. Welch, *Error Correction of Algebraic Block Codes*, U.S. Patent number 4633470, 1986.

[3] D. Bleichenbacher and P. Q. Nguyen, *Noisy polynomial interpolation and noisy Chinese remaindering*, in Advances in Cryptology (EUROCRYPT), Lecture Notes in Comput, Sci. 1807, Springer, Berlin, 2000, pp. 53–69.

[4] Qi Cheng, *On the bounded sum-of-digits discrete logarithm problem in finite fields*, SIAM J. Comput., 34 (2005), pp. 1432–1442.

[5] F. R. K. Chung, *Diameters and eigenvalues*, J. Amer. Math. Soc., 2 (1989), pp. 187–196.

[6] P. Elias, *List decoding for noisy channels*, in IRE WESCON Convention Record, Institute of Radio Engineers, New York, 1957, pp. 94–104.

[7] O. Goldreich, R. Rubinfeld, and M. Sudan, *Learning polynomials with queries: The highly noisy case*, SIAM J. Discrete Math., 13 (2000), pp. 535–570.

[8] V. Guruswami, *Limits to list decodability of linear codes*, in Proceedings of the 34th Annual ACM Symposium on Theory of Computing, ACM, New York, 2002, pp. 802–811.

[9] V. Guruswami, J. Hastad, M. Sudan, and D. Zuckerman, *Combinatorial bounds for list decoding*, IEEE Trans. Inform. Theory, 48 (2002), pp. 1021–1034.

[10] V. Guruswami, *List Decoding of Error-Correcting Codes*, Ph.D. thesis, MIT, Cambridge, MA, 2001.

[11] V. Guruswami and M. Sudan, *Improved decoding of Reed–Solomon and algebraic-geometry codes*, IEEE Trans. Inform. Theory, 45 (1999), pp. 1757–1767.

[12] S. M. Johnson, *A new upper bound for error-correcting codes*, IEEE Trans. Inform. Theory, 8 (1962), pp. 203–207.

[13] J. Justesen and T. Hoholdt, *Bounds on list decoding of MDS codes*, IEEE Trans. Inform. Theory, 47 (2001), pp. 1604–1609.

[14] N. M. Katz, *Factoring polynomials in finite fields: An application of Lang–Weil to a problem in graph theory*, Math. Ann., 286 (1990), pp. 625–637.

[15] A. Kiayias and M. Yung, *Cryptographic hardness based on the decoding of Reed–Solomon codes*, in Automata, Languages, and Programming, Lecture Notes in Comput. Sci. 2380, Springer, Berlin, 2002, pp. 232–243.

[16] A. M. Odlyzko, *Discrete logarithms: The past and the future*, Des. Codes Cryptogr., 19 (2000), pp. 129–145.

[17] C. Pomerance, *Fast, rigorous factorization, and discrete logarithm algorithms*, in Discrete Algorithms and Complexity, Academic Press, Boston, MA, 1987, pp. 119–143.

[18] M. Sudan, *Decoding of Reed–Solomon codes beyond the error-correction bound*, J. Complexity, 13 (1997), pp. 180–193.

[19] M. Sudan, *Coding theory: Tutorial & survey*, in Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, CA, 2001, pp. 36–53.

[20] J. F. Voloch, *On some subgroups of the multiplicative group of finite rings*, J. Théor. Nombres Bordeaux, 16 (2004), pp. 233–239.

[21] D. Wan, *Generators and irreducible polynomials over finite fields*, Math. Comp., 66 (1997), pp. 1195–1212.

[22] S. B. Wicker and V. K. Bhargava, eds., *Reed–Solomon Codes and Their Applications*, John Wiley and Sons, New York, 1999.

SIAM J. Comput.
Vol. 37, No. 1, pp. 210–239

QUANTUM WALK ALGORITHM FOR ELEMENT DISTINCTNESS*

ANDRIS AMBAINIS[†]

Abstract. We use quantum walks to construct a new quantum algorithm for element distinctness and its generalization. For element distinctness (the problem of finding two equal items among N given items), we get an $O(N^{2/3})$ query quantum algorithm. This improves the previous $O(N^{3/4})$ quantum algorithm of Buhrman et al. [*SIAM J. Comput.*, 34 (2005), pp. 1324–1330] and matches the lower bound of Aaronson and Shi [*J. ACM,* 51 (2004), pp. 595–605]. We also give an $O(N^{k/(k+1)})$ query quantum algorithm for the generalization of element distinctness in which we have to find k equal items among N items.

Key words. quantum computing, quantum query algorithms, element distinctness

AMS subject classifications. 81P68, 68Q25, 68Q10

DOI. 10.1137/S0097539705447311

1. Introduction.

Element distinctness is the following problem: Given numbers $x_1, \ldots, x_N \in [M]$, are they all distinct?

This problem has been extensively studied in both classical and quantum computing. Classically, the best way to solve element distinctness is by sorting, which requires $\Omega(N)$ queries. In the quantum setting, Buhrman et al. [14] have constructed a quantum algorithm that uses $O(N^{3/4})$ queries. Aaronson and Shi [1] have shown that any quantum algorithm requires at least $\Omega(N^{2/3})$ quantum queries.

In this paper, we give a new quantum algorithm that solves element distinctness with $O(N^{2/3})$ queries to $x_1, \ldots, x_N$. This matches the lower bound of [1, 5].

Our algorithm uses a combination of the following ideas: quantum search on graphs [2] and quantum walks [30]. While each of those ideas has been used before, the present combination is new.

We first reduce element distinctness to searching a certain graph with vertices $S \subseteq \{1, \ldots, N\}$ as vertices. The goal of the search is to find a marked vertex. Both examining the current vertex and moving to a neighboring vertex cost one time step. (This contrasts with the usual quantum search [26], where only examining the current vertex costs one time step.)

We then search this graph by quantum random walk. We start in a uniform superposition over all vertices of a graph and perform a quantum random walk with one transition rule for unmarked vertices of the graph and another transition rule for marked vertices of the graph. The result is that the amplitude gathers in the marked vertices and, after $O(N^{2/3})$ steps, the probability of measuring the marked state is a constant.

*Received by the editors March 7, 2005; accepted for publication (in revised form) October 19, 2005; published electronically May 14, 2007.
http://www.siam.org/journals/sicomp/37-1/44731.html

[†] Department of Combinatorics and Optimization, Faculty of Mathematics, University of Waterloo, 200 University Avenue West, Waterloo, ON N2L 2T2, Canada (ambainis@math.uwaterloo.ca). Parts of this research were done at University of Latvia, University of California, Berkeley, and Institute for Advanced Study, Princeton. This author was supported by Latvia Science Council grant 01.0354 (at University of Latvia), DARPA and Air Force Laboratory, Air Force Materiel Command, USAF, under agreement F30602-01-2-0524 (at University of California, Berkeley), NSF grant DMS-0111298 (at Institute for Advanced Study), NSERC, ARDA, an IQC University Professorship, and CIAR (at University of Waterloo).

We also give several extensions of our algorithm. If we have to find whether $x_1, \ldots, x_N$ contain k numbers that are equal, i.e., $x_{i_1} = \cdots = x_{i_k}$, we get a quantum algorithm with $O(N^{k/(k+1)})$ queries for any constant[1] k.

If the quantum algorithm is restricted to storing r numbers, $r \leq N^{2/3}$, then we have an algorithm which solves element distinctness with $O(N/\sqrt{r})$ queries and which is quadratically better than the classical $O(N^2/r)$ query algorithm. Previously, such a quantum algorithm was known only for $r \leq \sqrt{N}$ [14]. For the problem of finding k equal numbers, we get an algorithm that uses $O(\frac{N^{k/2}}{r^{(k-1)/2}})$ queries and stores r numbers for $r \leq N^{(k-1)/k}$.

For the analysis of our algorithm, we develop a generalization of Grover's algorithm (Lemma 3) which might be of independent interest.

1.1. Related work. Classical element distinctness. Element distinctness has been extensively studied classically. It can be solved with $O(N)$ queries and $O(N \log N)$ time by querying all the elements and sorting them. Then, any two equal elements must be next to one another in the sorted order and can be found by going through the sorted list.

In the usual query model (where one query gives one value of x_i), it is easy to see that $\Omega(N)$ queries are also necessary. Classical lower bounds have also been shown for more general models (e.g., [25]).

The algorithm described above requires $\Omega(N)$ space to store all of $x_1, \ldots, x_N$. If we are restricted to space $S < N$, the running time increases. The straightforward algorithm needs $O(\frac{N^2}{S})$ queries. Yao [38] has shown that, for the model of comparison-based branching programs, this is essentially optimal. Namely, any space-S algorithm needs time $T = \Omega(\frac{N^{2-o(1)}}{S})$. For more general models, lower bounds on algorithms with restricted space S is an object of ongoing research [10].

Related problems in quantum computing. In the *collision problem*, we are given a 2-1 function f and have to find x, y such that $f(x) = f(y)$. As shown by Brassard, Høyer, and Tapp [17], the collision problem can be solved in $O(N^{1/3})$ quantum steps instead of $\Theta(N^{1/2})$ steps classically. $\Omega(N^{1/3})$ is also a quantum lower bound [1, 31].

If element distinctness can be solved with M queries, then the collision problem can be solved with $O(\sqrt{M})$ queries. (This connection is credited to Yao in [1].) Thus, a quantum algorithm for element distinctness implies a quantum algorithm for collision but not the other way around.

Quantum search on graphs. The idea of quantum search on graphs was proposed by Aaronson and Ambainis [2] for finding a marked item on a d-dimensional grid (a problem first considered by Benioff [11]) and other graphs with good expansion properties. Our work has a similar flavor but uses completely different methods to search the graph (i.e., quantum walk instead of "divide-and-conquer").

Quantum walks. There has been a considerable amount of research on quantum walks (surveyed in [30]) and their applications (surveyed in [6]). Applications of walks [6] mostly fall into two classes. The first class is exponentially faster hitting times [24, 21, 19, 29]. The second class is quantum walk search algorithms [36, 22, 8].

Our algorithm is most closely related to the second class. In this direction, Shenvi, Kempe, and Whaley [36] have constructed a counterpart of Grover's search [26] based on a quantum walk on the hypercube. Childs and Goldstone [22, 23] and Ambainis,

[1] The big-O constant depends on k. For nonconstant k, we can show that the number of queries is $O(k^2 N^{k/(k+1)})$. The proof of that is mostly technical and is omitted in this version.

Kempe, and Rivosh [8] have used a quantum walk to produce search algorithms on d-dimensional lattices ($d \geq 2$), which is faster than the naive application of Grover's search. This direction is quite closely related to our work. The algorithms of [36, 22, 8] and this paper solve different problems but all have a similar structure.

Recent developments. After the work described in this paper, the results and ideas from this paper have been used to construct several other quantum algorithms. Magniez, Santha, and Szegedy [32] have used our element distinctness algorithm to give an $O(n^{1.3})$ query quantum algorithm for finding triangles in a graph. Ambainis, Kempe, and Rivosh [8] have used ideas from the current paper to construct a faster algorithm for search on a 2-dimensional grid. Childs and Eisenberg [20] have given a different analysis of our algorithm.

Szegedy [37] has generalized our results on a quantum walk for element distinctness to an arbitrary graph with a large eigenvalue gap and cast them into the language of Markov chains. His main result is that, for a class of Markov chains, quantum walk algorithms are quadratically faster than the corresponding classical algorithm. An advantage of Szegedy's approach is that it can simultaneously handle any number of solutions (unlike in the present paper, which has separate algorithms for the single solution case (Algorithm 2) and multiple-solution case (Algorithm 3)).

Buhrman and Špalek [15] have used Szegedy's result to construct an $O(n^{5/3})$ quantum algorithm for verifying if a product of two $n \times n$ matrices A and B is equal to a third matrix C.

2. Preliminaries.

2.1. Quantum query algorithms. Let $[N]$ denote $\{1, \ldots, N\}$. We consider the *element distinctness* problem: Given numbers $x_1, \ldots, x_N \in [M]$, are there $i, j \in [N]$, $i \neq j$ such that $x_i = x_j$?

Element distinctness is a particular case of the *element k-distinctness* problem: Given numbers $x_1, \ldots, x_N \in [M]$, are there k distinct indices $i_1, \ldots, i_k \in [N]$ such that $x_{i_1} = x_{i_2} = \cdots = x_{i_k}$?

We call such k indices $i_1, \ldots, i_k$ a *k-collision*.

Our model is the quantum query model (for surveys on the query model, see [7, 18]). In this model, our goal is to compute a function $f(x_1, \ldots, x_N)$. For example, k-distinctness is viewed as the function $f(x_1, \ldots, x_N)$, which is 1 if there exists a k-collision consisting of $i_1, \ldots, i_k \in [N]$ and is 0 otherwise.

The input variables x_i can be accessed by queries to an oracle X, and the complexity of f is the number of queries needed to compute f. A quantum computation with T queries is just a sequence of unitary transformations

$$U_0 \rightarrow O \rightarrow U_1 \rightarrow O \rightarrow \cdots \rightarrow U_{T-1} \rightarrow O \rightarrow U_T.$$

U_j's can be arbitrary unitary transformations that do not depend on the input bits $x_1, \ldots, x_N$. O are query (oracle) transformations. To define O, we represent basis states as $|i, a, z\rangle$, where i consists of $\lceil \log N \rceil$ bits, a consists of $\lceil \log M \rceil$ quantum bits, and z consists of all other bits. Then, O maps $|i, a, z\rangle$ to $|i, (a + x_i) \bmod M, z\rangle$.

In our algorithm, we use queries in two situations. The first situation is when $a = |0\rangle$. Then, the state before the query is some superposition $\sum_{i,z} \alpha_{i,z} |i, 0, z\rangle$ and the state after the query is the same superposition with the information about x_i: $\sum_{i,z} \alpha_{i,z} |i, x_i, z\rangle$. The second situation is when the state before the query is $\sum_{i,z} \alpha_{i,z} |i, -x_i \bmod M, z\rangle$ with the information about x_i from a previous query. Then, applying the query transformation makes the state $\sum_{i,z} \alpha_{i,z} |i, 0, z\rangle$, erasing the information about x_i. This can be used to erase the information about x_i from a state

$\sum_{i,z} \alpha_{i,z}|i, x_i, z\rangle$. We first perform a unitary that maps $|x_i\rangle \to |-x_i \bmod M\rangle$, obtaining the state $\sum_{i,z} \alpha_{i,z}|i, -x_i \bmod M, z\rangle$, and then apply the query transformation.

The computation starts with a state $|0\rangle$. Then, we apply U_0, $O, \ldots, O$, U_T and measure the final state. The result of the computation is the rightmost bit of the state obtained by the measurement.

We say that the quantum computation computes f with bounded error if, for every $x = (x_1, \ldots, x_N)$, the probability that the rightmost bit of $U_T O_x U_{T-1} \ldots O_x U_0|0\rangle$ equals $f(x_1, \ldots, x_N)$ is at least $1 - \epsilon$ for some fixed $\epsilon < 1/2$.

To simplify the exposition, we occasionally describe a quantum computation as a classical algorithm with several quantum subroutines of the form $U_t O_x U_{t-1} \ldots O_x U_0|0\rangle$. Any such classical algorithm with quantum subroutines can be transformed into an equivalent sequence $U_T O_x U_{T-1} \ldots O_x U_0|0\rangle$ with the number of queries being equal to the number of queries in the classical algorithm plus the sum of numbers of queries in all quantum subroutines.

Comparison oracle. In a different version of query model, we are allowed only comparison queries. In a comparison query, we give two indices i, j to the oracle. The oracle answers whether $x_i < x_j$ or $x_i \geq x_j$. In the quantum model, we can query the comparison oracle with a superposition $\sum_{i,j,z} a_{i,j,z}|i, j, z\rangle$, where i, j are the indices being queried and z is the rest of the quantum state. The oracle then performs a unitary transformation $|i, j, z\rangle \to -|i, j, z\rangle$ for all i, j, z such that $x_i < x_j$ and $|i, j, z\rangle \to |i, j, z\rangle$ for all i, j, z such that $x_i \geq x_j$. In section 6, we show that our algorithms can be adapted to this model with a logarithmic increase in the number of queries.

2.2. *d*-wise independence.

To make our algorithms efficient in terms of running time, and to make the multiple-solution algorithm of section 5 efficient in terms of space, we use d-wise independent functions. A reader who is interested only in the query complexity of the algorithms may skip this subsection.

DEFINITION 1. *Let $\mathcal{F}$ be a family of functions $f : [N] \to \{0, 1\}$. $\mathcal{F}$ is d-wise independent if, for all d-tuples of pairwise distinct $i_1, \ldots, i_d \in [N]$ and all $c_1, \ldots, c_d \in \{0, 1\}$,*

$$Pr[f(i_1) = c_1, f(i_2) = c_2, \ldots, f(i_d) = c_d] = \frac{1}{2^d}.$$

THEOREM 1 (see [4]). *There exists a d-wise independent family $\mathcal{F} = \{f_j | j \in [R]\}$ of functions $f_j : [N] \to \{0, 1\}$ such that*

1. *$R = O(N^{\lceil d/2 \rceil})$;*
2. *$f_j(i)$ is computable in $O(d \log^2 N)$ time, given j and i.*

We will also use families of permutations with similar properties. It is not known how to construct small d-wise independent families of permutations. There are, however, constructions of approximately d-wise independent families of permutations.

DEFINITION 2. *Let $\mathcal{F}$ be a family of permutations on $f : [n] \to [n]$. $\mathcal{F}$ is ϵ-approximately d-wise independent if, for all d-tuples of pairwise distinct $i_1, \ldots, i_d \in [n]$ and pairwise distinct $j_1, \ldots, j_d \in [n]$,*

$$Pr[f(i_1) = j_1, \ldots, f(i_d) = j_d] \in \left[\frac{1 - \epsilon}{n(n-1)\cdots(n-d+1)}, \frac{1 + \epsilon}{n(n-1)\cdots(n-d+1)}\right].$$

THEOREM 2 (see [28]). *Let n be an even power of a prime number. For any $d \leq n$, $\epsilon > 0$, there exists an ϵ-approximate d-wise independent family $\mathcal{F} = \{\pi_j | j \in [R]\}$ of permutations $\pi_j : [n] \to [n]$ such that*

1. $R = O((n^{d^2}/\epsilon^d)^{3+o(1)})$;
2. $\pi_j(i)$ *is computable in* $O(d \log^2 n)$ *time, given* j *and* i.

3. Results and algorithms.

Our main results are as follows.

THEOREM 3. *Element* k*-distinctness can be solved by a quantum algorithm with* $O(N^{k/(k+1)})$ *queries. In particular, element distinctness can be solved by a quantum algorithm with* $O(N^{2/3})$ *queries.*

THEOREM 4. *Let* $r \geq k$, $r = o(N)$. *There is a quantum algorithm that solves element distinctness with* $O(\max(\frac{N}{\sqrt{r}}, r))$ *queries and* k*-distinctness with* $O(\max(\frac{N^{k/2}}{r^{(k-1)/2}}, r))$ *queries, using* $O(r(\log M + \log N))$ *qubits of memory.*

Theorem 3 follows from Theorem 4 by setting $r = \lfloor N^{2/3} \rfloor$ for element distinctness and $r = \lfloor N^{k/(k+1)} \rfloor$ for k-distinctness. (These values minimize the expressions for the number of queries in Theorem 4.)

Next, we present Algorithm 2, which solves element distinctness if we have a promise that $x_1, \ldots, x_N$ are either all distinct or there is exactly one pair i, j, $i \neq j$, $x_i = x_j$ (and k-distinctness if we have a promise that there is at most one set of k indices $i_1, \ldots, i_k$ such that $x_{i_1} = x_{i_2} = \cdots = x_{i_k}$). The proof of correctness of Algorithm 2 is given in section 4. After that, in section 5, we present Algorithm 3, which solves the general case, using Algorithm 2 as a subroutine.

3.1. Main ideas.

We start with an informal description of our main ideas. For simplicity, we restrict our attention to element distinctness and postpone the more general k-distinctness until the end of this subsection.

Let $r = N^{2/3}$. We define a graph G with $\binom{N}{r} + \binom{N}{r+1}$ vertices. The vertices v_S correspond to sets $S \subseteq [N]$ of sizes r and $r+1$. Two vertices v_S and v_T are connected by an edge if $T = S \cup \{i\}$ for some $i \in [N]$. A vertex is marked if S contains i, j, $x_i = x_j$.

Element distinctness reduces to finding a marked vertex in this graph. If we find a marked vertex v_S, then we know that $x_i = x_j$ for some $i, j \in S$; i.e., $x_1, \ldots, x_N$ are not all distinct.

The naive way to find a marked vertex would be to use Grover's quantum search algorithm [26, 16]. If an ϵ fraction of vertices are marked, then Grover's search finds a marked vertex after $O(\frac{1}{\sqrt{\epsilon}})$ vertices. Assume that there exists a single pair $i, j \in [N]$ such that $i \neq j$, $x_i = x_j$. For a random S, $|S| = N^{2/3}$, the probability of v_S being marked is

$$Pr[i \in S; j \in S] = Pr[i \in S]Pr[j \in S|i \in S] = \frac{N^{2/3}}{N}\frac{N^{2/3}-1}{N-1} = (1-o(1))\frac{1}{N^{2/3}}.$$

Thus, a quantum algorithm can find a marked vertex by examining $O(\frac{1}{\sqrt{\epsilon}}) = O(N^{1/3})$ vertices. However, to find out if a vertex is marked, the algorithm needs to query $N^{2/3}$ items x_i, $i \in S$. This makes the total query complexity $O(N^{1/3}N^{2/3}) = O(N)$, giving no speedup compared to the classical algorithm, which queries all items.

We improve on this naive algorithm by reusing the information from previous queries. Assume that we just checked if v_S is marked by querying all x_i, $i \in S$. If the next vertex v_T is such that T contains only m elements $i \notin S$, then we need only query m elements x_i, $i \in T \setminus S$, instead of $r = N^{2/3}$ elements x_i, $i \in T$.

To formalize this, we use the following model. At each moment, we are at one vertex of G (the superposition of vertices in the quantum case). In one time step, we can examine if the current vertex v_S is marked and move to an adjacent vertex v_T. Assume

Algorithm 1 (one step of quantum walk).

1. Apply the transformation mapping $|S\rangle|y\rangle$ to
$$|S\rangle\left(\left(-1+\frac{2}{N-r}\right)|y\rangle+\frac{2}{N-r}\sum_{y'\notin S, y'\neq y}|y'\rangle\right)$$
on the S and y registers of the state in $\mathcal{H}$. (This transformation is a variant of "diffusion transformation" in [26].)
2. Map the state from $\mathcal{H}$ to $\mathcal{H}'$ by adding y to S and changing x to a vector of length $k+1$ by introducing 0 in the location corresponding to y.
3. Query for x_y and insert it into the location of x corresponding to y.
4. Apply the transformation mapping $|S\rangle|y\rangle$ to
$$|S\rangle\left(\left(-1+\frac{2}{r+1}\right)|y\rangle+\frac{2}{r+1}\sum_{y'\in S, y'\neq y}|y'\rangle\right)$$
on the y register.
5. Erase the element of x corresponding to the new y by using it as the input to query for x_y.
6. Map the state back to $\mathcal{H}$ by removing the 0 component corresponding to y from x and removing y from S.

that there is an algorithm A that finds a marked vertex with M moves between vertices. Then, there is an algorithm that solves element distinctness in $M+r$ steps in the following way:

1. We use r queries to query all x_i, $i\in S$, for the starting vertex v_S.
2. We then repeat the following two operations M times:
   (a) Check if the current vertex v_S is marked. This can be done without any queries because we already know all x_i, $i\in S$.
   (b) We simulate the algorithm A until the next move, finding the vertex v_T to which it moves from v_S. We then move to v_T by querying x_i, $i\in T\setminus S$. After that, we know all $x_i, i\in T$. We then set $S=T$.

The total number of queries is at most $M+r$, consisting of r queries for the first step and one query to simulate each move of A.

In the next sections, we will show how to search this graph by a quantum walk in $O(N^{2/3})$ steps for element distinctness and in $O(N^{k/(k+1)})$ steps for k-distinctness.

3.2. The algorithm. Let $x_1,\dots,x_N\in[M]$. We consider two Hilbert spaces $\mathcal{H}$ and $\mathcal{H}'$. $\mathcal{H}$ has dimension $\binom{N}{r}M^r(N-r)$, and the basis states of $\mathcal{H}$ are $|S,x,y\rangle$ with $S\subseteq[N]$, $|S|=r$, $x\in[M]^r$, $y\in[N]\setminus S$. $\mathcal{H}'$ has dimension $\binom{N}{r+1}M^{r+1}(r+1)$. The basis states of $\mathcal{H}'$ are $|S,x,y\rangle$ with $S\subseteq[N]$, $|S|=r+1$, $x\in[M]^{r+1}$, $y\in S$. Our algorithm thus uses
$$O\left(\binom{N}{r}M^r(N-r)+\binom{N}{r+1}M^{r+1}(r+1)\right)=O(r(\log N+\log M))$$
qubits of memory.

In the states used by our algorithm, x will always be equal to $(x_{i_1},\dots,x_{i_r})$, where $i_1,\dots,i_r$ are elements of S in increasing order.

```
ALGORITHM 2 (single-solution algorithm).
1. Generate the uniform superposition $\frac{1}{\sqrt{\binom{N}{r}(N-r)}}\sum_{|S|=r,y\notin S}|S\rangle|y\rangle$.
2. Query all $x_i$ for $i \in S$. This transforms the state to
       $\frac{1}{\sqrt{\binom{N}{r}(N-r)}}\sum_{|S|=r,y\notin S}|S\rangle|y\rangle\bigotimes_{i\in S}|x_i\rangle$.
3. $t_1 = O((N/r)^{k/2})$ times repeat:
   (a) Apply the conditional phase flip (the transformation $|S\rangle|y\rangle|x\rangle \to -|S\rangle|y\rangle|x\rangle$) for $S$ such that $x_{i_1} = x_{i_2} = \cdots = x_{i_k}$ for $k$ distinct $i_1, \ldots, i_k \in S$.
   (b) Perform $t_2 = O(\sqrt{r})$ steps of the quantum walk (algorithm 1).
4. Measure the final state. Check if $S$ contains a $k$-collision and answer "there is a $k$-collision" or "there is no $k$-collision" according to the result.
```

We start by defining a quantum walk on $\mathcal{H}$ and $\mathcal{H}'$ (Algorithm 1). Each step of the quantum walk starts in a superposition of states in $\mathcal{H}$. The first three steps map the state from $\mathcal{H}$ to $\mathcal{H}'$ and the last three steps map it back to $\mathcal{H}$.

If there is at most one k-collision, we apply Algorithm 2 (t_1 and t_2 are $c_1\sqrt{r}$ and $c_2(\frac{N}{r})^{k/2}$ for constants c_1 and c_2, which can be calculated from the analysis in section 4). This algorithm alternates the quantum walk with a transformation that changes the phase if the current state contains a k-collision. We give a proof of correctness for Algorithm 2 in section 4.

If there can be more than one k-collision, element k-distinctness is solved by Algorithm 3, which is a classical algorithm that randomly selects several subsets of x_i and runs Algorithm 2 on each subset. We give Algorithm 3 and its analysis in section 5.

4. Analysis of single k-collision algorithm.

4.1. Overview. The number of queries for Algorithm 2 is r for creating the initial state and $O((N/r)^{k/2}\sqrt{r}) = O(\frac{N^{k/2}}{r^{(k-1)/2}})$ for the rest of the algorithm. Thus, the overall number of queries is $O(\max(r, \frac{N^{k/2}}{r^{(k-1)/2}}))$. The correctness of Algorithm 2 follows from the next theorem.

THEOREM 5. *Let the input $x_1, \ldots, x_N$ be such that $x_{i_1} = \cdots = x_{i_k}$ for exactly one set of k distinct values $i_1, \ldots, i_k$. With a constant probability, measuring the final state of Algorithm* 2 *gives S such that $i_1, \ldots, i_k \in S$.*

Proof. The main ideas are as follows. We first show (Lemma 1) that the algorithm's state always stays in a $(2k+1)$-dimensional subspace of $\mathcal{H}$. After that (Lemma 2), we find the eigenvalues for the unitary transformation induced by one step of the quantum walk (Algorithm 1), restricted to this subspace. We then look at Algorithm 2 as a sequence of the form $(U_2U_1)^{t_1}$ with U_1 being a conditional phase flip and U_2 being a unitary transformation whose eigenvalues have certain properties (in this case, U_2 is t_2 steps of a quantum walk). We then prove a general result (Lemma 3) about such sequences, which implies that the algorithm finds the k-collision with a constant probability.

Let $|S, y\rangle$ be a shortcut for the basis state $|S\rangle \otimes_{i\in S} |x_i\rangle|y\rangle$. In our algorithm, the $|x\rangle$ register of a state $|S, x, y\rangle$ always contains the state $\otimes_{i\in S}|x_i\rangle$. Therefore, the state of the algorithm is always a linear combination of the basis states $|S, y\rangle$.

We classify the basis states $|S, y\rangle$ ($|S| = r$, $y \notin S$) into $2k+1$ types. A state $|S, y\rangle$ is of type $(j, 0)$ if $|S \cap \{i_1, \ldots, i_k\}| = j$ and $y \notin \{i_1, \ldots, i_k\}$ and of type $(j, 1)$ if $|S \cap \{i_1, \ldots, i_k\}| = j$ and $y \in \{i_1, \ldots, i_k\}$. For $j \in \{0, \ldots, k-1\}$, there are both type $(j, 0)$ and type $(j, 1)$ states. For $j = k$, there are only $(k, 0)$ type states. (The $(k, 1)$ type is impossible because, if $|S \cap \{i_1, \ldots, i_k\}| = k$, then $y \notin S$ implies $y \notin \{i_1, \ldots, i_k\}$.)

Let $|\psi_{j,l}\rangle$ be the uniform superposition of basis states $|S, y\rangle$ of type (j, l). Let $\tilde{H}$ be the $(2k+1)$-dimensional space spanned by states $|\psi_{j,l}\rangle$.

For the space $\mathcal{H}'$, its basis states $|S, y\rangle$ ($|S| = r+1$, $y \in S$) can be similarly classified into $2k+1$ types. We denote those types (j, l) with $j = |S \cap \{i_1, \ldots, i_k\}|$, $l = 1$ if $y \in \{i_1, \ldots, i_k\}$, and $l = 0$ otherwise. (Notice that, since $y \in S$ for the space $\mathcal{H}'$, we have type $(k, 1)$ but no type $(0, 1)$.) Let $|\varphi_{j,l}\rangle$ be the uniform superposition of basis states $|S, y\rangle$ of type (j, l) for space $\mathcal{H}'$. Let $\tilde{H}'$ be the $(2k+1)$-dimensional space spanned by $|\varphi_{j,l}\rangle$. Notice that the transformation $|S, y\rangle \to |S \cup \{y\}, y\rangle$ maps

$$|\psi_{i,0}\rangle \to |\varphi_{i,0}\rangle,\ |\psi_{i,1}\rangle \to |\varphi_{i+1,1}\rangle.$$

We claim the following.

LEMMA 1. *In Algorithm* 1, *steps* 1–3 *map* $\tilde{\mathcal{H}}$ *to* $\tilde{\mathcal{H}}'$ *and steps* 4–6 *map* $\tilde{\mathcal{H}}'$ *to* $\tilde{\mathcal{H}}$.

Proof. See section 4.2 for the proof. □

Thus, Algorithm 1 maps $\tilde{\mathcal{H}}$ to itself. Also, in Algorithm 2, step 3(a) maps $|\psi_{k,0}\rangle \to -|\psi_{k,0}\rangle$ and leaves $|\psi_{j,l}\rangle$ for $j < k$ unchanged (because $|\psi_{j,l}\rangle$, $j < k$ are superpositions of states $|S, y\rangle$ which are unchanged by step 3(b), and $|\psi_{k,0}\rangle$ is a superposition of states $|S, y\rangle$ which are mapped to $-|S, y\rangle$ by step 3(b)). Thus, every step of Algorithm 2 maps $\tilde{\mathcal{H}}$ to itself. Also, the starting state of Algorithm 2 can be expressed as a combination of $|\psi_{j,l}\rangle$. Therefore, it suffices to analyze Algorithms 1 and 2 on subspace $\tilde{\mathcal{H}}$.

In this subspace, we will be interested in two particular states. Let $|\psi_{start}\rangle$ be the uniform superposition of all $|S, y\rangle$, $|S| = r$, $y \notin S$. Let $|\psi_{good}\rangle = |\psi_{k,0}\rangle$ be the uniform superposition of all $|S, y\rangle$ with $i_1, \ldots, i_k \in S$. $|\psi_{start}\rangle$ is the algorithm's starting state. $|\psi_{good}\rangle$ is the state we would like to obtain (because measuring $|\psi_{good}\rangle$ gives a random set S such that $\{i_1, \ldots, i_k\} \subseteq S$).

We start by analyzing a single step of the quantum walk.

LEMMA 2. *Let U be the unitary transformation induced on $\tilde{\mathcal{H}}$ by one step of the quantum walk* (*Algorithm* 1). *U has $2k+1$ different eigenvalues in $\tilde{\mathcal{H}}$. One of them is* 1, *with $|\psi_{start}\rangle$ being the eigenvector. The other eigenvalues are $e^{\pm\theta_1 i}, \ldots, e^{\pm\theta_k i}$ with* $\theta_j = (2\sqrt{j} + o(1))\frac{1}{\sqrt{r}}$.

Proof. See section 4.2 for the proof. □

We set $t_2 = \lceil \frac{\pi}{3\sqrt{k}}\sqrt{r} \rceil$. Since one step of the quantum walk fixes $\tilde{\mathcal{H}}$, t_2 steps fix $\tilde{\mathcal{H}}$ as well. Moreover, $|\psi_{start}\rangle$ will still be an eigenvector with eigenvalue 1. The other $2k$ eigenvalues become $e^{\pm i(\frac{2\pi\sqrt{j}}{3\sqrt{k}} + o(1))}$. Thus, every of those eigenvalues is $e^{i\theta}$, with $\theta \in [c, 2\pi - c]$, for a constant c independent of N and r.

Let step U_1 be step 3(a) of Algorithm 2 and $U_2 = U^{t_2}$ be step 3(b). Then, the entire algorithm consists of applying $(U_2U_1)^{t_1}$ to $|\psi_{start}\rangle$. We will apply the following.

LEMMA 3. *Let $\mathcal{H}$ be a finite-dimensional Hilbert space and $|\psi_1\rangle, \ldots, |\psi_m\rangle$ be an orthonormal basis for $\mathcal{H}$. Let $|\psi_{good}\rangle$, $|\psi_{start}\rangle$ be two states in $\mathcal{H}$ which are superpositions of $|\psi_1\rangle, \ldots, |\psi_m\rangle$ with real amplitudes and $\langle\psi_{good}|\psi_{start}\rangle = \alpha$. Let U_1, U_2 be unitary transformations on $\mathcal{H}$ with the following properties:*

1. *U_1 is the transformation that flips the phase on $|\psi_{good}\rangle$ ($U_1|\psi_{good}\rangle = -|\psi_{good}\rangle$) and leaves any state orthogonal to $|\psi_{good}\rangle$ unchanged.*
2. *U_2 is a transformation which is described by a real-valued $m \times m$ matrix in the basis $|\psi_1\rangle, \ldots, |\psi_m\rangle$. Moreover, $U_2|\psi_{start}\rangle = |\psi_{start}\rangle$ and, if $|\psi\rangle$ is*

an eigenvector of U_2 perpendicular to $|\psi_{start}\rangle$, then $U_2|\psi\rangle = e^{i\theta}|\psi\rangle$ for $\theta \in [\epsilon, 2\pi - \epsilon]$, $\theta \neq \pi$ (where ϵ is a constant, $\epsilon > 0$).[2]

Then, there exists $t = O(\frac{1}{\alpha})$ such that $|\langle\psi_{good}|(U_2U_1)^t|\psi_{start}\rangle| = \Omega(1)$. (The constant under $\Omega(1)$ is independent of α but can depend on ϵ.)

Proof. See section 4.3 for the proof. □

By Lemma 3, we can set $t_1 = O(\frac{1}{\alpha})$ so that the inner product of $(U_2U_1)^{t_1}|\psi_{start}\rangle$ and $|\psi_{good}\rangle$ is a constant. Since $|\psi_{good}\rangle$ is a superposition of $|S, y\rangle$ over S satisfying $\{i_1, \ldots, i_k\} \subseteq S$, measuring $(U_2U_1)^{t_1}|\psi_{start}\rangle$ gives a set S satisfying $\{i_1, \ldots, i_k\} \subseteq S$ with a constant probability.

It remains to calculate α. Let α' be the fraction of S satisfying $\{i_1, \ldots, i_k\} \subseteq S$. Since $|\psi_{start}\rangle$ is the uniform superposition of all $|S, y\rangle$ and $|\psi_{good}\rangle$ is the uniform superposition of $|S, y\rangle$ with $\{i_1, \ldots, i_k\} \subseteq S$, we have $\alpha = \sqrt{\alpha'}$ and

$$\alpha' = Pr[\{i_1, \ldots, i_k\} \subseteq S] = \frac{\binom{N-k}{r-k}}{\binom{N}{r}} = \frac{r}{N}\prod_{j=1}^{k-1}\frac{r-j}{N-j} = (1 - o(1))\frac{r^k}{N^k}.$$

Therefore, $\alpha = \Omega(\frac{r^{k/2}}{N^{k/2}})$ and $t_1 = O((\frac{N}{r})^{k/2})$. □

Lemma 3 might also be interesting by itself. It generalizes one of the analyses of Grover's algorithm [3]. Informally, the lemma says that, in a Grover-like sequence of transformations $(U_2U_1)^t$, we can significantly relax the constraints on U_2, and the algorithm will still give a similar result. It is quite likely that such situations might appear in the analysis of other algorithms.

For the quantum walk for element k-distinctness, Childs and Eisenberg [20] have improved the analysis of Lemma 3 by showing that $\langle\psi_{good}|(U_2U_1)^t|\psi_{start}\rangle$ (and, hence, the algorithm's success probability) is $1 - o(1)$. Their result, however, does not apply to arbitrary transformations U_1 and U_2 satisfying conditions of Lemma 3.

4.2. Proofs of Lemmas 1 and 2.

Proof of Lemma 1. To show that $\tilde{\mathcal{H}}$ is mapped to $\tilde{\mathcal{H}}'$, it suffices to show that each of the basis vectors $|\psi_{j,l}\rangle$ is mapped to a vector in $\tilde{\mathcal{H}}'$. Consider vectors $|\psi_{j,0}\rangle$ and $|\psi_{j,1}\rangle$ for $j \in \{0, 1, \ldots, k-1\}$. Fix S, $|S \cap \{i_1, \ldots, i_k\}| = j$. We divide $[N] \setminus S$ into two sets S_0 and S_1. Let

$$S_0 = \{y : y \in [N] \setminus S, y \notin \{i_1, \ldots, i_k\}\},$$
$$S_1 = \{y : y \in [N] \setminus S, y \in \{i_1, \ldots, i_k\}\}.$$

Since $|S \cap \{i_1, \ldots, i_k\}| = j$, S_1 contains $s_1 = k - j$ elements. Since $S_0 \cup S_1 = [N] \setminus S$ contains $N - r$ elements, S_0 contains $s_0 = N - r - k + j$ elements. Define $|\psi_{S,0}\rangle = \frac{1}{\sqrt{N-r-k+j}}\sum_{y\in S_0}|S, y\rangle$ and $|\psi_{S,1}\rangle = \frac{1}{\sqrt{k-j}}\sum_{y\in S_1}|S, y\rangle$. Then, we have

$$(1) \qquad |\psi_{j,0}\rangle = \frac{1}{\sqrt{\binom{k}{j}\binom{N-k}{r-j}}}\sum_{\substack{S:|S|=r \\ |S\cap\{i_1,\ldots,i_k\}|=j}}|\psi_{S,0}\rangle,$$

and similarly for $|\psi_{j,1}\rangle$ and $|\psi_{S,1}\rangle$.

Consider step 1 of Algorithm 1 applied to the state $|\psi_{S,0}\rangle$. Let $|\psi'_{S,0}\rangle$ be the resulting state. Since the $|S\rangle$ register is unchanged, $|\psi'_{S,0}\rangle$ is some superposition of

[2] The requirement $\theta \neq \pi$ is made to simplify the proof of the lemma. The lemma remains true if $\theta = \pi$ is allowed. At the end of section 4.3, we sketch how to modify the proof for this case.

states $|S, y\rangle$. Moreover, both the state $|\psi_{S,0}\rangle$ and the transformation applied to this state in step 1 are invariant under permutation of states $|S, y\rangle$, $y \in S_0$, or states $|S, y\rangle$, $y \in S_1$. Therefore, the resulting state must be invariant under such permutations as well. This means that every $|S, y\rangle$, $y \in S_0$, and every $|S, y\rangle$, $y \in S_1$, has the same amplitude in $|\psi'_{S,0}\rangle$. This is equivalent to $|\psi'_{S,0}\rangle = a|\psi_{S,0}\rangle + b|\psi_{S,1}\rangle$ for some a, b. Because of (1), this means that step 1 maps $|\psi_{j,0}\rangle$ to $a|\psi_{j,0}\rangle + b|\psi_{j,1}\rangle$. Steps 2 and 3 then map $|\psi_{j,0}\rangle$ to $|\varphi_{j,0}\rangle$ and $|\psi_{j,1}\rangle$ to $|\varphi_{j+1,1}\rangle$. Thus, $|\psi_{j,0}\rangle$ is mapped to a superposition of two basis states of $\tilde{\mathcal{H}}'$: $|\varphi_{j,0}\rangle$ and $|\varphi_{j+1,1}\rangle$. Similarly, $|\psi_{j,1}\rangle$ is mapped to a (different) superposition of those two states.

For $j = k$, we have only one state $|\psi_{k,0}\rangle$. A similar argument shows that this state is unchanged by step 1 and then mapped to $|\varphi_{k,0}\rangle$, which belongs to $\tilde{\mathcal{H}}'$.

Thus, steps 1–3 map $\tilde{\mathcal{H}}$ to $\tilde{\mathcal{H}}'$. The proof that steps 4–6 map $\tilde{\mathcal{H}}'$ to $\tilde{\mathcal{H}}$ is similar. □

Proof of Lemma 2. We fix a basis for $\tilde{\mathcal{H}}$ consisting of $|\psi_{j,0}\rangle$, $|\psi_{j,1}\rangle$, $j \in \{0, \ldots, k-1\}$, and $|\psi_{k,0}\rangle$ and a basis for $\tilde{\mathcal{H}}'$ consisting of $|\varphi_{0,0}\rangle$ and $|\varphi_{j,1}\rangle$, $|\varphi_{j,0}\rangle$, $j \in \{1, \ldots, k\}$. Let D_ϵ be the matrix

$$D_\epsilon = \begin{pmatrix} -1 + 2\epsilon & 2\sqrt{\epsilon - \epsilon^2} \\ 2\sqrt{\epsilon - \epsilon^2} & 1 - 2\epsilon \end{pmatrix}.$$

Claim 1. *Let U_1 be the unitary transformation mapping $\tilde{\mathcal{H}}$ to $\tilde{\mathcal{H}}'$ induced by steps* 1–3 *of the quantum walk. Then, U_1 is described by a block diagonal matrix*

$$U_1 = \begin{pmatrix} D_{\frac{k}{N-r}} & 0 & \ldots & 0 & 0 \\ 0 & D_{\frac{k-1}{N-r}} & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & D_{\frac{1}{N-r}} & 0 \\ 0 & 0 & \ldots & 0 & 1 \end{pmatrix},$$

where the columns are in the basis $|\psi_{0,0}\rangle$, $|\psi_{0,1}\rangle$, $|\psi_{1,0}\rangle$, $|\psi_{1,1}\rangle, \ldots, |\psi_{k,0}\rangle$ and the rows are in the basis $|\varphi_{0,0}\rangle$, $|\varphi_{1,1}\rangle$, $|\varphi_{1,0}\rangle$, $|\varphi_{2,1}\rangle, \ldots, |\varphi_{k,1}\rangle$, $|\varphi_{k,0}\rangle$.

Proof. Let $\mathcal{H}_j$ be the 2-dimensional subspace of $\tilde{\mathcal{H}}$ spanned by $|\psi_{j,0}\rangle$ and $|\psi_{j,1}\rangle$. Let $\mathcal{H}'_j$ be the 2-dimensional subspace of $\tilde{\mathcal{H}}'$ spanned by $|\varphi_{j,0}\rangle$ and $|\varphi_{j+1,1}\rangle$.

From the proof of Lemma 1, we know that the subspace $\mathcal{H}_j$ is mapped to the subspace $\mathcal{H}'_j$. Thus, we have a block diagonal matrix with 2×2 blocks mapping $\mathcal{H}_j$ to $\mathcal{H}'_j$ and a 1×1 identity matrix mapping $|\psi_{k,0}\rangle$ to $|\varphi_{k,0}\rangle$. It remains to show that the transformation from $\mathcal{H}_j$ to $\mathcal{H}'_j$ is $D_{\frac{k-j}{N-r}}$. Let S be such that $|S \cap \{i_1, \ldots, i_k\}| = j$. Let S_0, S_1, $|\psi_{S,0}\rangle$, $|\psi_{S,1}\rangle$ be as in the proof of Lemma 1.

Then, step 1 of algorithm 1 maps $|\psi_{S,0}\rangle$ to

$$\frac{1}{\sqrt{s_0}} \sum_{y \in S_0} \left(\left(-1 + \frac{2}{N-r} \right) |S, y\rangle + \sum_{y' \neq y, y' \notin S} \frac{2}{N-r} |S, y'\rangle \right)$$
$$= \frac{1}{\sqrt{s_0}} \left(-1 + \frac{2}{N-r} + (s_0 - 1) \frac{2}{N-r} \right) \sum_{y \in S_0} |S, y\rangle + s_0 \frac{1}{\sqrt{s_0}} \frac{2}{N-r} \sum_{y \in S_1} |S, y\rangle$$
$$= \left(-1 + \frac{2s_0}{N-r} \right) |\psi_{S,0}\rangle + \frac{2\sqrt{s_0 s_1}}{N-r} |\psi_{S,1}\rangle.$$

By a similar calculation, $|\psi_{S,1}\rangle$ is mapped to

$$\left(-1+\frac{2s_1}{N-r}\right)|\psi_{S,1}\rangle+\frac{2\sqrt{s_0s_1}}{N-r}|\psi_{S,0}\rangle=\left(1-\frac{2s_0}{N-r}\right)|\psi_{S,1}\rangle+\frac{2\sqrt{s_0s_1}}{N-r}|\psi_{S,0}\rangle.$$

Thus, step 1 produces the transformation $D_{\frac{k-j}{N-r}}$ on $|\psi_{S,0}\rangle$ and $|\psi_{S,1}\rangle$. Since $|\psi_{j,0}\rangle$ and $|\psi_{j,1}\rangle$ are uniform superpositions of $|\psi_{S,0}\rangle$ and $|\psi_{S,1}\rangle$ over all S, step 1 also produces the same transformation $D_{\frac{k-j}{N-r}}$ on $|\psi_{j,0}\rangle$ and $|\psi_{j,1}\rangle$. Steps 2 and 3 just map $|\psi_{j,0}\rangle$ to $|\varphi_{j,0}\rangle$ and $|\psi_{j,1}\rangle$ to $|\varphi_{j+1,1}\rangle$. □

Similarly, steps 4–6 give the transformation U_2 described by the block diagonal matrix

$$U_2=\begin{pmatrix}1&0&0&\dots&0\\0&D'_{\frac{1}{r+1}}&0&\dots&0\\0&0&D'_{\frac{2}{r+1}}&\dots&0\\\vdots&\vdots&\vdots&\ddots&\vdots\\0&0&0&\dots&D'_{\frac{k}{r+1}}\end{pmatrix}$$

from $\tilde{\mathcal{H}}'$ to $\tilde{\mathcal{H}}$. Here, D'_ϵ denotes the matrix

$$D'_\epsilon=\begin{pmatrix}-1+2\epsilon&2\sqrt{\epsilon-\epsilon^2}\\2\sqrt{\epsilon-\epsilon^2}&1-2\epsilon\end{pmatrix}.$$

A step of the quantum walk is $U=U_2U_1$. Let V be the diagonal matrix with odd entries on the diagonal being -1 and even entries being 1. Since $V^2=I$, we have $U=U_2V^2U_1=U'_2U'_1$ for $U'_2=U_2V$ and $U'_1=VU_1$. Let

$$E_\epsilon=\begin{pmatrix}1-2\epsilon&2\sqrt{\epsilon-\epsilon^2}\\-2\sqrt{\epsilon-\epsilon^2}&1-2\epsilon\end{pmatrix}.$$

Then, U'_1 and U'_2 are equal to U_1 and U_2, with every D_ϵ or D'_ϵ replaced by the corresponding E_ϵ. We will first diagonalize U'_1 and U'_2 separately and then argue that eigenvalues of $U'_2U'_1$ are almost the same as eigenvalues of U'_2.

Since U'_2 is block diagonal, it suffices to diagonalize each block. A 1×1 identity block has eigenvalue 1. For a matrix E_ϵ, its characteristic polynomial is $\lambda^2-(2-4\epsilon)\lambda+1=0$ and its roots are $1-2\epsilon\pm 2\sqrt{\epsilon-\epsilon^2}i$. For $\epsilon=o(1)$, this is equal to $e^{\pm(2+o(1))i\sqrt{\epsilon}}$. Thus, the eigenvalues of U'_2 are 1, and $e^{\pm(2+o(1))\frac{\sqrt{j}}{\sqrt{r+1}}i}$ for $j\in\{1,2,\dots,k\}$. Similarly, the eigenvalues of U'_1 are 1, and $e^{\pm(2+o(1))\frac{\sqrt{j}}{\sqrt{N-r}}i}$ for $j\in\{1,2,\dots,k\}$.

To complete the proof, we use the following bound on the eigenvalues of the product of two matrices which follows from Hoffman–Wielandt theorem in matrix analysis [27].

THEOREM 6. *Let A and B be unitary matrices. Assume that A has eigenvalues $1+\delta_1,\dots,1+\delta_m$, B has eigenvalues $\mu_1,\dots,\mu_m$ and AB has eigenvalues $\mu'_1,\dots,\mu'_m$. Then,*

$$|\mu_j-\mu'_j|\le\sum_{i=1}^m|\delta_i|$$

for all $j\in[m]$.

Proof. See section 4.4 for the proof. □

Let $A = U'_1$ and $B = U'_2$. Since $|e^{\epsilon i} - 1| \leq |\epsilon|$, each $|\delta_i|$ is of order $O(\frac{1}{\sqrt{N-r}})$. Therefore, their sum is of order $O(\frac{1}{\sqrt{N-r}})$ as well. Thus, for each eigenvalue of U'_2, there is a corresponding eigenvalue of $U'_2 U'_1$ that differs by at most $O(\frac{1}{\sqrt{N-r}})$. The lemma now follows from $\frac{1}{\sqrt{N-r}} = o(\frac{1}{\sqrt{r+1}})$. □

4.3. Proof of Lemma 3. We assume that $|\alpha| < c\epsilon^2$ for some sufficiently small positive constant c. Otherwise, we can just take $t = 0$ and get $|\langle\psi_{good}|(U_2U_1)^t|\psi_{start}\rangle| = |\langle\psi_{good}|\psi_{start}\rangle| = |\alpha| \geq c\epsilon^2$.

Consider the eigenvalues of U_2. Since U_2 is described by a real $m \times m$ matrix (in the basis $|\psi_1\rangle, \ldots, |\psi_m\rangle$), its characteristic polynomial has real coefficients. Therefore, the eigenvalues are $1, -1, e^{\pm i\theta_1}, \ldots, e^{\pm i\theta_l}$. From conditions of the lemma, we know that the eigenvalue of $e^{i\pi} = -1$ never occurs.

Let $|w_{j,+}\rangle$, $|w_{j,-}\rangle$ be the eigenvectors of U_2 with eigenvalues $e^{i\theta_j}$, $e^{-i\theta_j}$. Let $|w_{j,+}\rangle = \sum_{j'=1}^{l} c_{j,j'}|\psi_{j'}\rangle$. Then, we can assume that $|w_{j,-}\rangle = \sum_{j'=1}^{l} c^*_{j,j'}|\psi_{j'}\rangle$. (Since U_2 is a real matrix, taking $U_2|w_{j,+}\rangle = e^{i\theta_j}|w_{j,+}\rangle$ and replacing every number with its complex conjugate gives $U_2|w\rangle = e^{-i\theta_j}|w\rangle$ for $|w\rangle = \sum_{j=1}^{l} c^*_{j,j'}|\psi_{j'}\rangle$.)

We write $|\psi_{good}\rangle$ in a basis consisting of eigenvectors of U_2:

$$|\psi_{good}\rangle = \alpha|\psi_{start}\rangle + \sum_{j=1}^{l}(a_{j,+}|w_{j,+}\rangle + a_{j,-}|w_{j,-}\rangle). \tag{2}$$

Without loss of generality, assume that α is a positive real. (Otherwise, multiply $|\psi_{start}\rangle$ by an appropriate factor to make α a positive real.)

We can also assume that $a_{j,+} = a_{j,-} = a_j$, with a_j being a positive real number. (To see that, let $|\psi_{good}\rangle = \sum_{j'=1}^{l} b_{j'}|\psi_{j'}\rangle$. Then, $b_{j'}$ are real (by the assumptions of Lemma 3). We have $\langle w_{j,+}|\psi_{good}\rangle = a_{j,+} = \sum_{j'=1}^{l} b_{j'}c^*_{j,j'}$ and $\langle w_{j,-}|\psi_{good}\rangle = a_{j,-} = \sum_{j'=1}^{l} b_{j'}(c^*_{j,j'})^* = (\sum_{j'=1}^{l} b_{j'}c^*_{j,j'})^* = a^*_{j,+}$. Multiplying $|w_{j,+}\rangle$ by $\frac{a^*_{j,+}}{|a_{j,+}|}$ and $|w_{j,-}\rangle$ by $\frac{a_{j,+}}{|a_{j,+}|}$ makes both $a_{j,+}$ and $a_{j,-}$ equal to $\frac{a_{j,+}a^*_{j,+}}{|a_{j,+}|} = |a_{j,+}|$, which is a positive real.

Consider the vector

$$\begin{aligned}|v_\beta\rangle = \alpha\left(1 + i\cot\frac{\beta}{2}\right)|\psi_{start}\rangle &+ \sum_{j=1}^{l} a_j\left(1 + i\cot\frac{-\theta_j + \beta}{2}\right)|w_{j,+}\rangle \\ &+ \sum_{j=1}^{l} a_j\left(1 + i\cot\frac{\theta_j + \beta}{2}\right)|w_{j,-}\rangle.\end{aligned} \tag{3}$$

We will prove that, for some $\beta = \Omega(\alpha)$, $|v_\beta\rangle$ and $|v_{-\beta}\rangle$ are eigenvectors of U_2U_1, with eigenvalues $e^{\pm i\beta}$. After that, we show that the starting state $|\psi_{start}\rangle$ is close to the state $\frac{1}{\sqrt{2}}|v_\beta\rangle + \frac{1}{\sqrt{2}}|v_{-\beta}\rangle$. Therefore, repeating U_2U_1 $\frac{\pi}{2\beta}$ times transforms $|\psi_{start}\rangle$ to a state close to $\frac{i}{\sqrt{2}}|v_\beta\rangle + \frac{-i}{\sqrt{2}}|v_{-\beta}\rangle$, which is equivalent to $\frac{1}{\sqrt{2}}|v_\beta\rangle - \frac{1}{\sqrt{2}}|v_{-\beta}\rangle$. We then complete the proof by showing that this state has a constant inner product with $|\psi_{good}\rangle$.

We first state some bounds on trigonometric functions that will be used throughout the proof.

Claim 2.

1. $\frac{2x}{\pi} \leq \sin x \leq x$ *for all* $x \in [0, \frac{\pi}{2}]$;
2. $\frac{\pi}{4x} \leq \cot x \leq \frac{1}{x}$ *for all* $x \in [0, \frac{\pi}{4}]$.

We now start the proof by establishing a sufficient condition for $|v_\beta\rangle$ and $|v_{-\beta}\rangle$ to be eigenvectors. We have $|v_\beta\rangle = |\psi_{good}\rangle + i|v'_\beta\rangle$, where

$$|v'_\beta\rangle = \alpha \cot\frac{\beta}{2}|\psi_{start}\rangle + \sum_{j=1}^{l} a_j \cot\frac{-\theta_j+\beta}{2}|w_{j,+}\rangle + \sum_{j=1}^{l} a_j \cot\frac{\theta_j+\beta}{2}|w_{j,-}\rangle. \tag{4}$$

Claim 3. *If* $|v'_\beta\rangle$ *is orthogonal to* $|\psi_{good}\rangle$*, then* $|v_\beta\rangle$ *is an eigenvector of* U_2U_1 *with an eigenvalue of* $e^{i\beta}$ *and* $|v_{-\beta}\rangle$ *is an eigenvector of* U_2U_1 *with an eigenvalue of* $e^{-i\beta}$.

Proof. Since $|v'_\beta\rangle$ is orthogonal to $|\psi_{good}\rangle$, we have $U_1|v'_\beta\rangle = |v'_\beta\rangle$ and $U_1|v_\beta\rangle = -|\psi_{good}\rangle + i|v'_\beta\rangle$. Therefore,

$$U_2U_1|v_\beta\rangle = \alpha\left(-1 + i\cot\frac{\beta}{2}\right)|\psi_{start}\rangle + \sum_{j=1}^{l} a_j e^{i\theta_j}\left(-1 + i\cot\frac{-\theta_j+\beta}{2}\right)|w_{j,+}\rangle$$
$$+ \sum_{j=1}^{l} a_j e^{-i\theta_j}\left(-1 + i\cot\frac{\theta_j+\beta}{2}\right)|w_{j,-}\rangle.$$

Furthermore,

$$1 + i\cot x = \frac{\sin x + i\cos x}{\sin x} = \frac{e^{i(\frac{\pi}{2}-x)}}{\sin x},$$

$$-1 + i\cot x = \frac{-\sin x + i\cos x}{\sin x} = \frac{e^{i(\frac{\pi}{2}+x)}}{\sin x}.$$

Therefore,

$$\left(-1 + i\cot\frac{\beta}{2}\right) = e^{i\beta}\left(1 + i\cot\frac{\beta}{2}\right),$$

$$e^{i\theta_j}\left(-1 + i\cot\frac{-\theta_j+\beta}{2}\right) = \frac{e^{i(\frac{\pi}{2}+\frac{\theta_j}{2}+\frac{\beta}{2})}}{\sin\frac{-\theta_j+\beta}{2}} = e^{i\beta}\left(1 + i\cot\frac{-\theta_j+\beta}{2}\right)$$

and similarly for the coefficient of $|w_{j,-}\rangle$. This means that $U_2U_1|v_\beta\rangle = e^{i\beta}|v_\beta\rangle$.

For $|v_{-\beta}\rangle$, we write out the inner products $\langle\psi_{good}|v'_\beta\rangle$ and $\langle\psi_{good}|v'_{-\beta}\rangle$. Then, we see that $\langle\psi_{good}|v'_{-\beta}\rangle = -\langle\psi_{good}|v'_\beta\rangle$. Therefore, if $|\psi_{good}\rangle$ and $|v'_\beta\rangle$ are orthogonal, so are $|\psi_{good}\rangle$ and $|v'_{-\beta}\rangle$. By the argument above, this implies that $|v_{-\beta}\rangle$ is an eigenvector of U_2U_1 with an eigenvalue $e^{-i\beta}$. □

Next, we use this necessary condition to bound β for which $|v_\beta\rangle$ and $|v_{-\beta}\rangle$ are eigenvectors.

Claim 4. *There exists* β *such that* $|v'_\beta\rangle$ *is orthogonal to* $|\psi_{good}\rangle$ *and* $\frac{\epsilon\alpha}{\sqrt{\pi}} \leq \beta \leq 2.6\alpha$.

Proof. Let $f(\beta) = \langle\psi_{good}|v'_\beta\rangle$. We have

$$f(\beta) = \alpha^2\cot\frac{\beta}{2} + \sum_{j=1}^{l}|a_j|^2\left(\cot\frac{-\theta_j+\beta}{2} + \cot\frac{\theta_j+\beta}{2}\right).$$

We bound $f(\beta)$ from below and above for $\beta \in [0, \frac{\epsilon}{2}]$. For the first term, we have $\frac{\pi}{2\beta} \leq \cot\frac{\beta}{2} \leq \frac{2}{\beta}$ (by Claim 2). For the second term, we have

$$\cot\frac{-\theta_j+\beta}{2} + \cot\frac{\theta_j+\beta}{2} = -\frac{\sin\beta}{\sin\frac{\theta_j+\beta}{2}\sin\frac{\theta_j-\beta}{2}}. \tag{5}$$

For the numerator, we have $\frac{2\beta}{\pi} \leq \sin\beta \leq \beta$ because of Claim 2. The denominator can be bounded from below as follows:

$$\sin\frac{\theta_j+\beta}{2}\sin\frac{\theta_j-\beta}{2} \geq \sin\frac{\epsilon}{2}\sin\frac{\epsilon}{4} \geq \frac{\epsilon^2}{2\pi^2},$$

with the first inequality following from $\theta_j \geq \epsilon$ and $\beta \leq \frac{\epsilon}{2}$ and the last inequality following from Claim 2. This means

$$\alpha^2\frac{\pi}{2\beta} - \frac{(1-\alpha^2)\pi^2}{\epsilon^2}\beta \leq f(\beta) \leq \alpha^2\frac{2}{\beta} - \frac{1-\alpha^2}{\pi}\beta, \tag{6}$$

where we have used $\|\psi_{good}\|^2 = |\alpha|^2 + 2\sum_{j=1}^{l}|a_j|^2$ (by (2)) and $\|\psi_{good}\| = 1$ to replace $\sum_{j=1}^{l}|a_j|^2$ with $\frac{1-\alpha^2}{2}$.

The lower bound of (6) implies that $f(\beta) \geq 0$ for $\beta = \frac{\epsilon}{\sqrt{2\pi(1-\alpha^2)}}\alpha$. The upper bound implies that $f(\beta) \leq 0$ for $\beta = \frac{\sqrt{2\pi}}{\sqrt{1-\alpha^2}}\alpha$. Since f is continuous, it must be the case that $f(\beta) = 0$ for some $\beta \in [\frac{\epsilon}{\sqrt{2\pi(1-\alpha^2)}}\alpha, \frac{\sqrt{2\pi}}{\sqrt{1-\alpha^2}}\alpha]$. The proposition now follows from $0 \leq \alpha \leq 0.1$. □

Let $|u_1\rangle = \frac{|v_\beta\rangle}{\|v_\beta\|}$ and $|u_2\rangle = \frac{|v_{-\beta}\rangle}{\|v_{-\beta}\|}$. We show that $|\psi_{start}\rangle$ is almost a linear combination of $|u_1\rangle$ and $|u_2\rangle$. Define $|\psi_{end}\rangle = \frac{|v_{end}\rangle}{\|v_{end}\|}$, where

$$|v_{end}\rangle = \sum_{j=1}^{l} a_j\left(1 + i\cot\frac{-\theta_j}{2}\right)|w_{j,+}\rangle + \sum_{j=1}^{l} a_j\left(1 + i\cot\frac{\theta_j}{2}\right)|w_{j,-}\rangle. \tag{7}$$

Claim 5.

$$|u_1\rangle = c_{start}i|\psi_{start}\rangle + c_{end}|\psi_{end}\rangle + |u'_1\rangle,$$
$$|u_2\rangle = -c_{start}i|\psi_{start}\rangle + c_{end}|\psi_{end}\rangle + |u'_2\rangle,$$

where c_{start}, c_{end} are positive real numbers, and u'_1, u'_2 satisfy $\|u'_1\| \leq \frac{3\beta}{\epsilon}$ and $\|u'_2\| \leq \frac{3\beta}{\epsilon}$ for β from Claim 4.

Proof. By regrouping terms in (3), we have

$$|v_\beta\rangle = \alpha i\cot\frac{\beta}{2}|\psi_{start}\rangle + |v_{end}\rangle + |v''_\beta\rangle, \tag{8}$$

where

$$|v''_\beta\rangle = \alpha|\psi_{start}\rangle + \sum_{j=1}^{l} a_j i\left(\cot\frac{-\theta_j+\beta}{2} - \cot\frac{-\theta_j}{2}\right)|w_{j,+}\rangle$$
$$+\sum_{j=1}^{l} a_j i\left(\cot\frac{\theta_j+\beta}{2} - \cot\frac{\theta_j}{2}\right)|w_{j,-}\rangle.$$

We claim that $\|v''_\beta\| \leq \frac{3\beta}{\epsilon}\|v_\beta\|$. We prove this by showing that the absolute value of each coefficient in $|v''_\beta\rangle$ is at most $\frac{3\beta}{\epsilon}$ times the absolute value of the coefficient of the same eigenvector in (3). The coefficient of $|\psi_{start}\rangle$ is α in $|v''_\beta\rangle$ and $\alpha(1+i\cot\frac{\beta}{2})$ in $|v_\beta\rangle$. We have

$$\left|\alpha\left(1+i\cot\frac{\beta}{2}\right)\right| \geq \alpha\cot\frac{\beta}{2} \geq \alpha\frac{8}{\pi\beta},$$

which means that the absolute value of the coefficient of $|\psi_{start}\rangle$ in $|v''_\beta\rangle$ is at most $\frac{\pi\beta}{8}$ times the absolute value of the coefficient in $|v_\beta\rangle$. For the coefficient of $|w_{j,+}\rangle$, we have

$$\cot\frac{-\theta_j+\beta}{2} - \cot\frac{-\theta_j}{2} = \frac{\sin\frac{\beta}{2}}{\sin\frac{-\theta_j+\beta}{2}\sin\frac{-\theta_j}{2}}.$$

If $\theta_j - \beta \geq \frac{\pi}{2}$, then

$$\left|\frac{\sin\frac{\beta}{2}}{\sin\frac{-\theta_j+\beta}{2}\sin\frac{-\theta_j}{2}}\right| \leq \frac{\frac{\beta}{2}}{\sin\frac{\pi}{4}\sin\frac{\pi}{4}} = \frac{\frac{\beta}{2}}{\frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}} = \beta \leq \beta\left|1+i\cot\frac{-\theta_j+\beta}{2}\right|.$$

If $\theta_j - \beta \leq \frac{\pi}{2}$, then

$$\left|\frac{\sin\frac{\beta}{2}}{\sin\frac{-\theta_j+\beta}{2}\sin\frac{-\theta_j}{2}}\right| = \left|\frac{\sin\frac{\beta}{2}}{\cos\frac{-\theta_j+\beta}{2}\sin\frac{-\theta_j}{2}}\cot\frac{-\theta_j+\beta}{2}\right|$$

$$\leq \frac{\frac{\beta}{2}}{\frac{1}{\sqrt{2}}\frac{\theta_j}{\pi}}\cot\left|\frac{-\theta_j+\beta}{2}\right| \leq 3\frac{\beta}{\epsilon}\left|\cot\frac{-\theta_j+\beta}{2}\right|,$$

with the first inequality following from $|\cos\frac{-\theta_j+\beta}{2}| \geq |\cos\frac{\pi}{4}| = \frac{1}{\sqrt{2}}$ and $|\sin x| = \sin|x| \geq \frac{2|x|}{\pi}$ (using Claim 2). Therefore, the absolute value of the coefficient of $|w_{j,+}\rangle$ in $|v''_\beta\rangle$ is at most $\frac{3\beta}{\epsilon}$ times the absolute value of the coefficient of $|w_{j,+}\rangle$ in $|v_\beta\rangle$ (which is $|a_j(1+i\cot\frac{-\theta_j+\beta}{2})|$). Similarly, we can bound the absolute value of the coefficient of $|w_{j,-}\rangle$.

By dividing (8) by $\|v_\beta\|$, we get

$$|u_1\rangle = c_{start}i|\psi_{start}\rangle + c_{end}|\psi_{end}\rangle + |u'_1\rangle$$

for $c_{start} = \frac{\alpha\cot\frac{\beta}{2}}{\|v_\beta\|}$, $c_{end} = \frac{\|v_{end}\|}{\|v_\beta\|}$, and $|u'_1\rangle = \frac{1}{\|v_\beta\|}|v''_\beta\rangle$. Since $\|v''_\beta\| \leq \frac{3\beta}{\epsilon}\|v_\beta\|$, we have $\|u'_1\| \leq \frac{3\beta}{\epsilon}$. The proof for u_2 is similar. □

Since $|u_1\rangle$ and $|u_2\rangle$ are eigenvectors of U_2U_1 with different eigenvalues, they must be orthogonal. Therefore,

$$\langle u_1|u_2\rangle = -c_{start}^2 + c_{end}^2 + O\left(\frac{\beta}{\epsilon}\right) = 0,$$

where $O(\frac{\beta}{\epsilon})$ denotes a term that is at most const$\frac{\beta}{\epsilon}$ in absolute value for some constant const that does not depend on β and ϵ. Also,

$$\|u_1\|^2 = c_{start}^2 + c_{end}^2 + O\left(\frac{\beta}{\epsilon}\right) = 1.$$

These two equalities, together with c_{start} and c_{end} being positive reals, imply that $c_{start} = \frac{1}{\sqrt{2}} + O(\frac{\beta}{\epsilon})$ and $c_{end} = \frac{1}{\sqrt{2}} + O(\frac{\beta}{\epsilon})$. Therefore,

$$|u_1\rangle = \frac{1}{\sqrt{2}} i|\psi_{start}\rangle + \frac{1}{\sqrt{2}}|\psi_{end}\rangle + |u_1''\rangle,$$

$$ketu_2 = -\frac{1}{\sqrt{2}} i|\psi_{start}\rangle + \frac{1}{\sqrt{2}}|\psi_{end}\rangle + |u_2''\rangle,$$

with $\|u_1''\| = O(\beta/\epsilon)$ and $\|u_2''\| = O(\beta/\epsilon)$. This means that

$$|\psi_{start}\rangle = -\frac{i}{\sqrt{2}}|u_1\rangle + \frac{i}{\sqrt{2}}|u_2\rangle + |w'\rangle,$$

$$|\psi_{end}\rangle = \frac{1}{\sqrt{2}}|u_1\rangle + \frac{1}{\sqrt{2}}|u_2\rangle + |w''\rangle,$$

where w' and w'' are states with $\|w'\| = O(\beta/\epsilon)$ and $\|w''\| = O(\beta/\epsilon)$. Let $t = \lfloor \frac{\pi}{2\beta} \rfloor$. Then, $(U_2U_1)^t|u_1\rangle$ is almost $i|u_1\rangle$ (plus a term of order $O(\beta)$) and $(U_2U_1)^t|u_2\rangle$ is almost $-i|u_2\rangle$. Therefore,

$$(U_2U_1)^t|\psi_{start}\rangle = |\psi_{end}\rangle + |v'\rangle,$$

where $\|v'\| = O(\beta/\epsilon)$. This means that

$$|\langle\psi_{good}|(U_2U_1)^t|\psi_{start}\rangle| \geq |\langle\psi_{good}|\psi_{end}\rangle| - O\left(\frac{\beta}{\epsilon}\right). \tag{9}$$

Since $\beta \leq 2.6\alpha$ and $\alpha = c\epsilon^2$, we have $O(\beta/\epsilon) = O(\epsilon)$. By choosing c to be sufficiently small, we can make the $O(\beta/\epsilon)$ term less than 0.1ϵ. Then, Lemma 3 follows from the next claim.

Claim 6.

$$|\langle\psi_{good}|\psi_{end}\rangle| \geq \min\left(\frac{1-\alpha^2}{2}, \frac{1-\alpha^2}{4}\epsilon\right).$$

Proof. Since $|\psi_{end}\rangle = \frac{|v_{end}\rangle}{\|v_{end}\|}$, we have $\langle\psi_{good}|\psi_{end}\rangle = \frac{\langle\psi_{good}|v_{end}\rangle}{\|v_{end}\|}$. By definition of $|v_{end}\rangle$ (see (7)), $\langle\psi_{good}|v_{end}\rangle = 2\sum_{j=1}^{l} a_j^2$. By (2), $\|\psi_{good}\|^2 = \alpha^2 + 2\sum_{j=1}^{l} a_j^2$. Since $\|\psi_{good}\|^2 = 1$, we have $\langle\psi_{good}|v_{end}\rangle = 1 - \alpha^2$. Therefore, $\langle\psi_{good}|\psi_{end}\rangle \geq \frac{1-\alpha^2}{\|v_{end}\|}$.

We have $\|v_{end}\|^2 = 2\sum_{j=1}^{l} a_j^2(1+\cot^2\frac{\theta_j}{2})$. Since $\theta_k \in [\epsilon, 2\pi-\epsilon]$, $\|v_{end}\|^2 \leq 2\sum_{j=1}^{l} a_j^2(1+\cot^2\frac{\epsilon}{2}) \leq (1+\cot^2\frac{\epsilon}{2})$ and

$$\langle\psi_{good}|\psi_{end}\rangle \geq \frac{1-\alpha^2}{\sqrt{1+\cot^2(\frac{\epsilon}{2})}} \geq \frac{1-\alpha^2}{2\max(1, \cot\frac{\epsilon}{2})} \geq \min\left(\frac{1-\alpha^2}{2}, \frac{1-\alpha^2}{4}\epsilon\right). \qquad \square$$

If α is set to be sufficiently small, $|\langle\psi_{good}|\psi_{end}\rangle|$ is close to 0.5ϵ and, together with (9), this means that $|\langle\psi_{good}|(U_2U_1)^t|\psi_{start}\rangle|$ is of order $\Omega(\epsilon)$. $\square$

Remark. If U_2 has eigenvectors with eigenvalue -1, then (2) becomes

$$|\psi_{good}\rangle = \alpha|\psi_{start}\rangle + \sum_{j=1}^{l}(a_{j,+}|w_{j,+}\rangle + a_{j,-}|w_{j,-}\rangle) + a_{l+1}|w_{l+1}\rangle,$$

with $|w_{l+1}\rangle$ being an eigenvector with eigenvalue -1. We also add $a_{l+1}(1 - i\tan\frac{\beta}{2})|w_{l+1}\rangle$, $-a_{l+1}i\tan\frac{\beta}{2}|w_{l+1}\rangle$, and $a_{l+1}|w_{l+1}\rangle$ terms to the right-hand sides of (3), (4), and (8), respectively. Claims 3, 4, 5, and 6 remain true, but proofs of the claims require some modifications to handle the $|w_{l+1}\rangle$ term.

4.4. Derivation of Theorem 6. In this section, we derive Theorem 6 (which was used in the proof of Lemma 2) from the Hoffman–Wielandt inequality.

DEFINITION 3. *For a matrix $C = (c_{ij})$, we define its l_2-norm as $\|C\| = \sqrt{\sum_{i,j}|c_{ij}^2|}$.*

THEOREM 7 (see [27, pp. 292]). *If U is unitary, then $\|UC\| = \|C\|$ for any C.*

THEOREM 8 (see [27, Theorem 6.3.5]). *Let C and D be $m \times m$ matrices. Let $\mu_1, \dots, \mu_m$ and $\mu'_1, \dots, \mu'_m$ be eigenvalues of C and D, respectively. Then,*

$$\sum_{i=1}^{m}(\mu_i - \mu'_i)^2 \leq \|C - D\|^2.$$

To derive Theorem 6 from Theorem 8, let $C = B$ and $D = AB$. Then, $C - D = (I - A)B$. Since B is unitary, $\|C - D\| = \|I - A\|$ (Theorem 7). Let U be a unitary matrix that diagonalizes A. Then, $U(I - A)U^{-1} = I - UAU^{-1}$ and $\|I - A\| = \|I - UAU^{-1}\|$. Since UAU^{-1} is a diagonal matrix with $1 + \delta_i$ on the diagonal, $I - UAU^{-1}$ is a diagonal matrix with δ_i on the diagonal and $\|I - UAU^{-1}\|^2 = \sum_{i=1}^{m}|\delta_i|^2$. By applying Theorem 8 to I and UAU^{-1}, we get

$$\sum_{i=1}^{m}(\mu_i - \mu'_i)^2 \leq \sum_{i=1}^{m}|\delta_i|^2.$$

In particular, for every i, we have $(\mu_i - \mu'_i)^2 \leq (\sum_{i=1}^{m}|\delta_i|^2)$ and

$$|\mu_i - \mu'_i| \leq \sqrt{\sum_{i=1}^{m}|\delta_i|^2} \leq \sum_{i=1}^{m}|\delta_i|.$$

5. Analysis of the multiple *k*-collision algorithm. To solve the general case of k-distinctness, we run Algorithm 2 several times on subsets of the input $x_i, i \in [N]$.

The simplest approach is as follows. We first run Algorithm 2 on the entire input $x_i, i \in [N]$. We then choose a sequence of subsets $T_1 \subseteq [N]$, $T_2 \subseteq [N], \dots,$ with T_i being a random subset of size $|T_i| = (\frac{2k}{2k+1})^i N$, and run Algorithm 2 on $x_i, i \in T_1$, then on $x_i, i \in T_2$, and so on. It can be shown that, if the input $x_i, i \in [N]$, contains a k-collision, then with probability at least 1/2, there exists j such that $x_i, i \in T_j$, contains exactly one k-collision. This means that running Algorithm 2 on $x_i, i \in T_j$, finds the k-collision with a constant probability.

```
ALGORITHM 3 (multiple-solution algorithm).
1. Let $T_1 = [N]$. Let $j = 1$.
2. While $|T_j| > \max(r, \sqrt{N})$ repeat:
    (a) Run Algorithm 2 on $x_i$, $i \in T_j$, using memory size $r_j = \frac{r|T_j|}{N}$. Measure the final state, obtaining a set $S$. If there are $k$ equal elements $x_i$, $i \in S$, stop, answer "there is a $k$-collision."
    (b) Let $q_j$ be an even power of a prime with $|T_j| \leq q_j \leq (1+\frac{1}{2k^2})|T_j|$. Select a random permutation $\pi_j$ on $[q_j]$ from a $\frac{1}{N}$-approximately $2k \log N$-wise independent family of permutations (Theorem 2).
    (c) Let
            $T_{j+1} = \left\{ \pi_1^{-1}\pi_2^{-1} \cdots \pi_j^{-1}(i), i \in \left[ \left\lceil \frac{2k}{2k+1} q_j \right\rceil \right] \right\}.$
    (d) Let $j = j + 1$.
3. If $|T_j| \leq r$, query all $x_i$, $i \in T_j$, classically. If $k$ equal elements are found, answer "there is a $k$-collision"; otherwise, answer "there is no $k$-collision."
4. If $|T_j| \leq \sqrt{N}$, run Grover's search on the set of at most $N^{k/2}$ $k$-tuples $(i_1, \ldots, i_k)$ of pairwise distinct $i_1, \ldots, i_k \in T_j$, searching for a tuple $(i_1, \ldots, i_k)$ such that $x_{i_1} = \cdots = x_{i_k}$. If such a tuple is found, answer "there is a $k$-collision"; otherwise, answer "there is no $k$-collision."
```

The difficulty with this solution is choosing subsets T_j. If we choose a subset of size $\frac{2k}{2k+1}N$ uniformly at random, we need $\Omega(N)$ space to store the subset and $\Omega(N)$ time to generate it. Thus, the straightforward implementation of this solution is efficient in terms of query complexity but not in terms of time or space. Algorithm 3 is a more complicated implementation of the same approach that also achieves time-efficiency and space-efficiency.

We claim the following.

THEOREM 9.

(a) *Algorithm* 3 *uses* $O(r + \frac{N^{k/2}}{r^{(k-1)/2}})$ *queries.*

(b) *Let p be the success probability of Algorithm* 2 *if there is exactly one k-collision. For any $x_1, \ldots, x_N$ containing at least one k-collision, Algorithm* 3 *finds a k-collision with probability at least* $(1 - o(1))p/2$.

Proof. (a) The second to last step of Algorithm 3 use at most r queries. The last step uses $O(N^{k/4})$ queries and is performed only if $\sqrt{N} \geq r$. In this case, $\frac{N^{k/2}}{r^{(k-1)/2}} \geq \frac{N^{k/2}}{N^{(k-1)/4}} \geq N^{k/4}$. Thus, the last two steps use $O(r + \frac{N^{k/2}}{r^{(k-1)/2}})$ queries, and it suffices to show that Algorithm 3 uses $O(r + \frac{N^{k/2}}{r^{(k-1)/2}})$ queries in its second step (the while loop).

Let T_j and r_j be as in Algorithm 3. Then, we have $|T_1| = N$ and $|T_{j+1}| \leq \frac{2k}{2k+1}(1 + \frac{1}{2k^2})|T_j|$. The number of queries in the jth iteration of the while loop is of the order

$$\frac{|T_j|^{k/2}}{r_j^{(k-1)/2}} + r_j = \frac{|T_j|^{k/2}}{(|T_j|r/N)^{(k-1)/2}} + \frac{|T_j|r}{N} = \frac{N^{(k-1)/2}}{r^{(k-1)/2}}\sqrt{|T_j|} + \frac{|T_j|r}{N}.$$

The total number of queries in the while loop is of the order

$$\sum_j \left(\frac{N^{(k-1)/2}}{r^{(k-1)/2}} \sqrt{|T_j|} + \frac{|T_j| r}{N} \right)$$

$$\leq \sum_{j=0}^{\infty} \left(\left(\frac{2k}{2k+1} \frac{2k^2+1}{2k^2} \right)^{j/2} \frac{N^{k/2}}{r^{(k-1)/2}} + \left(\frac{2k}{2k+1} \frac{2k^2+1}{2k^2} \right)^{j} r \right)$$

$$= O\left(\frac{N^{k/2}}{r^{(k-1)/2}} + r \right). \tag{10}$$

(b) If $x_1, \ldots, x_N$ contain exactly one k-collision, then running Algorithm 2 on all of $x_1, \ldots, x_N$ finds the k-collision with probability at least p. If $x_1, \ldots, x_N$ contain more than one k-collision, we can have three cases as follows:

1. For some j, T_j contains more than one k-collision but T_{j+1} contains exactly one k-collision.
2. For some j, T_j contains more than one k-collision but T_{j+1} contains no k-collisions.
3. All T_j's contain more than one k-collision (until $|T_j|$ becomes smaller than $\max(r, \sqrt{N})$ and the loop is stopped).

In the first case, performing Algorithm 2 on x_j, $j \in T_{i+1}$, finds the k-collision with probability at least p. In the second case, we have no guarantees about the probability at all. In the third case, the last step of Algorithm 3 finds one k-collision with probability 1.

We will show that the probability of the second case is always less than the probability of the first case plus an asymptotically small quantity. This implies that, with probability at least $1/2 - o(1)$, either the first or third case occurs. Therefore, the probability of Algorithm 3 finding a k-collision is at least $(1/2 - o(1))p$. To complete the proof, we show the following.

Lemma 4. *Let T be a set containing a k-collision. Let $None_j$ be the event that $x_i, i \in T_j$, contains no k-collision and $Unique_j$ be the event that $x_i, i \in T_j$, contains a unique k-collision. Then,*

$$Pr[Unique_{j+1} | T_j = T] > Pr[None_{j+1} | T_j = T] - o\left(\frac{1}{N^{1/4}} \right), \tag{11}$$

where $Pr[Unique_{j+1} | T_j = T]$ and $Pr[None_{j+1} | T_j = T]$ denote the conditional probabilities of $Unique_{j+1}$ and $None_{j+1}$ if $T_j = T$.

The probability of the first case is just the sum of probabilities

$$Pr[Unique_{j+1} \wedge T_j = T] = Pr[T_j = T] Pr[Unique_{j+1} | T_j = T]$$

over all j and T such that $|T| > \max(r, \sqrt{N})$ and T contains more than one k-collision. The probability of the second case is a similar sum of probabilities

$$Pr[None_{j+1} \wedge T_j = T] = Pr[T_j = T] Pr[None_{j+1} | T_j = T].$$

Therefore, $Pr[Unique_{j+1} | T_j = T] > Pr[None_{j+1} | T_j = T] + o(\frac{1}{N^{1/4}})$ implies that the probability of the second case is less than the probability of the first case plus a term of order $\frac{1}{N^{1/4}}$ times the number of repetitions for the while loop. The number of repetitions is $O(k \log N)$ because $|T_{j+1}| \leq \frac{2k}{2k+1}(1+\frac{1}{2k^2})|T_j| \leq (1-\frac{1}{5k})|T_j|$. Therefore, the probability of the second case is less than the probability of the first case plus a term of order $o(\frac{k \log N}{N^{1/4}}) = o(1)$.

It remains to prove the lemma.

Proof of Lemma 4. We fix the permutations $\pi_1, \ldots, \pi_{j-1}$ and let π_j be chosen uniformly at random from the family of permutations given by Theorem 2.

We consider two cases. The first case is when T_j contains many k-collisions. We show that, in this case, the lemma is true because the probability of $None_{j+1}$ is small (of order $o(\frac{1}{N^{1/4}})$). The second case is when T_j contains few k-collisions. In this case, we pick one x such that there are at least k elements i, $x_i = x$. We compare the following probabilities:

- T_{j+1} contains no k-collisions.
- T_{j+1} contains exactly one k-collision consisting of i with $x_i = x$.

The first event is the same as $None_{j+1}$; the second event implies $Unique_{j+1}$. We prove the lemma by showing that the probability of the second event is at least the probability of the first event minus a small amount. This is proven by first conditioning on T_{j+1} containing no k-collisions consisting of i with $x_i \neq x$ and then comparing the probability that less than k of $i : x_i = x$ belong to T_{j+1} with the probability that exactly k of $i : x_i = x$ belong to T_{j+1}.

Case 1. T_j contains at least $\log N$ pairwise disjoint sets $S_l = \{i_{l,1}, \ldots, i_{l,k}\}$ with $x_{i_{l,1}} = \cdots = x_{i_{l,k}}$.

Let $S = S_1 \cup S_2 \cdots \cup S_{\log N}$. If event $None_{j+1}$ occurs, then at least $\log N$ of $\pi_j \pi_{j-1} \ldots \pi_1(i)$, $i \in S$, (at least one from each of sets $S_1, \ldots, S_{\log N}$) must belong to $\{\lceil \frac{2k}{2k+1} q_j \rceil + 1, \ldots, q_j\}$. By the next proposition, this probability is almost the same as the probability that at least $\log N$ of $k \log N$ random elements of $[q_j]$ belong to $\{\lceil \frac{2k}{2k+1} q_j \rceil + 1, \ldots, q_j\}$.

Claim 7. *Let $S \subseteq T_j$, $|S| \leq 2k \log N$. Let $V \subseteq [q_j]^{|S|}$. Let p be the probability that $(\pi_j \pi_{j-1} \ldots \pi_1(i))_{i \in S}$ belongs to V and let p' be the probability that a tuple consisting of $|S|$ uniformly random elements of $[q_j]$ belongs to V. Then,*

$$|p - p'| \leq \frac{|S|^2 + 1}{q_j}.$$

Proof. Let $S' = \{\pi_{j-1} \ldots \pi_1(i) | i \in S\}$. Then, p is the probability that $(\pi_j(i))_{i \in S'}$ belongs to V. Let p'' be the probability that $(v_1, \ldots, v_{|S|})$ belongs to V for $(v_1, \ldots, v_{|S|})$ picked uniformly at random from among all tuples of $|S|$ distinct elements of $[q_j]$. By Definition 2, $|p - p''| \leq \frac{1}{N}$.

It remains to bound $|p'' - p'|$. If $(v_1, \ldots, v_{|S|})$ is picked uniformly at random from among tuples of distinct elements, every tuple of $|S|$ distinct elements has a probability $\frac{1}{q_j(q_j-1)\ldots(q_j-|S|+1)}$ and the tuples of nondistinct elements have probability 0. If $(v_1, \ldots, v_{|S|})$ is uniformly at random among all tuples, every tuple has probability $\frac{1}{q_j^{|S|}}$. Therefore,

$$\frac{q_j(q_j-1)\ldots(q_j-|S|+1)}{q_j^{|S|}} p'' \leq p' \leq \frac{q_j \ldots (q_j-|S|+1)}{q_j^{|S|}} p'' + \left(1 - \frac{q_j \ldots (q_j-|S|+1)}{q_j^{|S|}}\right),$$

which implies

$$|p' - p''| \leq 1 - \frac{q_j(q_j-1)\dots(q_j-|S|+1)}{q_j^{|S|}}.$$

We have

$$1 - \frac{q_j(q_j-1)\dots(q_j-|S|+1)}{q_j^{|S|}} \leq 1 - \left(\frac{q_j-|S|}{q_j}\right)^{|S|} \leq 1 - \left(1 - \frac{|S|^2}{q_j}\right) = \frac{|S|^2}{q_j}. \qquad \square$$

The probability that, out of $k \log N$ uniformly random $i_1, \dots, i_{k \log N} \in \{1, \dots, q_j\}$, at least $\log N$ belong to $\{\lceil \frac{2k}{2k+1} q_j \rceil + 1, \dots, q_j\}$, can be bounded using Chernoff bounds [33]. Let X_l be a random variable that is 1 if $i_l \in \{\lceil \frac{2k}{2k+1} q_j \rceil + 1, \dots, q_j\}$. Let $X = X_1 + \dots + X_{k \log N}$. We need to bound $Pr[X \geq \log N]$. We have $E[X] = k \log N \cdot E[X_1] = \frac{k}{2k+1} \log N - o(1)$ and

$$Pr[X \geq \log N] < \left(\frac{e^{(k+1)/(2k+1)}}{\frac{2k+1}{k}}\right)^{\log N} = e^{-0.316\dots \log N} = o\left(\frac{1}{N^{1/4}}\right),$$

with the first inequality following from Theorem 4.4 of [33] ($Pr[X \geq (1+\delta)E[X]] < (\frac{e^\delta}{(1+\delta)^{1+\delta}})^{E[X]}$ for X that is a sum of independent identically distributed 0-1 valued random variables).

By combining this bound with Claim 7, the probability of $None_{j+1}$ is

$$o\left(\frac{1}{N^{1/4}}\right) + \frac{(k \log N)^2 + 1}{q_j} = o\left(\frac{1}{N^{1/4}}\right),$$

where we used $q_j \geq |T_j| \geq \sqrt{N}$ (otherwise, the algorithm finishes the while loop).

Case 2. T_j contains less than $\log N$ pairwise disjoint sets $S_l = \{i_{l,1}, \dots, i_{l,k}\}$ with $x_{i_{l,1}} = \dots = x_{i_{l,k}}$.

Let S be the set of all i such that x_i is a part of a k-collision among x_i, $i \in T_j$.

CLAIM 8. *$|S| < 2k \log N$.*

Proof. We first select a maximal collection of pairwise disjoint S_l. This collection contains less than $k \log N$ elements. It remains to prove that $|S - \cup_l S_l| < k \log N$.

Since the collection $\{S_l\}$ is maximal, any k-collision between x_i, $i \in T_j$, must involve at least one element from $\cup_l S_l$. Therefore, for any x, $S \setminus \cup_l S_l$ contains at most $k-1$ values i with $x_i = x$. Also, there are less than $\log N$ possible x because any k-collision must involve an element from one of the sets S_l and there are less than $\log N$ sets S_l. This means that $|S - \cup_l S_l| < (k-1) \log N$. $\square$

Let $y_1, y_2, \dots$ be an enumeration of all distinct y such that T_j contains a k-collision $i_1, \dots, i_k$ with $x_{i_1} = \dots = x_{i_k} = y$. Let $UniqueColl_l$ be the event that T_{j+1} contains exactly one k-collision $i_1, \dots, i_k$ with $x_{i_1} = \dots = x_{i_k} = y_l$ and $NoColl_l$ be the event that T_{j+1} contains no such collision. The event $None_{j+1}$ is the same as $\bigwedge_l NoColl_l$. The event $Unique_{j+1}$ is implied by $UniqueColl_1 \wedge \bigwedge_{l>1} NoColl_l$. Therefore, it suffices to show

$$Pr\left[\bigwedge_l NoColl_l\right] < Pr\left[UniqueColl_1 \wedge \bigwedge_{l>1} NoColl_l\right] + \frac{2((2k \log N)^2 + 1)}{q_j}. \tag{12}$$

The events $UniqueColl_l$ and $NoColl_l$ are equivalent to the cardinality of

$$\left\{ i : x_i = y_l, i \in T_j, \text{ and } \pi_j \dots \pi_1(i) \in \left\{ 1, \dots, \left\lceil \frac{2k}{2k+1} q_j \right\rceil \right\} \right\}$$

being exactly k and less than k, respectively.

By Claim 7, the probabilities of both $\bigwedge_l NoColl_l$ and $UniqueColl_1 \wedge \bigwedge_{l>1} NoColl_l$ change by at most $\frac{(2k \log N)^2+1}{N}$ if we replace $(\pi_j \dots \pi_1(i))_{i \in S}$ with a tuple of $|S|$ random elements of $[q_j]$. Then, the events $NoColl_l$ and $UniqueColl_l$ are independent of events $NoColl_{l'}$ and $UniqueColl_{l'}$ for $l' \neq l$. Therefore,

$$Pr\left[\bigwedge_l NoColl_l \right] = Pr[NoColl_1] \prod_{l>1} Pr[NoColl_l],$$

$$Pr\left[UniqueColl_1 \wedge \bigwedge_{l>1} NoColl_l \right] = Pr[UniqueColl_1] \prod_{l>1} Pr[NoColl_l].$$

This means that, to show (12) for the actual probability distribution $(\pi_j \dots \pi_1(i))_{i \in S}$, it suffices to prove $Pr[UniqueColl_1] \geq Pr[NoColl_1]$ for tuples consisting of $|S|$ random elements.

Let I be the set of all $i \in T_j$ such that $x_i = y_1$. Let $m = |I|$. Notice that $m \geq k$ (by definition of x and I). Let P_l be the event that exactly l of $\pi_j \dots \pi_1(i)$, $i \in I$, belong to T_{j+1}. Then, $Pr[UniqueColl_1] = Pr[P_k]$ and $Pr[NoColl_1] = \sum_{l=0}^{k-1} Pr[P_l]$. When $\pi_j \dots \pi_1(i)$, $i \in I$, are replaced by random elements of $[q_j]$, we have

$$Pr[P_l] = \binom{m}{l} \left(1 - \frac{1}{2k+1} \right)^l \left(\frac{1}{2k+1} \right)^{m-l},$$

$$\frac{Pr[P_l]}{Pr[P_{l+1}]} = \frac{\binom{m}{l}}{\binom{m}{l+1}} \cdot \frac{1}{2k+1} \cdot \frac{1}{1 - \frac{1}{2k+1}} = \frac{l+1}{m-l} \cdot \frac{1}{2k}.$$

For $l \leq k-1$, we have $\frac{l+1}{m-l} \frac{1}{2k} \leq k \frac{1}{2k} = \frac{1}{2}$. This implies $Pr[P_l] \leq \frac{1}{2^{k-l}} Pr[P_k]$ and

$$\sum_{l=0}^{k-1} Pr[P_l] \leq \left(\sum_{l=0}^{k-1} \frac{1}{2^{k-l}} \right) Pr[P_k] \leq Pr[P_k],$$

which is equivalent to $Pr[NoColl_1] \leq Pr[UniqueColl_1]$. □

6. Running time and other issues.

6.1. Comparison model. Our algorithm can be adapted to the model of comparison queries similarly to the algorithm of [14]. Instead of having the register $\otimes_{j \in S} |x_j\rangle$, we have a register $|j_1, j_2, \dots, j_r\rangle$, where $|j_l\rangle$ is the index of the lth smallest element in the set S. Given such a register and $y \in [N]$, we can add y to $|j_1, \dots, j_r\rangle$ by binary search, which takes $O(\log N^{k/(k+1)}) = O(\log N)$ queries. We can also remove a given $x \in [N]$ in $O(\log N)$ queries by reversing this process. This gives an algorithm with $O(N^{k/(k+1)} \log N)$ queries.

6.2. Running time. So far, we have shown that our algorithm solves element k-distinctness with $O(N^{k/(k+1)})$ queries. In this section, we consider the actual running time of our algorithm (when nonquery transformations are taken into account).

Overview. All that we do between queries is Grover's diffusion operator, which can be implemented in $O(\log N)$ quantum time, and some data structure operations on set S (for example, insertions and deletions).

We now show how to store S in a classical data structure which supports the necessary operations in $O(\log^4(N+M))$ time. In a sufficiently powerful quantum model, it is possible to transform these $O(\log^4(N+M))$ time classical operations into $O(\log^c(N+M))$ step quantum computation. Then, our quantum algorithm runs in $O(N^{k/(k+1)}\log^c(N+M))$ steps. We will first show this for the standard query model and then describe how the implementation should be modified for it to work in the comparison model.

Required operations. To implement Algorithm 2, we need the following operations:

1. Adding y to S and storing x_y (step 2 of Algorithm 1);
2. removing y from S and erasing x_y (step 5 of Algorithm 1);
3. checking if S contains $i_1, \ldots, i_k$, $x_{i_1} = \cdots = x_{i_k}$ (to perform the conditional phase flip in step 3(a) of Algorithm 2);
4. diffusion transforms on $|x\rangle$ register in steps 1 and 4 of Algorithm 1.

Additional requirements. Making a data structure part of a quantum algorithm creates two subtle issues. First, there is the uniqueness problem. In many classical data structures, the same set S can be stored in many equivalent ways, depending on the order in which elements were added and removed. In the quantum case, this would mean that the basis state $|S\rangle$ is replaced by many states $|S^1\rangle$, $|S^2\rangle, \ldots,$ which in addition to S store some information about the previous sets. This can have a very bad result. In the original quantum algorithm, we might have $\alpha|S\rangle$ interfering with $-\alpha|S\rangle$, resulting in 0 amplitude for $|S\rangle$. If $\alpha|S\rangle - \alpha|S\rangle$ becomes $\alpha|S^1\rangle - \alpha|S^2\rangle$, there is no interference between $|S^1\rangle$ and $|S^2\rangle$ and the result of the algorithm will be different.

To avoid this problem, we need a data structure in which the same set $S \subseteq [N]$ is always stored in the same way, independent of how S was created.

Second, if we use a classical subroutine, it must terminate in a fixed time t. Only then can we can replace it with an $O(poly(t))$ time quantum algorithm. The subroutines that take time t on average (but might take longer time) are not acceptable.

Model. To implement our algorithm, we use a standard quantum circuit model, augmented with gates for random access to a quantum memory. A random access gate takes three inputs $|i\rangle$, $|b\rangle$, and $|z\rangle$, with b being a single qubit, z being an m-qubit register, and $i \in [m]$. It then implements the mapping

$$|i, b, z\rangle \to |i, z_i, z_1 \ldots z_{i-1} b z_{i+1} \ldots z_m\rangle.$$

Random access gates are not commonly used in quantum algorithms but are necessary in our case because, otherwise, simple data structure operations (for example, removing y from S), which require $O(\log N)$ time classically, would require $\Omega(r)$ time quantumly.

In addition to random access gates, we allow the standard one and two qubit gates [9].

Data structure: Overview. Our data structure is a combination of a hash table and a skip list. We use the hash table to store pairs (i, x_i) in the memory and

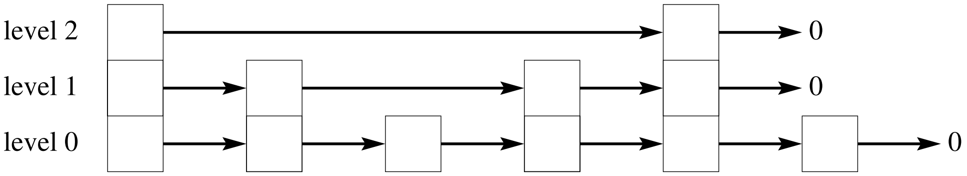


Fig. 1. *A skip list with three levels.*

to access them when we need to find x_i for a given i. We use the skip list to keep the items sorted in the order of increasing x_i so that, when a new element i is added to S, we can quickly check if x_i is equal to any of x_j, $j \in S$.

We also maintain a variable v counting the number of different $x \in [M]$ such that the set S contains $i_1, \ldots, i_k$ with $x_{i_1} = \cdots = x_{i_k} = x$.

Data structure: Hash table. Our hash table consists of r buckets, each of which contains memory for $\lceil \log N \rceil$ entries. Each entry uses $O(\log^2 N + \log M)$ qubits. The total memory is, thus, $O(r \log^3(N + M))$, slightly more than in the case when we were concerned only about the number of queries.

We hash $\{1, \ldots, N\}$ to the r buckets using a fixed hash function $h(i) = \lfloor i \cdot r/N \rfloor + 1$. The jth bucket stores pairs (i, x_i) for $i \in S$ such that $h(i) = j$ in the order of increasing i.

In the case when there are more than $\lceil \log N \rceil$ entries with $h(i) = j$, the bucket stores only $\lceil \log N \rceil$ of them. This means that our data structure malfunctions. We will show that the probability of that happening is small.

Besides the $\lceil \log N \rceil$ entries, each bucket also contains memory for storing $\lfloor \log r \rfloor$ counters $d_1, \ldots, d_{\lfloor \log r \rfloor}$. The counter d_1 in the jth bucket counts the number of $i \in S$ such that $h(i) = j$. The counter d_l, $l > 1$, is used only if j is divisible by 2^l. Then, it counts the number of $i \in S$ such that $j - 2^l + 1 \leq h(i) \leq j$.

The entry for (i, x_i) contains (i, x_i), together with a memory for $\lceil \log N \rceil + 1$ pointers to other entries that are used to set up a skip list (described below).

Data structure: Skip list. In a skip list [35], each $i \in S$ has a randomly assigned level l_i between 0 and $l_{max} = \lceil \log N \rceil$. The skip list consists of $l_{max} + 1$ lists, from the level-0 list to the level-l_{max} list. The level-l list contains all $i \in S$ with $l_i \geq l$. Each element of the level-l level list has a level-l pointer pointing to the next element of the level-l list (or 0 if there is no next element). The skip list also uses one additional "start" entry. This entry does not store any (i, x_i) but has $l_{max} + 1$ pointers, with the level-l pointer pointing to the first element of the level-l list. An example is shown in Figure 1.

In our case, each list is in the order of increasing x_i. (If several i have the same x_i, they are ordered by i.) Instead of storing an address for a memory location, pointers store the value of the next element $i \in S$. Given i, we can find the entry for (i, x_i) by computing $h(i)$ and searching the $h(i)$th bucket.

Given x, we can search the skip list as follows:

1. Traverse the level-l_{max} list until we find the last element $i_{l_{max}}$ with $x_{i_{l_{max}}} < x$.
2. For each $l = l_{max} - 1, l_{max} - 2, \ldots, 0$, traverse the level-$l$ list, starting at i_{l+1}, until we find the last element i_l with $x_{i_l} < x$.

The result of the last stage is i_0, the last element of the level-0 list (which contains all $i \in S$) with $x_{i_0} < x$. If we are given i and x_i, a similar search can find the last element i_0 which satisfies either $x_{i_0} < x_i$ or $x_{i_0} = x_i$ and $i_0 < i$. This is the element which would precede i if i were inserted into the skip list.

It remains to specify the levels l_i. The level l_i is assigned to each $i \in [N]$ before the beginning of the computation and does not change during the computation. l_i is equal to j with probability $1/2^{j+1}$ for $j < l_{max}$ and probability $1/2^{l_{max}}$ for $j = l_{max}$.

The straightforward implementation (in which we choose the level independently for each i) has the drawback that we have to store the level for each N possible $i \in [N]$, which requires $\Omega(N)$ time to choose the levels and $\Omega(N)$ space to store them. To avoid this problem, we define the levels using l_{max} functions $h_1, h_2, \dots, h_{l_{max}} : [N] \to \{0, 1\}$. $i \in [N]$ belongs to level l (for $l < l_{max}$) if $h_1(i) = \cdots = h_l(i) = 1$ but $h_{l+1}(i) = 0$. $i \in [N]$ belongs to level l_{max} if $h_1(i) = \cdots = h_{l_{max}}(i) = 1$. Each hash function is picked uniformly at random from a d-wise independent family of hash functions (Theorem 1) for $d = \lceil 4 \log_2 N + 1 \rceil$.

In the quantum case, we augment the quantum state by an extra register holding $|h_1, \dots, h_{l_{max}}\rangle$. The register is initialized to a superposition in which every basis state $|h_1, \dots, h_{l_{max}}\rangle$ has an equal amplitude. The register is then used to perform transformations dependent on $h_1, \dots, h_{l_{max}}$ on other registers.

Operations: Insertion and deletion. To add i to S, we first query the value x_i. Then, we compute $h(i)$ and add (i, x_i) to the $h(i)$th bucket. If the bucket already contains some entries, we may move some of them so that, after inserting (i, x_i), the entries are still in the order of increasing i. We then add 1 to the counter d_1 for the $h(i)$th bucket and the counter d_l for the $(\lceil \frac{h(i)}{2^l} \rceil 2^l)$th bucket, for each $l \in \{2, \dots, \lfloor \log r \rfloor\}$. We then update the skip list as follows:

1. Run the search for the last element before i (as described earlier). The search finds the last element i_l before i on each level $l \in \{0, \dots, l_{max}\}$.
2. For each level $l \in \{0, \dots, l_i\}$, let j_l be the level-l pointer of i_l. Set the level-l pointer of i to be equal to j_l and the level-l pointer of i_l to be equal to i.

After the update is complete, we use the skip list to find the smallest j such that $x_j = x_i$ and then use level-0 pointers to count if the number of $j : x_j = x_i$ is less than k, exactly k, or more than k. If there are exactly k such j, we increase v by 1. (In this case, before adding i to S, there were $k-1$ such j and, after adding i, there are k such j. Thus, the number of x such that S contains $i_1, \dots, i_k$ with $x_{i_1} = \cdots = x_{i_k} = x$ has increased by 1.)

An element i can be deleted from S by running this procedure in reverse.

Operations: Checking for k-collisions To check for k-collisions in set S, we just check if $v > 0$.

Operations: Diffusion transform. As shown by Grover [26], the following transformation on $|1\rangle, \dots, |n\rangle$ can be implemented with $O(\log n)$ elementary gates:

$$|i\rangle \to \left(-1 + \frac{2}{n}\right)|i\rangle + \sum_{i' \in [n], i' \neq i} \frac{2}{n}|i'\rangle. \tag{13}$$

To implement our transformation in step 4 of Algorithm 1, we need to implement a 1–1 mapping f between S and $\{1, \dots, |S|\}$. Once we have such a mapping, we can carry out the transformation $|y\rangle \to |f(y)\rangle$ by $|y\rangle|0\rangle \to |y\rangle|f(y)\rangle \to |0\rangle|f(y)\rangle$, where the first step is a calculation of $f(y)$ from y and the second step is the reverse of a calculation of y from $f(y)$. Then, we perform the transformation (13) on $|1\rangle, \dots, ||S|\rangle$ and then apply the transformation $|f(y)\rangle \to |y\rangle$, mapping $\{1, \dots, |S|\}$ back to S.

The mapping f can be defined as follows. $f(y) = f_1(y) + f_2(y)$, where $f_1(y)$ is the number of items $i \in S$ that are mapped to buckets j, $j < h(y)$, and $f_2(y)$ is the number of items $y' \leq y$ that are mapped to bucket $h(y)$. It is easy to see that f is a 1–1 mapping from S to $\{1, \dots, |S|\}$. $f_2(y)$ can be computed by counting the number of items in bucket $h(y)$ in time $O(\log N)$. $f_1(y)$ can be computed as follows:

```
1. Let $i = 0$, $l = \lfloor \log r \rfloor$, $s = 0$.
2. While $l \geq 0$ repeat:
    (a) If $i + 2^l < y$, add $d_l$ from the $(i + 2^l)$th bucket to $s$; let $i = i + 2^l$.
    (b) Let $l = l - 1$.
3. Return $s$ as $f_1(y)$.
```

The transformation in step 1 of Algorithm 1 is implemented, using a similar 1–1 mapping f between $[N] \setminus S$ and $\{1, \ldots, N - |S|\}$.

Uniqueness. It is easy to see that a set S is always stored in the same way. The values $i \in S$ are always hashed to buckets by h in the same way and, in each bucket, the entries are located in the order of increasing i. The counters counting the number of entries in the buckets are uniquely determined by S. The structure of the skip list is also uniquely determined, once the functions $h_1, \ldots, h_{l_{max}}$ are fixed.

Guaranteed running time. We show that, for any S, the probability that lookup, insertion, or deletion of some element takes more than $O(\log^4(N + M))$ steps is very small. We then modify the algorithms for lookup, insertion, or deletion so that they abort after $c \log^4(N + M)$ steps and show that this has no significant effect on the entire quantum search algorithm. More precisely, let

$$|\psi_t\rangle = \sum_{S,y,h_1,\ldots,h_{l_{max}}} \alpha^t_{S,y} |\psi_{S,h_1,\ldots,h_{l_{max}}}\rangle |y\rangle |h_1, \ldots, h_{l_{max}}\rangle$$

be the state of the quantum algorithm after t steps (each step being the quantum translation of one data structure operation), using quantum translations of the perfect data structure operations (which do not fail but may take more than $c \log^4 N$ steps). Here, $|\psi_{S,h_1,\ldots,h_{l_{max}}}\rangle$ stands for the basis state corresponding to our data structure storing S and x_i, $i \in S$, using the hash functions $h_1, \ldots, h_{l_{max}}$. (Notice that the amplitude $\alpha^i_{S,y}$ is independent of $h_1, \ldots, h_{l_{max}}$, since $h_1, \ldots, h_{l_{max}}$ are all equally likely.)

We decompose $|\psi_t\rangle = |\psi_t^{good}\rangle + |\psi_t^{bad}\rangle$, with $|\psi_t^{good}\rangle$ consisting of $(S, h_1, \ldots, h_{l_{max}})$, for which the next operation successfully completes in $c \log^4(N + M)$ steps, and $|\psi_t^{bad}\rangle$ consisting of $(S, h_1, \ldots, h_{l_{max}})$, for which the next operation fails to complete in $c \log^4(N + M)$ steps. Let $|\psi'_t\rangle$ be the state of the quantum algorithm after t steps using the imperfect data structure algorithms, which may abort. The next lemma is an adaptation of a "hybrid argument" by Bennett et al. [12] to our context.

Lemma 5.

$$\|\psi_t - \psi'_t\| \leq \sum_{t'=1}^{t} 2\|\psi_{t'}^{bad}\|.$$

Proof. The proof is by induction. It suffices to show that

$$\|\psi_t - \psi'_t\| \leq \|\psi_{t-1} - \psi'_{t-1}\| + 2\|\psi_t^{bad}\|.$$

To prove that, we introduce an intermediate state $|\psi''_t\rangle$, which is obtained by applying the perfect transformations in the first $t - 1$ steps and the transformation that may fail in the last step. Then,

$$\|\psi_t - \psi'_t\| \leq \|\psi_t - \psi''_t\| + \|\psi''_t - \psi'_t\|.$$

The second term, $\|\psi''_t - \psi'_t\|$, is the same as $\|\psi_{t-1} - \psi'_{t-1}\|$ because the states $|\psi''_t\rangle$ and $|\psi'_t\rangle$ are obtained by applying the same unitary transformation (quantum translation of a data structure transformation which may fail) to states $|\psi_{t-1}\rangle$ and $|\psi'_{t-1}\rangle$, respectively. To bound the first term, $\|\psi_t - \psi''_t\|$, let U_p and U_i be the unitary transformations corresponding to perfect and imperfect version of the tth data structure operation. Then, $|\psi_t\rangle = U_p|\psi_{t-1}\rangle$ and $|\psi'_t\rangle = U_i|\psi_{t-1}\rangle$. Since U_p and U_i differ only for $(S, h_1, \dots, h_{l_{max}})$, for which the data structure operation does not finish in $c\log^4 N$, steps, we have

$$\|\psi_t - \psi'_t\| = \|U_p|\psi_{t-1}\rangle - U_i|\psi_{t-1}\rangle\| = \|U_p|\psi^{bad}_{t-1}\rangle - U_i|\psi^{bad}_{t-1}\rangle\| \leq 2\|\psi^{bad}_{t-1}\|. \qquad \square$$

LEMMA 6. *For every t, $\|\psi^{bad}_t\| = O(\frac{1}{N^{1.5}})$.*

Proof. We assume that there is exactly one k-collision $x_{i_1} = \dots = x_{i_k}$. (If there are no k-collisions, the checking step at the end of Algorithm 2 ensures that the answer is correct. The case with more than one k-collision reduces to the case with exactly one k-collision because of the analysis in section 5.)

By Lemma 1, every basis state $|S, x\rangle$ of the same type has equal amplitude. Also, all $h_1, \dots, h_{l_{max}}$ have equal probabilities. Therefore, it suffices to show that, for any fixed $s = |S \cap \{i_1, \dots, i_k\}|$ and $t = |\{x\} \cap \{i_1, \dots, i_k\}|$, the fraction of $|S, x, h_1, \dots, h_{l_{max}}\rangle$ for which the operation fails is at most $\frac{1}{N^3}$.

There are two parts of the update operation which can fail as follows:

1. The hash table can overflow if more than $\lceil \log N \rceil$ elements $i \in S$ have the same $h(i) = h$.
2. The update or lookup in the skip list can take more than $c\log^4 N$ steps.

For the first part, let $s = |S \cap \{i_1, \dots, i_k\}|$. If more than $\lceil \log N \rceil$ elements $i \in S$ have $h(i) = j$, then at least $\lceil \log N \rceil - s$ of them must belong to $[N] \setminus \{i_1, \dots, i_k\}$. We now show that, for a random set $S \subseteq [N] \setminus \{i_1, \dots, i_k\}$, $|S| = r - s$, the probability that more than $\lceil \log N \rceil - s$ of $i \in S$ satisfy $h(i) = j$ is small.

We introduce random variables $X_1, \dots, X_{r-s}$ with $X_l = 1$ if h maps the lth element of S to j. We need to bound $X = X_1 + \dots + X_{r-s}$. We have $\frac{N/r - s}{N-k} \leq E[X_l] \leq \frac{N/r}{N-k}$, which means that $E[X_l] = \frac{1}{r} + O(\frac{1}{N})$. (Here, we are assuming that k is a constant. s is also a constant because $s \leq k$.) Therefore, $E[X] = (r-s)E[X_l] = 1 + o(1)$.

The random variables X_l are negatively correlated: if one or more X_l's are equal to 1, then the probability that other variables $X_{l'}$ are equal to 1 decreases. Therefore (see [34]), we can apply Chernoff bounds to bound $Pr[X > \log N - s]$. By using the bound $Pr[X \geq (1+\delta)E[X]] < (\frac{e^\delta}{(1+\delta)^{1+\delta}})^{E[X]}$ [33, 34], we get

$$Pr[X > \log N - s] < \frac{e^{\log N - s - 1}}{(\log N - s)^{\log N - s}} = o\left(\frac{1}{N^4}\right).$$

For the second part, we consider the time required for insertion of a new element. (Removing an element requires the same time because it is done by running the insertion algorithm in reverse.) Adding (i, x_i) to the $(h(i))$th bucket requires comparing i to entries already in the bucket and, possibly, moving some of the entries so that they remain sorted in the order of increasing i. Since a bucket contains $O(\log N)$ entries and each entry uses $\log^2(N+M)$ bits, this can be done in $O(\log^3(N+M))$ time. Updating counters d_l requires $O(\log N)$ time for each of the $O(\log r) = O(\log N)$ counters.

To update the skip list, we first need to compute $h_1(i), \dots, h_{l_{max}}(i)$. This is the most time consuming step, requiring $O(d\log^2 N) = O(\log^3 N)$ steps for each of the $l_{max} = \lceil \log N \rceil$ functions h_l. The total time for this step is $O(\log^4 N)$. We then need to update the pointers in the skip list. We show that for any fixed S, y (and random

$h_1, \dots, h_{l_{max}}$), the probability that updating the pointers in the skip list takes more than $c \log^4 N$ steps is small.

Each time we access a pointer in the skip list, it may take $O(\log^2 N)$ steps because a pointer stores the number i of the next entry and, to find the entry (i, x_i) itself, we have to compute $h(i)$ and search the $h(i)$th bucket, which may contain $\log N$ entries, each of which uses $\log N$ bits to store i. Therefore, it suffices to show that the probability of a skip list operation accessing more than $c \log^2 N$ pointers is small.

We show that by proving that at most $d = 4 \log N + 1$ pointer accesses are needed on each of the $\log N + 1$ levels l. We first consider level 0. Let $j_1, j_2, \dots$ be the elements of S ordered so that $x_{j_1} \le x_{j_2} \le x_{j_3} \dots$ (and, if $x_{j_l} = x_{j_{l+1}}$ for some j, then $j_l < j_{l+1}$). If the algorithm requires more than d pointer accesses on level 0, it must be the case that, for some i', $j_{i'}, \dots, j_{i'+d-1}$ are all at level 0. That is equivalent to $h_1(j_{i'}) = h_1(j_{i'+1}) = \dots = h_1(j_{i'+d-1}) = 0$. Since h_1 is d-wise independent, the probability that $h_1(j_{i'}) = \dots = h_1(j_{i'+d-1}) = 0$ is $2^{-d} < N^{-4}$.

For level l ($0 < l < l_{max}$), we first fix the hash functions $h_1, \dots, h_l$. Let $j_1, j_2, \dots$ be the elements of S for which $h_1, \dots, h_l$ are all 1, ordered so that $x_{j_1} \le x_{j_2} \le x_{j_3} \dots$. By the same argument, the probability that the algorithm needs d or more pointer accesses on level l is the same as the probability that $h_{l+1}(j_{i'}) = \dots = h_{l+1}(j_{i'+d-1}) = 0$ for some i', and this probability is at most $2^{-d} < N^{-4}$. For level l_{max}, we fix hash functions $h_1, \dots, h_{l_{max}-1}$ and notice that i is on level l_{max} whenever $h_{l_{max}}(i) = 1$. The rest of the argument is as before, with $h_{l_{max}}(j_{i'}) = h_{l_{max}}(j_{i'+1}) = \dots = h_{l_{max}}(j_{i'+d-1}) = 1$ instead of $h_1(j_{i'}) = h_1(j_{i'+1}) = \dots = h_1(j_{i'+d-1}) = 0$.

Since there are $\log N + 1$ levels and r elements of S, the probability that the algorithm spends more than $k - 1$ steps on one level for some element of S is at most $O(\frac{|S| \log N}{N^4}) = O(\frac{1}{N^3})$.

Therefore, $\|\psi_t^{bad}\|^2 = O(\frac{1}{N^3})$ and $\|\psi_t^{bad}\| = O(\frac{1}{N^{1.5}})$, proving the lemma. □

By Lemmas 5 and 6, the distance between the final states of the ideal algorithm (where the data structures never fail) and the actual algorithm is of order $O(\frac{r}{N^{3/2}}) = O(\frac{1}{N^{1/2}})$. This also means that the probability distributions obtained by measuring the two states differ by at most $O(\frac{1}{N^{1/2}})$ in variational distance [13]. Therefore, the imperfectness of the data structure operations does not have a significant effect.

Implementation in the comparison model. The implementation in the comparison model is similar, except that the hash table stores only i instead of (i, x_i).

7. Open problems.

1. **Time-space trade-offs.** Our optimal $O(N^{2/3})$-query algorithm requires space to store $O(N^{2/3})$ items.

   How many queries do we need if the algorithm's memory is restricted to r items? Our algorithm needs $O(\frac{N}{\sqrt{r}})$ queries, and this is the best known. Curiously, the lower bound for deterministic algorithms in the comparison query model is $\Omega(\frac{N^2}{r})$ queries [38], which is quadratically more. This suggests that our algorithm might be optimal in this setting as well. However, the only lower bound is the $\Omega(N^{2/3})$ lower bound for algorithms with unrestricted memory [1].
2. **Optimality of k-distinctness algorithm.** While element distinctness is known to require $\Omega(N^{2/3})$ queries, it is open whether our $O(N^{k/(k+1)})$ query algorithm for k-distinctness is optimal.

   The best lower bound for k-distinctness is $\Omega(N^{2/3})$ by the following argument. We take an instance of element distinctness $x_1, \dots, x_N$ and transform it into k-distinctness by repeating every element $k-1$ times. If $x_1, \ldots, x_N$ are all distinct, there is no k equal elements. If there are i, j such that $x_i = x_j$ among the original N elements, then repeating each of them $k-1$ times creates $2k-2$ equal elements. Therefore, solving k-distinctness on $(k-1)N$ elements requires at least the same number of queries as solving distinctness on N elements (which requires $\Omega(N^{2/3})$ queries).

3. **Quantum walks on other graphs.** A quantum walk search algorithm based on similar ideas can be used for the Grover search on grids [8, 22]. What other graphs can quantum walks–based algorithms search? Is there a graph-theoretic property that determines if quantum walk algorithms work well on this graph?

   References [8] and [37] have shown that, for a class of graphs, the performance of a quantum walk depends on certain expressions consisting of a graph's eigenvalues. In particular, if a graph has a large eigenvalue gap, a quantum walk search performs well [37]. A large eigenvalue gap is, however, not necessary, as shown by quantum search algorithms for grids [8, 37].

Acknowledgments. Thanks to Scott Aaronson for showing that k-distinctness is at least as hard as distinctness (remark 2 in section 7), to Robert Beals, Greg Kuperberg, and Samuel Kutin for pointing out the "uniqueness" problem in section 6, and to Boaz Barak, Andrew Childs, Daniel Gottesman, Julia Kempe, Samuel Kutin, Frederic Magniez, Oded Regev, Mario Szegedy, Tathagat Tulsi, and the anonymous referees for comments and discussions.

REFERENCES

[1] S. Aaronson and Y. Shi, *Quantum lower bounds for the collision and the element distinctness problems*, J. ACM, 51 (2004), pp. 595–605.

[2] S. Aaronson and A. Ambainis, *Quantum search of spatial structures*, Theory Comput., 1 (2005), pp. 47–79. Preliminary version in Proceedings of the 44th IEEE Symposium on Foundations of Computer Science, 2003, pp. 200–209.

[3] D. Aharonov, *Quantum computation—a review*, in Annual Reviews of Computational Physics, vol. VI, D. Stauffer, ed., World Scientific, River Edge, NJ, 1999, pp. 259–346.

[4] N. Alon, L. Babai, and A. Itai, *A fast and simple randomized parallel algorithm for the maximal independent set problem*, J. Algorithms, 7 (1986), pp. 567–583.

[5] A. Ambainis, *Polynomial degree and lower bounds in quantum complexity: Collision and element distinctness with small range*, Theory Comput., 1 (2005), pp. 37–46.

[6] A. Ambainis, *Quantum walks and their algorithmic applications*, Int. J. Quantum Inform., 1 (2003), pp. 507–518.

[7] A. Ambainis, *Quantum query algorithms and lower bounds*, in Classical and New Paradigms of Computation and Their Complexity Hierarchies, Trends Log. Stud. Log. Libr. 23, Kluwer, Dordrecht, The Netherlands, 2004, pp. 15–32.

[8] A. Ambainis, J. Kempe, and A. Rivosh, *Coins make quantum walks faster*, in Proceedings of the 16th Annual ACM–SIAM Symposium on Discrete Algorithms, ACM, New York, SIAM, Philadelphia, 2005, pp. 1099–1108.

[9] A. Barenco, C. Bennett, R. Cleve, D. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter, *Elementary gates for quantum computation*, Phys. Rev. A, 52 (1995), pp. 3457–3467.

[10] P. Beame, M. Saks, X. Sun, and E. Vee, *Time-space trade-off lower bounds for randomized computation of decision problems*, J. ACM, 50 (2003), pp. 154–195. Preliminary version in Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science, 2000, pp. 169–179.

[11] P. Benioff, *Space Searches with a Quantum Robot*, in Quantum Computation and Information: A Millennium Volume, Contemp. Math. 305, AMS, Providence, RI, 2002, pp. 1–12.

[12] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, *Strengths and weaknesses of quantum computing*, SIAM J. Comput., 26 (1997), pp. 1510–1523.

[13] E. Bernstein and U. Vazirani, *Quantum Complexity Theory*, SIAM J. Comput., 26 (1997), pp. 1411–1473.

[14] H. Buhrman, C. Dürr, M. Heiligman, P. Høyer, F. Magniez, M. Santha, and R. de Wolf, *Quantum algorithms for element distinctness*, SIAM J. Comput., 34 (2005), pp. 1324–1330.

[15] H. Buhrman and R. Špalek, *Quantum verification of matrix products*, in Proceedings of the 17th Annual ACM–SIAM Symposium on Discrete Algorithms, ACM, New York, SIAM, Philadelphia, 2006, pp. 880–889.

[16] G. Brassard, P. Høyer, M. Mosca, and A. Tapp, *Quantum amplitude amplification and estimation*, in Quantum Computation and Information: A Millennium Volume, Contemp. Math. 305, AMS, Providence, RI, 2002, pp. 53–74.

[17] G. Brassard, P. Høyer, and A. Tapp, *Quantum cryptanalysis of hash and claw-free functions*, in Proceedings of LATIN '98, Lecture Notes in Comput. Sci. 1380, Springer, Berlin, 1998, pp. 163–169.

[18] H. Buhrman and R. de Wolf, *Complexity measures and decision tree complexity: A survey*, Theoret. Comput. Sci., 288 (2002), pp. 21–43.

[19] A. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. Spielman, *Exponential algorithmic speedup by quantum walk*, in Proceedings of the 35th Annual Symposium on Theory of Computing, ACM, New York, 2003, pp. 59–68.

[20] A. Childs and J. Eisenberg, *Quantum algorithms for subset finding*, Quantum Inform. Comput., 5 (2005), pp. 593–604.

[21] A. Childs, E. Farhi, and S. Gutmann, *An example of the difference between quantum and classical random walks*, J. Quantum Inform. Process., 1 (2002), p. 35.

[22] A. Childs and J. Goldstone, *Spatial search by quantum walk*, Phys. Rev. A, 70 (2004), article 022314.

[23] A. Childs and J. Goldstone, *Spatial search and the Dirac equation*, Phys. Rev. A, 70 (2004), article 023122.

[24] E. Farhi and S. Gutmann, *Quantum computation and decision trees*, Phys. Rev. A, 58 (1998), pp. 915–928.

[25] D. Grigoriev, M. Karpinski, F. Meyer auf der Heide, and R. Smolensky, *A lower bound for randomized algebraic decision trees*, in Proceedings of the 28th Annual Symposium on Theory of Computing, ACM, New York, 1996, pp. 612–619.

[26] L. Grover, *A fast quantum mechanical algorithm for database search*, in Proceedings of the 28th Annual Symposium on Theory of Computing, ACM, New York, 1996, pp. 212–219.

[27] R. Horn and C. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1985.

[28] T. Itoh, T. Nagatani, and J. Tarui, *Explicit construction of k-wise nearly random permutations by Feistel transform*, in Workshop on Randomness and Computation, Sendai, Japan, 2005, pp. 15–16.

[29] J. Kempe, *Quantum random walks hit exponentially faster*, in Proceedings of RANDOM '03, Lecture Notes in Comput. Sci. 2764, Springer-Verlag, New York, 2003, pp. 354–369.

[30] J. Kempe, *Quantum random walks—An introductory overview*, Contemp. Phys., 44 (2003), pp. 307–327.

[31] S. Kutin, *A quantum lower bound for the collision problem*, Theory Comput., 1 (2005), pp. 29–36.

[32] F. Magniez, M. Santha, and M. Szegedy, *Quantum algorithms for the triangle problem*, in Proceedings of the 16th Annual ACM–SIAM Symposium on Discrete Algorithms, ACM, New York, SIAM, Philadelphia, 2005, pp. 1109–1117.

[33] M. Mitzenmacher and E. Upfal, *Probability and Computing. Randomized Algorithms and Probabilistic Analysis*, Cambridge University Press, Cambridge, UK, 2005.

[34] A. Panconesi and A. Srinivasan, *Randomized distributed edge coloring via an extension of the Chernoff–Hoeffding bounds*, SIAM J. Comput., 26 (1997), pp. 350–368.

[35] W. Pugh, *Skip lists: A probabilistic alternative to balanced trees*, Comm. ACM, 33 (1990), pp. 668–676.

[36] N. Shenvi, J. Kempe, and K. Whaley, *Quantum random-walk search algorithm*, Phys. Rev. A, 67 (2003), pp. 052307.

[37] M. Szegedy, *Quantum speed-up of Markov chain based algorithms*, in Proceedings of the 45th Annual Symposium on Foundations of Computer Science, IEEE, Los Alamitos, CA, 2004, pp. 32–41. Preliminary version appeared as Tech. report quant-ph/0401053, available online at http://arxiv.org/abs/quant-ph/0401053.

[38] A. Yao, *Near-optimal time-space tradeoff for element distinctness*, in Proceedings of the 29th Annual Symposium on Foundations of Computer Science, IEEE, Los Alamitos, CA, 1988, pp. 91–97.

SIAM J. Comput.
Vol. 37, No. 1, pp. 240–251

DYNAMIC OPTIMALITY—ALMOST*

ERIK D. DEMAINE†, DION HARMON†, JOHN IACONO‡, AND MIHAI PĂTRAŞCU†

Abstract. We present an $O(\lg \lg n)$-competitive online binary search tree, improving upon the best previous (trivial) competitive ratio of $O(\lg n)$. This is the first major progress on Sleator and Tarjan's dynamic optimality conjecture of 1985 that $O(1)$-competitive binary search trees exist.

Key words. binary search trees, splay trees, competitiveness

AMS subject classifications. 68P05, G8P10

DOI. 10.1137/S0097539705447347

1. Introduction.

Binary search trees (BSTs) are one of the most fundamental data structures in computer science. Despite decades of research, the most fundamental question about BSTs remains unsolved: What is the asymptotically best BST data structure? This problem is unsolved even if we focus on the case where the BST stores a static set and does not allow insertions and deletions.

1.1. Model.

To make precise the notion of "asymptotically best BST," we now define the standard notions of BST data structures and dynamic optimality. Our definition is based on the one by Wilber [Wil89], which also matches the one used implicitly by Sleator and Tarjan [ST85].

BST data structures. We consider BST data structures supporting only searches on a static universe of keys $\{1, 2, \ldots, n\}$. We consider only successful searches, which we call *accesses*. The input to the data structure is thus a sequence X, called the *access sequence*, of keys $x_1, x_2, \ldots, x_m$ chosen from the universe.

A BST data structure is defined by an algorithm for serving a given access x_i, called the *BST access algorithm*. The BST access algorithm has a single pointer to a node in the BST. At the beginning of an access to a given key x_i, this pointer is initialized to the root of the tree, a unit-cost operation. The algorithm may then perform any sequence of the following unit-cost operations such that the node containing x_i is eventually the target of the pointer.

1. Move the pointer to its left child.
2. Move the pointer to its right child.
3. Move the pointer to its parent.
4. Perform a single rotation on the pointer and its parent.

Whenever the pointer moves to or is initialized to a node, we say that the node is *touched*. The time taken by a BST to execute a sequence X of accesses to keys $x_1, x_2, \ldots, x_m$ is the number of unit-cost operations performed, which is at least the number of nodes it touches, which in turn is at least m. There are several possible

*Received by the editors March 23, 2005; accepted for publication (in revised form) November 10, 2005; published electronically May 14, 2007. A preliminary version of this paper appeared in FOCS 2004 [DHIP04].

http://www.siam.org/journals/sicomp/37-1/44734.html

†MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar St., Cambridge, MA 02139 (edemaine@mit.edu, dion@mit.edu, mip@mit.edu). The work of the first author was supported in part by NSF grant CCF-0430849.

‡Department of Computer and Information Science, Polytechnic University, 5 MetroTech Center, Brooklyn, NY 11201 (jiacono@poly.edu). The work of this author was supported in part by NSF grant CCF-0430849.

variants of this definition that can be shown to be equivalent up to constant factors. For example, in one such variant, the pointer begins a new operation where it finished the previous operation, rather than at the root [Wil89].

An *online BST data structure* augments each node in a BST with additional data. Every unit-cost operation can change the data in the new node pointed to by the pointer. The access algorithm's choice of the next operation to perform is a function of the data and augmented data stored in the node currently pointed to. In particular, the algorithm's behavior depends only on the past. The amount of augmented information at each node should be as small as possible. For example, red-black trees use one bit [CLRS01, Chapter 13], and splay trees do not use any [ST85]. Any online BST that uses only $O(1)$ augmented words per node has a running time in the RAM model dominated by the number of unit-cost operations in the BST model.

Optimality. Given any particular access sequence X, there is some BST data structure that executes it optimally. Let $\mathrm{OPT}(X)$ denote the number of unit-cost operations made by this fastest BST data structure for X. In other words, $\mathrm{OPT}(X)$ is the fastest any *offline* BST can execute X, because the model does not restrict how a BST access algorithm chooses its next move; thus in particular it may depend on future accesses. Here we suppose that the offline BST can start from the best possible BST storing the n keys, in order to further minimize cost. This supposition does not reduce $\mathrm{OPT}(X)$ by more than an additive $O(n)$, because any BST can be rotated into any other at such a cost; thus starting with a particular BST would increase the cost by at most a constant factor assuming $m \geq c\,n$ for a sufficiently large constant c.

Standard balanced BSTs establish that $\mathrm{OPT}(X) = O(m \lg n)$. As noted above, $\mathrm{OPT}(X) \geq m$. Wilber [Wil89] proved that $\mathrm{OPT}(X) = \Theta(m \lg n)$ for some classes of sequences X.

An online BST data structure is *dynamically optimal* if it executes all sequences X in time $O(\mathrm{OPT}(X))$. It is not known whether such a data structure exists. More generally, an online BST data structure is *c-competitive* if it executes all sufficiently long sequences X in time at most $c\,\mathrm{OPT}(X)$.

The goal of this line of research is to design a dynamically optimal (i.e., $O(1)$-competitive) online BST data structure that uses $O(1)$ augmented bits per node. The result would be a single, asymptotically best BST data structure.

1.2. Previous work. Much of the previous work on the theory of BSTs centers around splay trees of Sleator and Tarjan [ST85]. Splay trees are an online BST data structure that use a simple restructuring heuristic to move the accessed node to the root. Splay trees are conjectured in [ST85] to be dynamically optimal. This conjecture remains unresolved.

Upper bounds. Several upper bounds have been proved on the performance of splay trees: the working-set bound [ST85], the static finger bound [ST85], the sequential access bound [Tar85], and the dynamic finger bound [CMSS00, Col00]. These bounds show that splay trees execute certain classes of access sequences in $o(m \lg n)$ time, but they all provide $O(m \lg n)$ upper bounds on access sequences that actually take $\Theta(m)$ time to execute on splay trees. There are no known upper bounds on any BST that are superior to these splay tree bounds. Thus, no BST is known to be better than $O(\lg n)$-competitive against the offline optimal BST data structure.

There are several related results in different models. The unified structure [Iac01, BD04] has an upper bound on its runtime that is stronger than all of the proved upper bounds on splay trees. However, this structure is not a BST data structure, it is augmented with additional pointers, and it too is no better than $O(\lg n)$-competitive against the offline optimal BST data structure.

Lower bounds. Information theory proves that any online BST data structure costs $\Omega(m \lg n)$ on an average access sequence, but this fact does not lower bound the offline execution of any particular access sequence X. There are two known lower bounds on $\mathrm{OPT}(X)$ in the BST model, both due to Wilber [Wil89]. Neither bound is simply stated; both are complex functions of X, computable by efficient algorithms. We use a variation on the first bound extensively in this paper, described in detail in section 2 and proved in the appendix.

Optimality. Several restricted optimality results have been proved for BSTs. The first result is the "optimal BST" of Knuth [Knu71]. Given an access sequence X over the universe $\{1, 2, \ldots, n\}$, let f_i be the number of accesses in X to key i. Optimal BSTs execute X in the entropy bound $O\big(\sum_{i=1}^{n} f_i \lg(1 + \frac{m}{f_i})\big)$. This bound is expected to be $O(\mathrm{OPT}(X))$ if the accesses are drawn independently at random from a fixed distribution matching the frequencies f_i. The bound is not optimal if the accesses are dependent or not random. Originally, these trees required the f values for construction, but this requirement is lifted by splay trees, which share the asymptotic runtime of the older optimal trees.

The second result is key-independent optimality [Iac05]. Suppose $Y = \langle y_1, y_2, \ldots, y_m\rangle$ is a sequence of accesses to a set S of n unordered items. Let b be a uniform random bijection from S to $\{1, 2, \ldots, n\}$. Let $X = \langle b(x_1), b(x_2), \ldots, b(x_m)\rangle$. The key-independent optimality result proves that splay trees, and any data structure with the working-set bound, execute X in time $O(E[\mathrm{OPT}(X)])$. In other words, if key values are assigned arbitrarily (but consistently) to unordered data, splay trees are dynamically optimal. This result uses the second lower bound of Wilber [Wil89].

The third result [BCK03] shows that there is an online BST data structure whose search cost is $O(\mathrm{OPT}(X))$ given *free* rotations between accesses. This data structure is heavily augmented and uses exponential time to decide which BST operations to perform next.

1.3. Our results.

In summary, while splay trees are conjectured to be $O(1)$-competitive for all access sequences, no online BST data structure is known to have a competitive factor better than the trivial $O(\lg n)$, no matter how much time or augmentation they use to decide the next BST operation to perform. In fact, no polynomial-time offline BST is known to exist either. (Offline and with exponential time, one can of course design a dynamically optimal structure by simulating all possible offline BST structures that run in time at most $2m \lg n$ to determine the best one, before executing a single BST operation.)

We present *Tango*, an online BST data structure that is $O(\lg \lg n)$-competitive against the optimal offline BST data structure on every access sequence. Tango uses $O(\lg \lg n)$ bits (less than one word) of augmentation per node, and the bookkeeping cost to determine the next BST operation is constant amortized.

1.4. Overview.

Our results are based on a slight variation of the first lower bound of Wilber [Wil89], called the *interleave lower bound.* This lower bound is computed with the aid of a static perfect binary search tree P. By simulating the execution of the access sequence X on the perfect tree P but counting only some of the unit-cost operations performed, we obtain a lower bound on the runtime of X on any BST data structure (including those that change by rotations). Specifically, the bound counts the number of "interleaves," i.e., switches from accessing the left subtree of a node in P to accessing the right subtree of that node, or vice versa. Section 2 defines the lower bound precisely, and the appendix gives a proof. Our results show that this lower bound is within an $O(\lg\lg n)$ factor of being tight against the offline optimal; we also show in section 3.5 that the lower bound is no tighter in the worst case.

Tango simulates the behavior of the lower-bound tree P using a tree of balanced BSTs (but represented as a single BST). Specifically, before any access, we can conceptually decompose P into *preferred paths* where, at each node, a path proceeds to the child subtree that was most recently accessed. Other than a startup cost of $O(n)$, the interleave lower bound is the sum, for each access, of the number of edges that connect different preferred paths. The Tango BST matches this bound up to an $O(\lg\lg n)$ factor by representing each preferred path as a balanced BST, called an *auxiliary tree.* Because the perfect tree P has height $O(\lg n)$, each auxiliary tree contains $O(\lg n)$ nodes; thus it costs $O(\lg\lg n)$ to visit each preferred path. The technical details are in showing how to maintain the auxiliary trees as the preferred paths change, conceptually using $O(1)$ split and concatenate operations per change, while maintaining the invariant that at all times the entire structure is a single BST sorted by key. Overall, the Tango BST runs any access sequence X in $O((\mathrm{OPT}(X)+n)\,(1+\lg\lg n))$ time, which is $O(\mathrm{OPT}(X)\,(1+\lg\lg n))$ provided $m=\Omega(n)$. Section 3 gives a detailed description and analysis of Tango.

The Tango BST is similar to link-cut trees of Sleator and Tarjan [ST83, Tar83]. Both data structures maintain a tree of auxiliary trees, where each auxiliary tree represents a path of a represented tree (in Tango, the lower-bound tree P). Some versions of link-cut trees also define the partition into paths in the same way as preferred paths, except that the represented tree is dynamic. The main distinction is that Tango is a BST data structure, and therefore the "tree of auxiliary trees" must be stored as a single BST sorted by key value. In contrast, the auxiliary trees in link-cut trees are sorted by depth, making it easier to splice preferred paths as necessary. We show that this difficulty is surmountable with only a little extra bookkeeping.

1.5. Further work. Since the conference version of this paper [DHIP04], Wang, Derryberry, and Sleator [WDS06] have developed a variant of Tango trees, called the *multi-splay tree*, in which auxiliary trees are splay trees. (Interestingly, this idea is also used in one version of link-cut trees [Tar83].) In this variant, they establish $O(\lg\lg n)$-competitiveness, an $O(\lg n)$ amortized time bound per operation, and an $O(\lg^2 n)$ worst-case time bound per operation. Furthermore, they generalize the lower-bound framework and data structure to handle insertions and deletions on the universe of keys.

We note that a more complicated but easier-to-analyze variant of Tango executes any access sequence X in $O\big((\mathrm{OPT}(X)+n)\,(1+\lg\frac{\lg n}{\mathrm{OPT}(X)/n})\big)$ time, which is always $O(n\lg n)$. The variant also achieves $O(\lg n)$ worst-case time per operation. Namely, we can replace the auxiliary tree data structure with a balanced BST supporting search, split, and concatenate operations in the worst-case dynamic finger bound, $O(1+\lg r)$ worst-case time, where r is 1 plus the rank difference between the accessed element and the previously accessed element. For example, one such data structure maintains the previously accessed element at the root and has subtrees hanging off the spine with size roughly exponentially increasing with distance from the root. Following the analysis in section 3.4, the cost of accessing an element in the tree of auxiliary trees then becomes $O\big(\sum_{i=1}^{k}(1+\lg r_i)\big)$, where $r_1, r_2, \ldots, r_k$ are the numbers of nodes along the k preferred paths we visit; thus $r_1+r_2+\cdots+r_k=\Theta(\lg n)$. This cost is maximized up to constant factors when $r_1=r_2=\cdots=r_k=\Theta\big(\frac{\lg n}{k}\big)$, for a cost of $O\big(k(1+\lg\frac{\lg n}{k})\big)$. Because $k = O(\lg n)$, we obtain an $O(\lg n)$ worst-case time bound per access. Summing over all accesses, the k's sum to the interleave lower bound plus an $O(n)$ startup cost, so the total cost is maximized up to constant factors when each access cost is approximately equal, for a total cost of $O\big((\mathrm{OPT}(X)+n)\,(1+\lg\frac{\lg n}{\mathrm{OPT}(X)/n})\big)$.

2. Interleave lower bound.

The *interleave bound* is a lower bound on the time taken by any BST data structure to execute an access sequence X, dependent only on X. The particular version of the bound that we use is a slight variation of the first bound of Wilber [Wil89]. (Specifically, our lower-bound tree P has a key at every node, instead of just at the leaves, we also fix P to be the complete tree for the purposes of upper bounds, and our bound differs by roughly a factor of two because we do not insist that the search algorithm brings the desired element to the root.) Our lower bound is also similar to lower bounds that follow from partial sums in the semigroup model [HF98, PD06]. Given the similarity to previous lower bounds, we just state the bound in this section, and delay the proof to the appendix.

We maintain a perfect binary tree, called the *lower-bound tree* P, on the keys $\{1, 2, \ldots, n\}$. (If n is not one less than a power of two, the tree is complete, not perfect.) This tree has a fixed structure over time.

For each node y in P, define the *left region* of y to consist of y itself plus all nodes in y's left subtree in P, and define the *right region* of y to consist of all nodes in y's right subtree in P. The left and right regions of y partition y's subtree in P and are temporally invariant. For each node y in P, we label each access x_i in the access sequence X by whether x_i is in the left or right region of y, discarding all accesses outside y's subtree in P. The *amount of interleaving through* y is the number of alternations between "left" and "right" labels in this sequence. The *interleave bound* $\mathrm{IB}(X)$ is the sum of these interleaving amounts over all nodes y in P.

The exact statement of the lower bound is as follows.

THEOREM 2.1. *$\mathrm{IB}(X)/2-n$ is a lower bound on $\mathrm{OPT}(X)$, the cost of the optimal offline BST that serves access sequence X.*

3. BST upper bound.

3.1. Overview of the Tango BST.

We now define a specific BST access algorithm, called *Tango*. Let T_i denote the state of the Tango BST after executing the first i accesses $x_1, x_2, \ldots, x_i$. We define T_i in terms of an *augmented* lower-bound tree P.

As in the interleave lower bound, P is a perfect binary tree on the same set of keys, $\{1, 2, \ldots, n\}$. We augment P to maintain, for each internal node y of P, a *preferred child* of either the left or the right, specifying whether the last access to a node within y's subtree in P was in the left or right region of y. In particular, because y is in its left region, an access to y sets the preferred child of y to the left. If no node within y's subtree in P has yet been accessed (or if y is a leaf), y has no preferred child. The state P_i of this augmented perfect binary tree after executing the first i accesses $x_1, x_2, \ldots, x_i$ is determined solely by the access sequence, independent of the Tango BST.

The following transformation converts a state P_i of P into a state T_i of the Tango BST. Start at the root of P and repeatedly proceed to the preferred child of the current node until reaching a node without a preferred child (e.g., a leaf). The nodes traversed by this process, including the root, form a *preferred path*. We compress this preferred path into an "auxiliary tree" R. (Auxiliary trees are BSTs defined in section 3.2.) Removing this preferred path from P splits P into several pieces; we recurse on each piece and hang the resulting BSTs as child subtrees of the auxiliary tree R.

The behavior of the Tango BST is now determined: At each access x_i, the state T_i of the Tango BST is given by the transformation described above applied to P_i. We have not yet defined how to efficiently obtain T_i from T_{i-1}. To address this algorithmic issue, we first describe auxiliary trees and the operations they support.

3.2. Auxiliary tree. The *auxiliary tree* data structure is an augmented BST that stores a subpath of a root-to-leaf path in P (in our case, a preferred path), but ordered by key value. With each node we also store its fixed *depth* in P. Thus, the depths of the nodes in an auxiliary tree form a subinterval of $[0, \lg(n + 1))$. We call the shallowest node the *top* of the path, and the deepest node the *bottom* of the path. We require the following operations of auxiliary trees:

1. *searching* for an element by key in an auxiliary tree;
2. *cutting* an auxiliary tree into two auxiliary trees, one storing the path of all nodes of depth at most a specified depth d, and the other storing the path of all nodes of depth greater than d;
3. *joining* two auxiliary trees that store two disjoint paths where the bottom of one path is the parent of the top of the other path.

We require that all of these operations take time $O(\lg k)$, where k is the total number of nodes in the auxiliary tree(s) involved in the operation. Note that the requirements of auxiliary trees (and indeed their solution) are similar to Sleator and Tarjan's link-cut trees [ST83, Tar83]; however, auxiliary trees have the additional property that the nodes are stored in a BST ordered by key value, not by depth in the path.

An auxiliary tree is implemented as an augmented red-black tree. In addition to storing the key value and depth, each node stores the minimum and maximum depth over the nodes in its subtree. This auxiliary data can be trivially maintained in red-black trees with a constant-factor overhead; see, e.g., [CLRS01, Chapter 14].

The additional complication is that the nodes which would normally lack a child in the red-black tree (e.g., the leaves) can nonetheless have child pointers which point to other auxiliary trees. In order to distinguish auxiliary trees within this tree-of-auxiliary-trees decomposition, we mark the root of each auxiliary tree.

Recall that red-black trees support search, split, and concatenate in $O(\lg k)$ time [CLRS01, Problem 13-2]. In particular, this allows us to search in an augmented tree in $O(\lg k)$ time. We use the following specific forms of split and concatenate phrased in terms of a tree-of-trees representation instead of a forest representation:

1. *Split* a red-black tree at a node x: Rearrange the tree so that x is at the root, the left subtree of x is a red-black tree on the nodes with keys less than x, and the right subtree of x is a red-black tree on the nodes with keys greater than x.
2. *Concatenate* two red-black trees whose roots are children of a common node x: Rearrange x's subtree to form a red-black tree on x and the nodes in its subtree.

Both operations do not descend into marked nodes, where other auxiliary trees begin, treating them as external nodes (i.e., ignoring the existence of marked subtrees but preserving the pointers to them automatically during rotations). It is easy to phrase existing split and concatenate algorithms in this framework.

Now we describe how to support cut and join using split and concatenate.

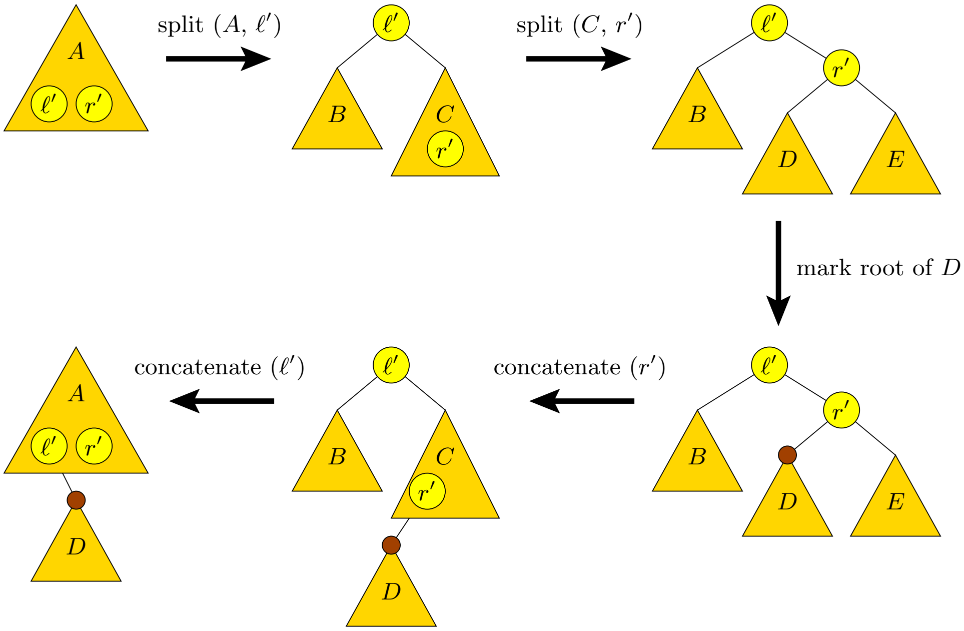


Fig. 1. *Implementing cut with split, mark, and concatenate.*

To cut an augmented tree A at depth d, first observe that the nodes of depth greater than d form an interval of key space within A. Using the augmented maximum depth of each subtree, we can find the node ℓ of minimum key value that has depth greater than d in $O(\lg k)$ time, by starting at the root and repeatedly walking to the leftmost child whose subtree has maximum depth greater than d. Symmetrically, we can find the node r of maximum key value that has depth greater than d. We also compute the predecessor ℓ' of ℓ and the successor r' of r.

With the interval $[\ell, r]$, or equivalently the open interval (ℓ', r'), defining the range of interest, we manipulate the trees using split and concatenate as shown in Figure 1. First we split A at ℓ' to form two subtrees B and C of ℓ' corresponding to key ranges $(-\infty, \ell')$ and (ℓ', ∞). (We skip this step, and the subsequent concatenate at ℓ', if $\ell' = -\infty$.) Then we split C at r' to form two subtrees D and E of r' corresponding to key ranges (ℓ', r') and (r', ∞). (We skip this step, and the subsequent concatenate at r', if $r' = \infty$.) Now we mark the root of D, effectively splitting D off from the remaining tree. The elements in D have keys in the range (ℓ', r'), which is equivalent to the range $[\ell, r]$, which are precisely the nodes of depth greater than d. Next we concatenate at r', which to the red-black tree appears to have no left child; thus the concatenation simply forms a red-black tree on r' and the nodes in its right subtree. Finally we concatenate at ℓ', effectively merging all nodes except those in D. The resulting tree therefore has all nodes of depth at most d.

Joining two augmented trees A and B is similar, except that we unmark instead of mark. First we determine which tree stores nodes of depth larger than all nodes in the other tree by comparing the depths of the roots of A and B. Suppose by relabeling that A stores nodes of larger depth. Symmetric to cuts, observe that the nodes in B have key values that fall in between two adjacent keys ℓ' and r' in A. We can find these keys by searching in A for the key of B's root. Indeed, if we split A at

ℓ' and then r' (skipping a split and the subsequent concatenate in the case of $\pm\infty$), the marked root of B becomes the left child of r'. Then we unmark the root of B, concatenate at r', and then concatenate at ℓ'. The result is a single tree containing all elements from A and B.

3.3. Tango algorithm. Now we describe how to construct the new state T_i of the BST given the previous state T_{i-1} and the next access x_i. The access algorithm follows a normal BST walk in T_{i-1} toward the query key x_i. Accessing x_i changes the necessary preferred children to make a preferred path from the root to x_i, sets the preferred child of x_i to the left, and does not change any other preferred children. Except for the last change to x_i's preferred child, the points of change in preferred children correspond exactly to where the BST walk in T_{i-1} crosses from one augmented tree to the next, i.e., where the walk visits a marked node. Thus, when the walk visits a marked node x, we cut the auxiliary tree containing the parent of x, cutting at a depth one less than the minimum depth of nodes in the auxiliary tree rooted at x, and then we join the resulting top path with the augmented tree rooted at x. Finally, when we reach x_i, we cut its auxiliary tree at the depth of x_i and join the resulting top path with the auxiliary tree rooted at the preceding marked node of x_i.

3.4. Analysis.

LEMMA 3.1. *The running time of an access x_i is $O((k+1)\,(1+\lg\lg n))$, where k is the number of nodes whose preferred child changes during access x_i.*

Proof. The running time consists of two parts: the cost of searching for x_i and the cost of rearranging the structure from state T_{i-1} into state T_i.

The search visits a root-to-x_i path in T_{i-1}, which we partition into subpaths according to the auxiliary trees visited. The transition between two auxiliary trees corresponds one-to-one to the edge between a node and its nonpreferred child in the root-to-x_i path in P, at which a node's preferred child changes because of this access. Thus the search path in T_{i-1} partitions into at most $k+1$ subpaths in $k+1$ auxiliary trees. The cost of the search within a single auxiliary tree is $O(\lg\lg n)$ because each auxiliary tree stores $O(\lg n)$ elements, corresponding to a subpath of a root-to-leaf path in P. Therefore the total search cost for x_i is $O((k+1)\,(1+\lg\lg n))$.

The update cost is the same as the search cost up to constant factors. For each of the at most $k+1$ auxiliary trees visited by the search, we perform one cut and one join, each costing $O(\lg\lg n)$. We also pay $O(\lg\lg n)$ to find the preceding marked node of x_i. The total cost is thus $O((k+1)\,(1+\lg\lg n))$. □

Define the *interleave bound* $\mathrm{IB}_i(X)$ *of access* x_i to be the interleave bound on the prefix $x_1, x_2, \ldots, x_i$ of the access sequence minus the interleave bound on the shorter prefix $x_1, x_2, \ldots, x_{i-1}$. In other words, the interleave bound of access x_i is the number of additional interleaves introduced by access x_i.

LEMMA 3.2. *The number of nodes whose preferred child changes from left to right or from right to left during an access x_i is equal to the interleave bound $\mathrm{IB}_i(X)$ of access x_i.*

Proof. The preferred child of a node y in P changes from left to right precisely when the previous access within y's subtree in P was in the left region of y and the next access x_i is in the right region of y. Symmetrically, the preferred child of node y changes from right to left precisely when the previous access within y's subtree in P was in the right region of y and the next access x_i is in the left region of y. Both of these events correspond exactly to interleaves. Note that these events do not include when node y previously had no preferred child and the first node within y's subtree in P is accessed. □

Theorem 3.3. *The running time of the Tango BST on a sequence X of m accesses over the universe $\{1, 2, \ldots, n\}$ is $O((\mathrm{OPT}(X)+n)\,(1+\lg\lg n))$, where $\mathrm{OPT}(X)$ is the cost of the offline optimal BST servicing X.*

Proof. Lemma 3.2 states that the total number of times a preferred child changes from left to right or from right to left is at most $\mathrm{IB}(X)$. There can be at most n first preferred child settings (i.e., changes from no preferred child to a left or right preference). Therefore the total number of preferred child changes is at most $\mathrm{IB}(X)+n$. Combining this bound with Lemma 3.1, the total cost of Tango is $O((\mathrm{IB}(X)+n+m)\,(1+\lg\lg n))$. On the other hand, Lemma 2.1 states that $\mathrm{OPT}(X) \geq \mathrm{IB}(X)/2-n$. A trivial lower bound on all access sequences X is that $\mathrm{OPT}(X) \geq m$. Therefore, the running time of Tango is $O((\mathrm{OPT}(X)+n)\,(1+\lg\lg n))$. □

Corollary 3.4. *When $m = \Omega(n)$, the running time of the Tango BST is $O(\mathrm{OPT}(X)\,(1+\lg\lg n))$.*

3.5. Tightness of approach. Observe that we cannot hope to improve the competitive ratio beyond $\Theta(\lg\lg n)$ using the current lower bound. At each moment in time, the preferred path from the root of P contains $\lg(n+1)$ nodes. Regardless of how the BST is organized, one of these $\lg(n+1)$ nodes must have depth $\Omega(\lg\lg n)$, which translates into a cost of $\Omega(\lg\lg n)$ for accessing that node. On the other hand, accessing any of these nodes increases the interleave bound by at most 1. Suppose we access node x along the preferred path from the root of P. The preferred children do not change for the nodes below x in the preferred path, nor do they change for the nodes above x. The preferred child of only x itself may change, in the case that the former preferred child was the right child, because we defined the preferred child of a just-accessed node x to be the left child. In conclusion, at any time, there is an access that costs $\Omega(\lg\lg n)$ in any fixed BST data structure, yet increases the interleave lower bound by at most 1, resulting in a ratio of $\Omega(\lg\lg n)$.

Appendix. Proof of interleave lower bound. In this appendix, we prove Theorem 2.1. We assume a fixed but arbitrary BST access algorithm, and argue that the time it takes is at least the interleave bound. Let T_i denote the state of this arbitrary BST after the execution of accesses $x_1, x_2, \ldots, x_i$.

Consider the interleaving through a node y in P. Define the *transition point* for y at time i to be the minimum-depth node z in the BST T_i such that the path from z to the root of T_i includes a node from the left region of y and a node from the right region of y. (Here we ignore nodes not from y's subtree in P.) Thus the transition point z is in either the left or the right region of y, and it is the first node of that type seen along this root-to-node path. Intuitively, any BST access algorithm applied both to an element in the left region of y and to an element in the right region of y must touch the transition point for y at least once.

First we show that the notion of transition point is well-defined.

Lemma A.1. *For any node y in P and any time i, there is a unique transition point for y at time i.*

Proof. Let ℓ be the lowest common ancestor of all nodes in T_i that are in the left region of y. Because the lowest common ancestor of any two nodes in a binary search tree has a key value nonstrictly between these two nodes, ℓ is in the left region of y. Thus ℓ is the unique node of minimum depth in T_i among all nodes in the left region of y. Similarly, the lowest common ancestor r of all nodes in T_i in the right region of y is in the right region of y and has the lowest depth among all such nodes. Also, the lowest common ancestor in T_i of all nodes in the left and right regions of y must be in either the left or right region of y (because they are consecutive in key space), and among such nodes it must be the unique node of minimum depth, so it must be either ℓ or r (whichever has smaller depth). Assume by symmetry that it is ℓ, so that ℓ is an ancestor of r. Thus r is a transition point for y in T_i, because the path in T_i from the root to r visits at least one node (ℓ) from the left region of y in P, and visits only one node (r) from the right region of y in P because it has minimum depth among such nodes. Furthermore, any path in T_i from the root must visit ℓ before any other node in the left or right region of y, because ℓ is an ancestor of all such nodes, and similarly it must visit r before any other node in the right region of y because it is an ancestor of all such nodes. Therefore r is the unique transition point for y in T_i. □

Second we show in the following lemma that the transition point is "stable," not changing until it is accessed.

LEMMA A.2. *If the BST access algorithm does not touch a node z in T_i for all i in the time interval $[j, k]$, and z is the transition point for a node y at time j, then z remains the transition point for node y for the entire time interval $[j, k]$.*

Proof. Define ℓ and r as in the proof of the previous lemma, and assume by symmetry that ℓ is an ancestor of r in T_j, so that r is the transition point for y at time j. Because the BST access algorithm does not touch r, it does not touch any node in the right region of y, and thus r remains the lowest common ancestor of these nodes. On the other hand, the algorithm may touch nodes in the left region of y, and in particular the lowest common ancestor $\ell = \ell_i$ of these nodes may change with time (i). Nonetheless, we claim that ℓ_i remains an ancestor of r. Because nodes in the left region of y cannot newly enter r's subtree in T_i, and y is initially outside this subtree, some node ℓ'_i in the left region of y must remain outside this subtree in T_i. As a consequence, the lowest common ancestor a_i of ℓ'_i and r cannot be r itself, so it must be in the left region of y. Thus ℓ_i must be an ancestor of a_i, which is an ancestor of r, in T_i. □

Next we prove that these transition points are different over all nodes in P, enabling us to charge to them.

LEMMA A.3. *At any time i, no node in T_i is the transition point for multiple nodes in P.*

Proof. Consider any two nodes y_1 and y_2 in P, and define ℓ_j and r_j in terms of y_j as in the proof of Lemma A.1. Recall that the transition point for y_j is either ℓ_j or r_j, whichever is deeper. If y_1 and y_2 are not ancestrally related in P, then their left and right regions are disjoint from each other; thus ℓ_1 and r_1 are distinct from ℓ_2 and r_2, and hence the transition points for y_1 and y_2 are distinct. Otherwise, suppose by symmetry that y_1 is an ancestor of y_2 in P. If the transition point for y_1 is not in y_2's subtree in P (e.g., it is y_1, or it is in the left or right subtree of y_1 in P while y_2 is in the opposite subtree of y_1 in P), then it differs from ℓ_2 and r_2 and thus the transition point for y_2. Otherwise, the transition point for y_1 is the lowest common ancestor of all nodes in y_2's subtree in P, and thus it is either ℓ_2 or r_2, whichever is less deep. On the other hand, the transition point for y_2 is either ℓ_2 or r_2, whichever is deeper. Therefore the two transition points differ in all cases. □

Finally we prove that the interleave bound is a lower bound.

THEOREM 2.1. *$\mathrm{IB}(X)/2-n$ is a lower bound on $\mathrm{OPT}(X)$, the cost of the optimal offline BST that serves access sequence X.*

Proof. Instead of counting the entire cost incurred by the (optimal offline) BST, we just count the number of transition points it touches (which can be only smaller). By

Lemma A.3, we can count the number of times the BST touches the transition point for y, separately for each y, and then sum these counts. Define ℓ and r as in the proof of Lemma A.1, so that the transition point for y is always either ℓ or r, whichever is deeper. Consider a maximal ordered subsequence $x_{i_1}, x_{i_2}, \ldots, x_{i_p}$ of accesses to nodes that alternate between being in the left and right regions of y. Thus p is the amount of interleaving through y. Assume by symmetry that the odd accesses $x_{i_{2j-1}}$ are nodes in the left region of y, and the even accesses $x_{i_{2j}}$ are nodes in the right region of y. Consider each j with $1 \leq j \leq \lfloor p/2 \rfloor$. Any access to a node in the left region of y must touch ℓ, and any access to a node in the right region of y must touch r. Thus, for both accesses $x_{i_{2j-1}}$ and $x_{i_{2j}}$ to avoid touching the transition point for y, the transition point must change from r to ℓ in between, which by Lemma A.2 requires touching the transition point for y. Thus the BST access algorithm must touch the transition point for y at least once during the time interval $[i_{2j-1}, i_{2j}]$. Summing over all j, the BST access algorithm must touch the transition point for y at least $\lfloor p/2 \rfloor \geq p/2 - 1$ times. Summing over all y, the amount p of interleaving through y adds up to the interleave bound $\mathrm{IB}(X)$; thus the number of transition points touched adds up to at least $\mathrm{IB}(X)/2 - n$. $\square$

Acknowledgments. We thank Richard Cole, Martin Farach-Colton, Michael L. Fredman, and Stefan Langerman for many helpful discussions.

REFERENCES

[BCK03] A. Blum, S. Chawla, and A. Kalai, *Static optimality and dynamic search-optimality in lists and trees*, Algorithmica, 36 (2003), pp. 249–260.

[BD04] M. Bădoiu and E. D. Demaine, *A simplified and dynamic unified structure*, in Proceedings of the 6th Latin American Symposium on Theoretical Informatics (Buenos Aires, 2004), Lecture Notes in Comput. Sci. 2976, Springer-Verlag, Berlin, 2004, pp. 466–473.

[CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed., MIT Press, Cambridge, MA, 2001.

[CMSS00] R. Cole, B. Mishra, J. Schmidt, and A. Siegel, *On the dynamic finger conjecture for splay trees. Part* I: *Splay sorting* $\log n$*-block sequences*, SIAM J. Comput., 30 (2000), pp. 1–43.

[Col00] R. Cole, *On the dynamic finger conjecture for splay trees. Part* II: *The proof*, SIAM J. Comput., 30 (2000), pp. 44–85.

[DHIP04] E. D. Demaine, D. Harmon, J. Iacono, and M. Pătraşcu, *Dynamic optimality—Almost*, in Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, Rome, 2004, pp. 484–490.

[HF98] H. Hampapuram and M. L. Fredman, *Optimal biweighted binary trees and the complexity of maintaining partial sums*, SIAM J. Comput., 28 (1998), pp. 1–9.

[Iac01] J. Iacono, *Alternatives to splay trees with* $O(\log n)$ *worst-case access times*, in Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms, Washington, DC, 2001, pp. 516–522.

[Iac05] J. Iacono, *Key-independent optimality*, Algorithmica, 42 (2005), pp. 3–10.

[Knu71] D. E. Knuth, *Optimum binary search trees*, Acta Inform., 1 (1971), pp. 14–25.

[PD06] M. Pătraşcu and E. D. Demaine, *Logarithmic lower bounds in the cell-probe model*, SIAM J. Comput., 35 (2006), pp. 932–963.

[ST83] D. D. Sleator and R. E. Tarjan, *A data structure for dynamic trees*, J. Comput. System Sci., 24 (1983), pp. 362–391.

[ST85] D. D. Sleator and R. E. Tarjan, *Self-adjusting binary search trees*, J. ACM, 32 (1985), pp. 652–686.

[Tar83] R. E. Tarjan, *Linking and cutting trees*, in Data Structures and Network Algorithms, CBMS-NSF Regional Conf. Ser. in Appl. Math. 44, SIAM, Philadelphia, 1983, pp. 59–70.

[Tar85] R. E. Tarjan, *Sequential access in splay trees takes linear time*, Combinatorica, 5 (1985), pp. 367–378.

[WDS06] C. C. Wang, J. Derryberry, and D. D. Sleator, $O(\log \log n)$*-competitive dynamic binary search trees*, in Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms, Miami, 2006, pp. 374–383.

[Wil89] R. Wilber, *Lower bounds for accessing binary search trees with rotations*, SIAM J. Comput., 18 (1989), pp. 56–67.

SIAM J. Comput.
Vol. 37, No. 1, pp. 252–266

ALGEBRAS WITH POLYNOMIAL IDENTITIES AND COMPUTING THE DETERMINANT[*]

STEVE CHIEN[†] AND ALISTAIR SINCLAIR[‡]

Abstract. In 1991, Nisan proved an exponential lower bound on the size of an algebraic branching program (ABP) that computes the determinant of a matrix in the noncommutative "free algebra" setting, in which there are no nontrivial relationships between the matrix entries. By contrast, when the matrix entries commute there are polynomial size ABPs for the determinant. This paper extends Nisan's result to a much wider class of noncommutative algebras, including all nontrivial matrix algebras over any field of characteristic 0, group algebras of all nonabelian finite groups over algebraically closed fields of characteristic 0, the quaternion algebra, and the Clifford algebras. As a result, we obtain more compelling evidence for the essential role played by commutativity in the efficient computation of the determinant. The key to our approach is a characterization of noncommutative algebras by means of the *polynomial identities* that they satisfy. Extending Nisan's lower bound framework, we find that any reduction in complexity compared to the free algebra must arise from the ability of the identities to reduce the rank of certain naturally associated matrices. Using results from the theory of algebras with polynomial identities, we are able to show that none of the identities of the above classes of algebras is able to achieve such a rank reduction.

Key words. determinants, permanents, algebraic computation, complexity lower bounds, algebraic branching programs, polynomial identities

AMS subject classifications. Primary, 68Q17; Secondary, 16R10, 65F40, 68Q25

DOI. 10.1137/S0097539705447359

1. Introduction.

All known polynomial time algorithms for computing the determinant of a matrix appear to rely on the fact that multiplication in the underlying field (in which the matrix entries reside) is commutative. How hard is it to compute the determinant in a noncommutative setting? This question is motivated by the broader aim of understanding the computational power of commutativity [18, 21], as well as by recent algorithmic applications of determinants over noncommutative algebras to approximating the permanent [3, 5].

In a landmark paper [18], Nisan pioneered the study of noncommutative computation. His main result was an exponential lower bound (of the form $\Omega(2^n)$) for the size of any algebraic branching program (ABP) that computes the determinant of an $n \times n$ matrix, viewed as a formal algebraic expression whose indeterminates $\{x_{11}, x_{12}, \ldots, x_{nn}\}$ do not commute. Since the determinant can be computed by an ABP of size $O(n^3)$ in the commutative setting [9, 16, 17, 23], this provides intriguing evidence of the computational power of commutativity.[1] Very recently, the ABP

[*] Received by the editors April 1, 2005; accepted for publication (in revised form) May 28, 2006; published electronically May 14, 2007.

http://www.siam.org/journals/sicomp/37-1/44735.html

[†] Microsoft Research, 1065 La Avenida, Mountain View, CA 94043 (schien@microsoft.com). Part of this work was done while this author was at the Computer Science Division, UC Berkeley, supported by NSF grants CCR-9820951 and CCR-0121555.

[‡] Computer Science Division, University of California at Berkeley, Berkeley, CA 94720-1776 (sinclair@cs.berkeley.edu). This author was supported in part by NSF ITR grant CCR-0121555. Part of this work was done while this author was on sabbatical leave at Microsoft Research, Redmond, CA, and at Université Paris Sud and Ecole Polytechnique, Paris.

[1] Nisan in fact stated his result in terms of formula size, rather than ABP complexity. His exponential lower bound on ABP size translates directly to a similar bound on formula size, which he contrasted with the fact that the determinant can be computed by a formula of size $n^{O(\log n)}$ in the commutative setting.

model has been used by Raz and Shpilka [21] to give an efficient deterministic algorithm for polynomial identity testing over noncommutative formulas.

The main limitation of Nisan's result, and of the ABP model as used to date, is that it is restricted to the free algebra $\mathbb{F}\langle x_{11}, \ldots, x_{nn}\rangle$ over a field $\mathbb{F}$, in which not only does commutativity fail to hold, but there is no interesting structure at all (i.e., no nontrivial relations hold between the indeterminates). However, it remains quite conceivable that, in some specific noncommutative algebra (e.g., the quaternion algebra), the determinant can be computed efficiently. It seems important that any comparison of commutative and noncommutative computation consider all noncommutative algebras, not only the free algebra.

In this paper, we address this issue by proposing a framework that allows the algebras to have much more structure than the free algebra, without being commutative. We characterize an algebra over $\mathbb{F}$ by the *polynomial identities* that it satisfies. In this view, commutative $\mathbb{F}$-algebras (when $\mathbb{F}$ has characteristic 0) are characterized by the polynomial identity $x_1x_2 - x_2x_1 = 0$. By contrast, the free algebra $\mathbb{F}\langle x_1, x_2, \ldots\rangle$ satisfies no nontrivial identities. Adding polynomial identities (without including the commutative identity) creates a spectrum of algebras between these two extremes. The study of polynomial identities has been an active topic in algebra for the past fifty years (see the book [7] for a survey). Our aim is to use the machinery of that field to study the power of noncommutativity in a manner that is more sensitive to the structure of the algebra; in particular, lower bounds for algebras that admit a rich class of identities give more compelling evidence for the importance of commutativity than that provided by Nisan's result.

Our first step is to extend Nisan's lower bound framework for ABPs, based on the combinatorial structure of the monomials of the function being computed, as expressed in the rank of certain naturally associated matrices. This simple step leads to a useful tool for comparing the ABP complexities in the free algebra and in the algebra of interest: essentially, any reduction in complexity corresponds to an ability of the polynomial identities to reduce the rank of the associated matrices.

We then go on to apply this framework to obtain exponential lower bounds for the ABP complexity of the determinant over a range of natural noncommutative algebras. The first class we consider are *matrix algebras*, whose elements are $d \times d$ matrices over a field $\mathbb{F}$. This is probably the most natural and most widely studied family of noncommutative algebras and has also arisen in connection with approximating the permanent [3, 5]. It is known [1] that the algebra of $d \times d$ matrices satisfies the symmetric polynomial identity $s_{2d}(x_1, \ldots, x_{2d}) = 0$, where $s_k(x_1, \ldots, x_k)$ is defined to be $\sum_\sigma \operatorname{sgn}(\sigma) \prod_{i=1}^k x_{\sigma i}$, with the sum ranging over all k-permutations σ. Thus s_2 is the commutative identity, and s_k for $k > 2$ can be viewed as higher-order generalizations of commutativity. Our first main result is an exponential ABP lower bound for the determinant over any nontrivial matrix algebra.

THEOREM 1.1. *Any ABP for computing the determinant of an $n \times n$ matrix whose entries belong to the algebra of $d \times d$ matrices, $d \geq 2$, over a field $\mathbb{F}$ of characteristic* 0 *has size at least 2^n.*

We then go on to apply the above result, together with some additional observations, to deduce a similar lower bound for computing the determinant over several other classes of noncommutative algebras. These results are summarized in the following theorem.

THEOREM 1.2. *The same lower bound as in Theorem* 1.1 *holds for ABPs computing the determinant over any of the following algebras:*

1. *the algebra of $d\times d$ upper triangular matrices over a field $\mathbb{F}$ of characteristic* 0*, for any $d \geq 2$;*
2. *the quaternion algebra, and indeed all higher Clifford algebras;*[2]
3. *the group algebra of any finite, nonabelian group G over any algebraically closed field $\mathbb{F}$ of characteristic* 0 *(i.e., the algebra whose elements are $\mathbb{F}$-vectors indexed by group elements, with multiplication inherited from the group).*

To conclude this introduction, we briefly mention some related work on lower bounds for the determinant and permanent in other restricted models of computation. In addition to the noncommutative case studied here and in [18], exponential lower bounds are known for other restricted models including formulas of depth 3 (over finite fields) [10, 11] and various restricted classes of multilinear formulas [19, 21]. In a recent breakthrough, Raz [20] obtained a superpolynomial bound (of the form $n^{\Omega(\log n)}$) on the size of an arbitrary multilinear formula for the permanent or the determinant (over any field). By contrast, the best known lower bound for the size of general arithmetic formulas for the determinant is $\Omega(n^3)$ [12].

The remainder of the paper is organized as follows. In section 2 we provide necessary background on ABPs and polynomial identities. In section 3 we review Nisan's framework for lower bounds based on rank and extend it to general algebras with polynomial identities. We apply this framework to obtain an exponential lower bound for matrix algebras (Theorem 1.1) in section 4, and for several other algebras (as enumerated in Theorem 1.2) in section 5. We conclude in section 6 with a discussion of some limitations of our results and some suggestions for future work.

2. Background.

2.1. Algebras with polynomial identities.

Let $\mathbb{F}$ be a field and $\mathcal{A}$ an associative algebra over $\mathbb{F}$, or an $\mathbb{F}$-*algebra* (i.e., $\mathcal{A}$ is a vector space over $\mathbb{F}$ together with a distributive multiplication operation). Note that we will always assume that multiplication in $\mathcal{A}$ is associative, but it need *not* be commutative. We also assume the existence of a multiplicative unity. Familiar examples of $\mathbb{F}$-algebras are the following:

1. $\mathbb{F}\langle X\rangle$, the *free algebra* over $\mathbb{F}$ generated by a countable set of indeterminates $X = \{x_1, x_2, \ldots\}$, corresponding to all polynomials with coefficients in $\mathbb{F}$ in which no nontrivial relationships exist between the indeterminates;
2. $\mathbb{F}[X]$, the standard *polynomial ring* over $\mathbb{F}$, corresponding to polynomials over $\mathbb{F}$ in which the indeterminates commute;
3. the *matrix algebra* $\mathrm{Mat}_d(\mathbb{F})$, consisting of all $d \times d$ matrices with entries in $\mathbb{F}$;
4. for any group G, the *group algebra* $\mathbb{F}G$, whose elements are vectors of the form $\sum_{g\in G} c_g g$ with $c_g \in \mathbb{F}$, with multiplication defined by $(\sum_{g\in G} a_g g)(\sum_{h\in G} b_h h) = \sum_{k\in G}\sum_{g,h:gh=k}(a_g b_h)k$.

We now introduce the central concept of *polynomial identities*, which we shall use to characterize different $\mathbb{F}$-algebras. We shall limit our treatment to the essentials; for more background on this topic, see the monograph [7].

DEFINITION 2.1. *Let $\mathcal{A}$ be any algebra over $\mathbb{F}$. A polynomial $z(x_1, \ldots, x_m) \in \mathbb{F}\langle X\rangle$ is a* polynomial identity *of $\mathcal{A}$ if and only if $z(a_1, \ldots, a_m) = 0$ for all $a_1, \ldots, a_m \in \mathcal{A}$.*

It is well known (see, e.g., [7]) that the set of all polynomial identities of a given algebra $\mathcal{A}$ forms a *T-ideal* of $\mathbb{F}\langle X\rangle$, i.e., a two-sided ideal that is closed under all

[2] See, e.g., [14] for a definition. These algebras were used in [5] in connection with approximating the permanent.

endomorphisms of $\mathcal{A}$.[3] Thus if $z(x_1, \ldots, x_m)$ is an identity of $\mathcal{A}$, then substitution for each of the x_i by an arbitrary element of $\mathbb{F}\langle X\rangle$ also yields an identity. We denote this T-ideal by $T(\mathcal{A})$. We say that a set of identities $B \subseteq T(\mathcal{A})$ is a *basis* or *generating set* of $T(\mathcal{A})$ if every element of $T(\mathcal{A})$ can be expressed as a linear combination of the form

$$\sum_{\ell} \alpha_\ell z_\ell(g_{1\ell}, \ldots, g_{m\ell})\beta_\ell,$$

with $z_\ell \in B$ and $\alpha_\ell, \beta_\ell, g_{i\ell} \in \mathbb{F}\langle X\rangle$.

For example, the free algebra $\mathbb{F}\langle X\rangle$ has no nontrivial identities, while any commutative algebra over a field $\mathbb{F}$ of characteristic 0 has a generating set consisting of the single identity $x_1x_2 - x_2x_1$. The study of polynomial identities has been an active topic in algebra for the past fifty years (see [7] for a survey), but it remains an important open problem to find (minimal) generating sets for most widely studied algebras.[4] A celebrated theorem of Amitsur and Levitzki [1] says that the matrix algebra $\mathrm{Mat}_d(\mathbb{F})$ satisfies the identity s_{2d}, where s_k is defined as

$$s_k(x_1, \ldots, x_k) = \sum_{\sigma \in \mathcal{S}_k} \mathrm{sgn}(\sigma) \prod_{i=1}^{k} x_{\sigma i}$$

and is known as the *standard identity* of degree k. (Note that the commutative identity $x_1x_2 - x_2x_1$ is the standard identity s_2; the standard identities can be viewed as natural, progressively weaker generalizations of commutativity.) Moreover, $\mathrm{Mat}_d(\mathbb{F})$ satisfies no identities of lower degree. For $d = 2$ (the 2×2 matrix algebra), it has been shown relatively recently by Drensky [6] that if $\mathbb{F}$ has characteristic 0, then s_4 together with the *Hall identity*

$$h(x_1, x_2) = [[x_1, x_2]^2, x_1]$$

(where $[a, b]$ denotes the "commutator" $ab - ba$) forms a minimal generating set. This fact will be crucial to the present paper. Generating sets (let alone minimal ones) for $\mathrm{Mat}_d(\mathbb{F})$ are not known for any $d > 2$; indeed, this remains one of the central open questions in the area of polynomial identities. (See [4] for recent computer-assisted efforts in this direction.) Note, however, that any identity satisfied by $\mathrm{Mat}_d(\mathbb{F})$ for $d > 2$ is also satisfied by $\mathrm{Mat}_2(\mathbb{F})$ (but not conversely).

Two other facts about polynomial identities will be useful later as well. First, if $\mathbb{F}$ has characteristic 0, then for any $\mathbb{F}$-algebra $\mathcal{A}$ the T-ideal $T(\mathcal{A})$ of polynomial identities of $\mathcal{A}$ has a generating set consisting entirely of *multilinear* identities. (See [7, Proposition 4.2.3] for a proof; a polynomial $f(x_1, \ldots, x_n)$ is considered multilinear if x_i has degree 1 in each monomial of f for all $1 \leq i \leq n$.) Second, if z is a polynomial identity of $\mathcal{A}$, then each homogeneous component of z (where the homogeneous component of z of degree k is the polynomial consisting of all terms in z whose total degree is k) is itself an identity; this follows from a slight variation of the proof of Proposition 4.2.3(i) in [7].

[3] Indeed, there is a 1-1 correspondence between the T-ideals of $\mathbb{F}\langle X\rangle$ and the varieties of algebras satisfying a given set of polynomial identities.

[4] It is known that any algebra over a field of characteristic 0 has a finite generating set for its polynomial identities [13].

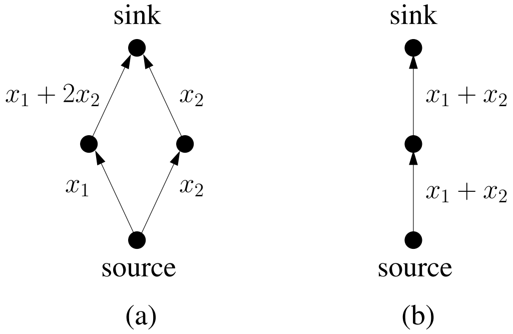


FIG. 1. (a) *An ABP that computes* $x_1^2 + 2x_1x_2 + x_2^2$ *over* $\mathbb{R}\langle X\rangle$ *and* (b) *an ABP that computes the same function over* $\mathbb{R}$.

2.2. Algebraic branching programs. ABPs were introduced by Nisan [18] as an algebraic analogue of standard (arithmetic) branching programs.

DEFINITION 2.2. *An* algebraic branching program (ABP) *is a directed acyclic graph with one source and one sink. The vertices of the graph are partitioned into levels numbered from* 0 *to* d *(the* degree *of the ABP), and edges may go only from level* i *to level* $i+1$. *The source is the only vertex at level* 0, *and the sink is the only vertex at level* d. *Each edge is labeled with a homogeneous linear polynomial in indeterminates* x_i *(i.e., a function of the form* $\sum_i c_i x_i$, *with coefficients* $c_i \in \mathbb{F}$*). The* size *of an ABP is the number of vertices.*

An ABP computes the degree-d homogeneous polynomial in $\mathbb{F}\langle X\rangle$ that is the sum, over all paths from the source to the sink, of the product of the linear functions associated with the edges along that path. Figure 1(a) shows a toy example of a branching program that computes $f(x_1, x_2) = x_1^2 + 2x_1x_2 + x_2^2$ in $\mathbb{R}\langle X\rangle$. Note that the order of multiplication is important and follows the order of the paths.

For a homogeneous polynomial $f \in \mathbb{F}\langle X\rangle$ over n variables $x_1, \ldots, x_n$, the *branching program complexity* $B(f)$ is defined as the size of a smallest ABP that computes f. (We shall see in the next section that $B(f) = 4$ for the above polynomial f, and so the ABP in Figure 1(a) is optimal.)

We stress that the original concepts of ABPs and branching program complexity refer to computation in the *free algebra* $\mathbb{F}\langle X\rangle$, where there are no nontrivial relations among the indeterminates x_i. However, we can generalize them rather naturally to describe computation over any $\mathbb{F}$-algebra $\mathcal{A}$, as stated in the following definition.

DEFINITION 2.3. *An ABP computes a function* $f(x_1, \ldots, x_n)$ *over* $\mathcal{A}$ *if and only if, for all substitutions of the indeterminates* x_i *by values* $a_i \in \mathcal{A}$, *the output of the ABP is* $f(a_1, \ldots, a_n)$.

The branching program complexity of f over an algebra $\mathcal{A}$ will be denoted by $B_{\mathcal{A}}(f)$ and is defined as the size of the smallest ABP that computes f over $\mathcal{A}$. (The unadorned notation $B(f)$ is reserved for the free algebra complexity of f.)

Note that in a specific algebra $\mathcal{A}$, branching programs may be able to take advantage of polynomial identities in $\mathcal{A}$ to reduce the complexity of computing some functions. As an example, suppose we are working over the real numbers $\mathbb{R}$ (viewed trivially as an $\mathbb{R}$-algebra). Then the branching program shown in Figure 1(b) computes the same toy polynomial f as above by making use of the identity $x_1x_2 - x_2x_1$ to instead compute the equivalent polynomial $x_1^2 + x_1x_2 + x_2x_1 + x_2^2$. Clearly since f has degree 2, this ABP must be optimal, and so $B_{\mathbb{R}}(f) = 3$.

Although it is not a major concern of this paper, we note that the measure $B(f)$ can be related to other measures such as formula size $F(f)$. For example, for any homogeneous polynomial f of degree d, we have (see [18])

$$B(f) \leq d(F(f)+1), \qquad F(f) \leq (nB(f))^{O(\log d)}.$$

2.3. ABPs for the determinant. We conclude this section by discussing ABPs in the context of the determinant function, which will be our main application. The *determinant* of an $n \times n$ matrix with entries $x_{11}, \ldots, x_{nn}$ is defined by

$$\det_n(x_{11}, \ldots, x_{nn}) = \sum_{\sigma \in \mathcal{S}_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} x_{i,\sigma i}.$$

As explained in the introduction, we are interested in the branching program complexity of $\det_n$ (as a function of n) over various $\mathbb{F}$-algebras $\mathcal{A}$. Nisan showed in [18] that $B(\det_n) = 2^n$ over the free (real) algebra $\mathbb{R}\langle X\rangle$. Recall that this algebra is noncommutative and indeed has no interesting structure. At the other extreme, we may consider the situation in which we are working over a commutative algebra, such as $\mathbb{R}$ itself. In this setting there exist *polynomial* size ABPs for $\det_n$, as the following theorem states.

Theorem 2.4. *Let $\mathcal{A}$ be any commutative algebra over a field $\mathbb{F}$. Then*

$$B_{\mathcal{A}}(\det_n) \leq O(n^3).$$

Note that some well-known determinant algorithms, such as Gaussian elimination, cannot be formulated as ABPs. However, there are a number of polynomial time combinatorial algorithms (e.g., [9, 16, 17, 23]) that can be used to prove Theorem 2.4. For completeness, and because it was not originally phrased as an ABP, we provide in the appendix a sketch of one of these, due to Mahajan and Vinay [16].

In the remainder of the paper, our goal will be to understand the complexity of computing the determinant in algebras between these two extremes.

3. A framework for lower bounds. In [18] Nisan introduced a characterization of the ABP complexity $B(f)$ (over the free algebra $\mathbb{F}\langle X\rangle$) in terms of the ranks of certain matrices describing the combinatorial structure of the monomials of f. In this section we first briefly describe Nisan's framework and then extend it to computation over general $\mathbb{F}$-algebras.

Let $f(x_1, \ldots, x_n)$ be a homogeneous polynomial of degree d. For each $0 \leq k \leq d$, $M_k(f)$ denotes an $n^k \times n^{d-k}$ matrix with entries in $\mathbb{F}$ as follows. Each row of $M_k(f)$ corresponds to an (ordered) monomial of degree k, and each column corresponds to a monomial of degree $d-k$; the matrix entry at position $(x_{i_1} \cdots x_{i_k}, x_{j_1} \cdots x_{j_{d-k}})$ is the coefficient of the combined monomial $x_{i_1} \cdots x_{i_k} x_{j_1} \cdots x_{j_{d-k}}$ in f.

The following theorem gives a precise relationship between $B(f)$ and the matrices $M_k(f)$.

Theorem 3.1 (Nisan [18]). *Let $f \in \mathbb{F}\langle X\rangle$ be any homogeneous polynomial of degree d. Then*

$$B(f) = \sum_{k=0}^{d} \operatorname{rank}(M_k(f)).$$

By applying this theorem to the determinant function, Nisan shows that $B(\det_n) = 2^n$. (Indeed, this result applies to any polynomial that is *weakly equivalent* to the determinant, where two polynomials f and g are weakly equivalent if for each monomial in f there exists a monomial in g consisting of the same variables—possibly in a different multiplicative order and with a different nonzero coefficient—and vice versa. The permanent is an example of such a function.)

We now extend Nisan's results to handle not just $\mathbb{F}\langle X\rangle$ but all $\mathbb{F}$-algebras for fields $\mathbb{F}$ of characteristic 0. In particular, we characterize $B_{\mathcal{A}}(f)$ in terms of the polynomial identities of $\mathcal{A}$ as follows.

Theorem 3.2. *Let $\mathbb{F}$ be a field of characteristic 0, $f \in \mathbb{F}\langle X\rangle$ be a homogeneous polynomial of degree d, and $\mathcal{A}$ be any $\mathbb{F}$-algebra. Then if $f \notin T(\mathcal{A})$ (i.e., if f is not identically zero over $\mathcal{A}$), the ABP complexity of f over $\mathcal{A}$ is given by*

$$B_{\mathcal{A}}(f) = \inf_{z(\cdot)\equiv 0} B(f+z) = \inf_{z(\cdot)\equiv 0} \sum_{k=0}^{d} \mathrm{rank}(M_k(f+z)),$$

where the infimum is over the zero polynomial and all degree-d homogeneous polynomial identities z of $\mathcal{A}$.

Proof. The upper bound on $B_{\mathcal{A}}(f)$ is immediate from Definition 2.3. For the lower bound, let P be a branching program that computes $f(x_1,\ldots,x_n)$ over $\mathcal{A}$. By definition P computes a homogeneous formal polynomial $g \in \mathbb{F}\langle X\rangle$; we need to show that g is of the form $f + z$ for some homogeneous degree-d polynomial identity z of $\mathcal{A}$.

This is almost immediate. Since P computes f over $\mathcal{A}$, we have by definition that $f(a_1,\ldots,a_n) = g(a_1,\ldots,a_n)$ for all instantiations $a_i \in \mathcal{A}$. Hence if we define the function $z(x_1,\ldots,x_n) = g(x_1,\ldots,x_n) - f(x_1,\ldots,x_n)$, we have that $z(a_1,\ldots,a_n) = 0$ for all $a_i \in \mathcal{A}$, and thus z is either the zero polynomial or a polynomial identity. But $z = g - f$ with f, g homogeneous; therefore, recalling from section 2.1 that each of the homogeneous components of z is also a polynomial identity and using the assumption that f is not itself an identity, we may conclude that z is in fact homogeneous of degree d. Since P computes $g = f + z$, we are done.

The second equality in the theorem is a direct application of Nisan's result (Theorem 3.1). □

We can see a very simple application of this general framework by referring to the examples in Figure 1, where we are computing $f(x_1,x_2) = x_1^2 + 2x_1x_2 + x_2^2$. For the case of the free algebra $\mathbb{R}\langle X\rangle$, we apply Nisan's theorem and find that

$$M_0(f) = \begin{array}{c|cccc} & x_1^2 & x_1x_2 & x_2x_1 & x_2^2 \\ \hline 1 & 1 & 2 & 0 & 1 \end{array}, \quad M_1(f) = \begin{array}{c|cc} & x_1 & x_2 \\ \hline x_1 & 1 & 2 \\ x_2 & 0 & 1 \end{array},$$

$$M_2(f) = \begin{array}{c|c} & 1 \\ \hline x_1^2 & 1 \\ x_1x_2 & 2 \\ x_2x_1 & 0 \\ x_2^2 & 1 \end{array},$$

which implies $B(f) = \mathrm{rank}(M_0(f)) + \mathrm{rank}(M_1(f)) + \mathrm{rank}(M_2(f)) = 1 + 2 + 1 = 4$. Hence the ABP shown in Figure 1(a) is minimal.

When working over $\mathbb{R}$ rather than $\mathbb{R}\langle X\rangle$, we can add the identity $z(x_1,x_2) = x_2x_1 - x_1x_2$ to obtain $g(x_1,x_2) = f(x_1,x_2) + z(x_1,x_2) = x_1^2 + x_1x_2 + x_2x_1 + x_2^2$.

We then observe that

$$M_0(g) \;=\; \begin{array}{c|cccc} & x_1^2 & x_1x_2 & x_2x_1 & x_2^2 \\ \hline 1 & 1 & 1 & 1 & 1 \end{array}\;, \quad M_1(g) \;=\; \begin{array}{c|cc} & x_1 & x_2 \\ \hline x_1 & 1 & 1 \\ x_2 & 1 & 1 \end{array}\;,$$

$$M_2(g) \;=\; \begin{array}{c|c} & 1 \\ \hline x_1^2 & 1 \\ x_1x_2 & 1 \\ x_2x_1 & 1 \\ x_2^2 & 1 \end{array}\;,$$

and so the ABP complexity of f over $\mathbb{R}$ is $B_{\mathbb{R}}(f) \leq 1+1+1 = 3$. Since this is clearly minimal (f has degree 2), equality must hold and we confirm that the ABP in Figure 1(b) is optimal.

In principle, then, given any function f, we can compare its branching program complexity over any given algebra $\mathcal{A}$ to its free algebra complexity by determining all of the polynomial identities of $\mathcal{A}$ and checking if any of them are able to reduce the rank of the matrices $M_k(f)$.

3.1. General algebraic branching programs. It is a consequence of the ABP model that the only identities we need to consider in Theorem 3.2 are homogeneous (of the same degree as f). This might be seen as a weakness of our framework, as one may conceivably be able to exploit nonhomogeneous identities to reduce the complexity of f. (The ABP model is fully general for computation in the free algebra, for if f is homogeneous then there is no advantage in allowing the ABP to be nonhomogeneous.) We conclude this section by observing that, even allowing nonhomogeneous ABPs (for a homogeneous function f), we can obtain the bound

$$\hat{B}_{\mathcal{A}}(f)^2 \geq C \inf_{z(\cdot)\equiv 0} B(f+z) \tag{1}$$

for a universal constant C, where $\hat{B}$ is the size of the smallest *general* ABP that computes f over $\mathcal{A}$, and the infimum is over the zero polynomial and all homogeneous identities z of degree d. Thus at the cost of a square we can assume all identities are homogeneous. Since our lower bounds will be exponential, this square will affect only the constant in the exponent.

We prove (1) as follows. First consider a generalized setting for Theorem 3.2, in which the ABP model is extended in a natural way to allow computation of nonhomogeneous polynomials. This is done by allowing the edge labels to be arbitrary linear polynomials (including constants) and dropping the requirement that the graph be leveled. Arguing exactly as in the proof of Theorem 3.2, we have that for any homogeneous f of degree d,

$$\hat{B}_{\mathcal{A}}(f) = \inf_{\hat{z}(\cdot)\equiv 0} \hat{B}(f+\hat{z}), \tag{2}$$

where $\hat{B}$ denotes the smallest general ABP that computes a particular function and the infimum is over the zero polynomial and all identities $\hat{z}$ (not necessarily homogeneous) of $\mathcal{A}$. However, recall from section 2.1 that for any such $\hat{z}$, its homogeneous components are also identities; thus, in particular, the degree-d homogeneous component of $f+\hat{z}$ is of the form $f+z$, where z is homogeneous of degree d.

Now, given a (general) ABP of size s and depth ℓ for a function g, it is not hard to construct (see [21, Lemma 2.2]) for each d, a (standard) ABP of size $O(s\ell)$ that computes the degree-d homogeneous component of g. Thus there is an ABP of size $O(s\ell) = O(s^2)$ that computes $f + z$. Combining this with (2) yields (1), as claimed.

4. Computing the determinant over matrix algebras. As mentioned earlier, Nisan [18] showed that the ABP complexity of the determinant over the free algebra[5] $\mathbb{F}\langle X\rangle$ satisfies

$$B(\det_n) = 2^n. \tag{3}$$

In this section we prove a similar lower bound for a much wider class of noncommutative algebras, namely, for all *matrix algebras* over any field of characteristic 0. This is Theorem 1.1 from the introduction, which we restate here.

THEOREM 4.1. *For any $d \geq 2$, the ABP complexity of computing the determinant over the $d \times d$ matrix algebra over any field $\mathbb{F}$ of characteristic* 0 *is given by*

$$B_{\mathrm{Mat}_d(\mathbb{F})}(\det_n) = 2^n.$$

Proof. We observe that Nisan proves (3) via Theorem 3.1, by showing that each of the matrices $M_k(\det_n)$ has rank exactly $\binom{n}{k}$. (This actually follows rather easily as the matrices have a very special form.) Following our generalized framework of Theorem 3.2, our task is to show that, for any homogeneous, degree-n polynomial identity z of the matrix algebra $\mathrm{Mat}_d(\mathbb{F})$, the rank of $M_k(\det_n + z)$ remains at least $\binom{n}{k}$. In other words, we have to show that no identity of the matrix algebras can reduce the rank of $M_k(\det_n)$. Note that since any polynomial identity of $\mathrm{Mat}_d(\mathbb{F})$ for $d > 2$ is also an identity for $\mathrm{Mat}_2(\mathbb{F})$, it is sufficient to show this for the identities of $\mathrm{Mat}_2(\mathbb{F})$.

In fact, for each value of k, we will examine only a submatrix of $M_k(\det_n + z)$ and show that this submatrix already contains $\binom{n}{k}$ linearly independent rows. We define our submatrix as follows: for each subset $S = \{a_1, \ldots, a_k\}$ of size k, $1 \leq a_1 < \cdots < a_k \leq n$, we keep those rows that correspond to monomials of the form $\prod_{j=1}^{k} x_{\sigma j, a_{\sigma j}}$ for all permutations $\sigma \in \mathcal{S}_k$; for each such subset, we assume that its rows are contiguous in the submatrix. Similarly, for each subset $S' = \{b_{k+1}, \ldots, b_n\}$ of size $n - k$, $1 \leq b_{k+1} < \cdots < b_n \leq n$, we keep those columns that correspond to monomials of the form $\prod_{j=1}^{n-k} x_{k+\sigma j, b_{k+\sigma j}}$ for all permutations $\sigma \in \mathcal{S}_{n-k}$; these columns are also assumed to be contiguous. Hence each pair of subsets (S, S') defines a "block" of size $k! \times (n-k)!$ in the submatrix.

We will denote this submatrix $\widetilde{M_k}(\det_n + z)$ and also denote by $\widetilde{M_k}(\det_n)$ and $\widetilde{M_k}(z)$ the submatrices created by restricting $M_k(\det_n)$ and $M_k(z)$ to the same sets of rows and columns. Note that $\widetilde{M_k}(\det_n + z) = \widetilde{M_k}(\det_n) + \widetilde{M_k}(z)$.

We now analyze the structure of $\widetilde{M_k}(\det_n)$ and $\widetilde{M_k}(z)$ in each of the (S, S') blocks. Note that S and S' correspond naturally to two sets of variables, $\Gamma(S) = \{x_{1a_1}, \ldots, x_{ka_k}\}$ and $\Gamma(S') = \{x_{k+1, b_{k+1}}, \ldots, x_{nb_n}\}$, and that a nonzero entry in an (S, S') block corresponds to a monomial whose variables are exactly those in $\Gamma(S) \cup \Gamma(S')$. The analysis of $\widetilde{M_k}(\det_n)$ is straightforward: if S and S' are disjoint (equivalently, if their union is $[n]$), then the (S, S') block in $\widetilde{M_k}(\det_n)$ will contain a single nonzero entry (either 1 or -1 in position $(x_{1a_1} \ldots x_{ka_k}, x_{k+1b_{k+1}} \ldots x_{nb_n})$); otherwise, it is entirely zero.

The key to the rest of the proof is the following claim about the structure of $\widetilde{M_k}(z)$.

[5] In fact, Theorem 3.2 shows that Nisan's lower bound also holds for any algebra (over a field of characteristic 0) that does not satisfy any polynomial identities.

Claim 4.2. *For any $z \in T(\mathrm{Mat}_2(\mathbb{F}))$, the sum of the entries of $\widetilde{M_k}(z)$ in any (S, S') block is zero.*

Before proving the claim, we use it to complete the proof of the theorem. We can think of $\widetilde{M_k}(\det_n + z)$ as a large matrix divided into an $\binom{n}{k} \times \binom{n}{k}$ grid of (S, S') blocks. In this grid, there is a diagonal of (S, S') blocks whose entries sum to ±1, but all other (S, S') blocks have an entry sum of 0. From this it is easy to see that $\mathrm{rank}(\widetilde{M_k}(\det_n + z)) \geq \binom{n}{k}$; indeed, by a sequence of elementary row and column operations we can arrange that the top left corner entry of each block is equal to the sum of the entries in that block, which means that $\widetilde{M_k}(\det_n + z)$ contains the identity matrix of dimension $\binom{n}{k}$ as an induced submatrix. Hence its rank is at least $\binom{n}{k}$, as required.

To conclude, we supply the proof of the above claim.

Proof of Claim 4.2. Recall from section 2.1 that any $z \in T(\mathrm{Mat}_2(\mathbb{F}))$ must be a sum of identities generated by the standard identity

$$s_4(x_1, x_2, x_3, x_4) = \sum_{\sigma \in \mathcal{S}_4} \mathrm{sgn}(\sigma) \prod_{i=1}^{4} x_{\sigma i}$$

and the Hall identity in two variables

$$h(x_1, x_2) = [[x_1, x_2]^2, x_1],$$

as these form a basis for $T(\mathrm{Mat}_2(\mathbb{F}))$. In fact, we will prove a stronger version of the claim by allowing z to take on a more general form. Notice that both s_4 and h can be generated by the polynomial $t(x_1, x_2, x_3, x_4) = [x_1, x_2][x_3, x_4]$ (though t is itself not an identity); specifically, $s_4(x_1, x_2, x_3, x_4)$ is the sum of six terms of the form $t(x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4})$ with appropriate signs,[6] while $h(x_1, x_2) = t(x_1, x_2, x_1, x_2)x_1 - x_1 t(x_1, x_2, x_1, x_2)$. Hence any identity z in $T(\mathrm{Mat}_2(\mathbb{F}))$ can be written in the form

$$z = \sum_{\ell} \alpha_\ell t(g_{1\ell}, g_{2\ell}, g_{3\ell}, g_{4\ell}) \beta_\ell,$$

where $\alpha_\ell, \beta_\ell, g_{i\ell} \in \mathbb{F}\langle X \rangle$. Furthermore, since t is multilinear, we can assume without loss of generality that α_ℓ, β_ℓ, and $g_{i\ell}$ are all monomials. (Note also that, since we assume that z is homogeneous of degree n, the sum of the degrees of these monomials must be n.)

We now show that for each ℓ, the sum of the entries of $M_k(\alpha_\ell t(g_{1\ell}, g_{2\ell}, g_{3\ell}, g_{4\ell})\beta_\ell)$ is zero over any (S, S') block. We do this by showing that any nonzero entry in an (S, S') block is canonically canceled by another entry in the same block. Suppose $\alpha_\ell t(g_{1\ell}, g_{2\ell}, g_{3\ell}, g_{4\ell})\beta_\ell$ is nonzero somewhere in an (S, S') block. Then the set of variables used in α_ℓ, β_ℓ, and $g_{i\ell}$ is exactly $\Gamma(S) \cup \Gamma(S')$. Furthermore, there exists some ordering of $g_{1\ell}$ and $g_{2\ell}$, and of $g_{3\ell}$ and $g_{4\ell}$ (without loss of generality, say $g_{1\ell}, g_{2\ell}$ and $g_{3\ell}, g_{4\ell}$), such that the first k variables of the resulting monomial $\alpha_\ell g_{1\ell} g_{2\ell} g_{3\ell} g_{4\ell} \beta_\ell$ are exactly those in $\Gamma(S)$ and the last $n - k$ variables are exactly those in $\Gamma(S')$. This implies that (1) the variables used in α_ℓ, $g_{1\ell}$, and $g_{2\ell}$ are contained in $\Gamma(S)$, or (2) the variables used in $g_{3\ell}$, $g_{4\ell}$, and β_ℓ are contained in $\Gamma(S')$. ((1) holds if α_ℓ, $g_{1\ell}$, and $g_{2\ell}$ contain at most k variables in total, and (2) holds if $g_{3\ell}$, $g_{4\ell}$, and β_ℓ contain at most $n - k$ variables in total; note that both may hold.) If the former is true, then the

[6] $s_4(x_1, x_2, x_3, x_4) = t(x_1, x_2, x_3, x_4) + t(x_2, x_3, x_1, x_4) - t(x_1, x_3, x_2, x_4) - t(x_2, x_4, x_1, x_3) + t(x_1, x_4, x_2, x_3) + t(x_3, x_4, x_1, x_2)$.

reordering $g_{2\ell}g_{1\ell}g_{3\ell}g_{4\ell}$ produces the same nonzero entry with opposite sign elsewhere in the same block. Otherwise, the reordering $g_{1\ell}g_{2\ell}g_{4\ell}g_{3\ell}$ has the same effect.

This finishes the proof of the claim and the theorem. □

Remark. As with Nisan's original result for $\mathbb{F}\langle X\rangle$, our theorem also holds for all polynomials that are weakly equivalent to $\det_n$ (as defined just after Theorem 3.1), using a similar proof.

5. Other noncommutative algebras. In this section we combine our result for matrix algebras in the previous section with some additional observations to obtain similar lower bounds for several other important classes of noncommutative algebras, as claimed in Theorem 1.2 of the introduction.

5.1. Upper triangular matrices. Let $\mathrm{UMat}_d(\mathbb{F})$ denote the set of $d \times d$ upper triangular matrices over a field $\mathbb{F}$. Clearly $\mathrm{UMat}_d(\mathbb{F})$ is a subalgebra of $\mathrm{Mat}_d(\mathbb{F})$. It turns out (see [7, Theorem 5.2.1(i)]) that the single identity $t(x_1, x_2, x_3, x_4) = [x_1, x_2][x_3, x_4]$ used in the proof of Theorem 4.1 is actually a generating set for $\mathrm{UMat}_2(\mathbb{F})$. Therefore, that proof actually establishes that the ABP complexity of the determinant is exponentially large even over $\mathrm{UMat}_d(\mathbb{F})$, a stronger result.

THEOREM 5.1. *For any $d \geq 2$, the ABP complexity of computing the determinant over the algebra of $d \times d$ upper triangular matrices over any field $\mathbb{F}$ of characteristic* 0 *is given by*

$$B_{\mathrm{UMat}_d(\mathbb{F})}(\det_n) = 2^n.$$

5.2. Group algebras. Let G be a finite group. The *group algebra* of G over a field $\mathbb{F}$, denoted $\mathbb{F}G$, consists of elements of the form $\sum_{g\in G} c_g g$ for $c_g \in \mathbb{F}$, with addition defined in the natural way. Multiplication is defined according to the group operation, namely $(\sum_{g\in G} a_g g)(\sum_{h\in G} b_h h) = \sum_{k\in G}\sum_{g,h:gh=k}(a_g b_h)k$. We will now combine our result for matrix algebras in the previous section with some elementary representation theory to prove an exponential lower bound on the ABP complexity of computing the determinant over any noncommutative group algebra, whenever $\mathbb{F}$ is algebraically closed and of characteristic 0.

THEOREM 5.2. *Let G be a finite nonabelian group, and $\mathbb{F}$ an algebraically closed field of characteristic* 0*. The ABP complexity of computing the determinant over the group algebra $\mathbb{F}G$ is given by*

$$B_{\mathbb{F}G}(\det_n) = 2^n.$$

Proof. We show that if G is nonabelian, then any polynomial identity satisfied by $\mathbb{F}G$ is also satisfied by $\mathrm{Mat}_d(\mathbb{F})$ for some $d \geq 2$. A direct application of Theorem 4.1 then yields our result.

A classical fact from group representation theory (see, e.g., [22, Theorem 9]) tells us that, since G is finite and nonabelian, it must have an irreducible representation of degree at least 2; i.e., there exists a homomorphism $\rho : G \to GL_d(\mathbb{F})$ for some $d \geq 2$ such that the image of G, $\{\rho(g) : g \in G\}$, has no nontrivial invariant subspaces in $\mathbb{F}^d$.

We can now apply the following result of Burnside (see, e.g., [15]).

LEMMA 5.3 (Burnside). *Let H be a group of invertible $d \times d$ matrices over an algebraically closed field $\mathbb{F}$. Then H has no nontrivial invariant subspaces in $\mathbb{F}^d$ if and only if H contains d^2 linearly independent matrices, i.e., if and only if the $\mathbb{F}$-span of H in* $\mathrm{Mat}_d(\mathbb{F})$ *is* $\mathrm{Mat}_d(\mathbb{F})$ *itself.*

From this we see that the induced homomorphism on algebras $\hat{\rho} : \mathbb{F}G \to \mathrm{Mat}_d(\mathbb{F})$ is surjective, and therefore any polynomial identity satisfied by $\mathbb{F}G$ also must be satisfied by $\mathrm{Mat}_d(\mathbb{F})$. □

5.3. The quaternion and Clifford algebras. One of the most familiar examples of a noncommutative algebra is that of Hamilton's quaternions. This is a real algebra of dimension four, with basis elements $\{1, i, j, k\}$ and defining relations $i^2 = j^2 = k^2 = -1$ and $ij = k = -ji$. We can again apply our result on $\mathrm{Mat}_2(\mathbb{F})$ to deduce an exponential lower bound for computing the determinant of a quaternion matrix.

THEOREM 5.4. *Let* $\mathbb{H}$ *denote the quaternion algebra. Then*

$$B_{\mathbb{H}}(\det_n) = 2^n.$$

Proof. We invoke the useful fact that if $\mathbb{F}$ has characteristic 0, the polynomial identities of an $\mathbb{F}$-algebra do not change when the base field $\mathbb{F}$ is extended. This fact is folklore, but according to Drensky [8] can be traced back to [24, Lemma 2.3]. Since both $\mathbb{H}$ and $\mathrm{Mat}_2(\mathbb{R})$ become $\mathrm{Mat}_2(\mathbb{C})$ when the base field is extended from $\mathbb{R}$ to $\mathbb{C}$, we deduce that $T(\mathbb{H}) = T(\mathrm{Mat}_2(\mathbb{R}))$; i.e., the quaternion algebra satisfies exactly the same identities as the 2×2 matrices. The theorem now follows directly from Theorem 4.1. □

This theorem can be immediately extended to all higher *Clifford algebras*, as defined, e.g., in [14]. The first three Clifford algebras are $CL_1 = \mathbb{R}$, $CL_2 = \mathbb{C}$, and $CL_3 = \mathbb{H}$; the mth Clifford algebra CL_m has dimension 2^{m-1} over the reals and is isomorphic to either $\mathrm{Mat}_d(\mathbb{F})$ or $\mathrm{Mat}_d(\mathbb{F}) \oplus \mathrm{Mat}_d(\mathbb{F})$ for some d and $\mathbb{F} = \mathbb{R}$, $\mathbb{C}$, or $\mathbb{H}$. (A more operational definition is given in [5]. Note that we are abusing notation here because $\mathbb{H}$ is not a field.) We need only note that CL_m is contained in CL_{m+1} for all $m \geq 1$, and therefore $T(CL_{m+1}) \subseteq T(CL_m)$. Thus each CL_m for $m \geq 3$ inherits the lower bound for $B_{\mathbb{H}}(\det_n)$ given in Theorem 5.4. Indeed, Theorem 5.4 extends to any real noncommutative *semisimple* algebra, since by virtue of Wedderburn's structure theorem any such algebra is isomorphic to a direct sum of matrix algebras over $\mathbb{R}, \mathbb{C}$, and $\mathbb{H}$.

Our results on Clifford algebras are relevant to recent proposals that reduce approximating the permanent of an $n \times n$ 0, 1-matrix A to computing the determinant of a related matrix $\widehat{A}$, obtained by replacing the 1-entries of A by suitable random matrices. For example, Barvinok [3] has shown that if $\det_n(\widehat{A})$ can be efficiently computed when each nonzero entry of $\widehat{A}$ is chosen independently from a standard Gaussian distribution over a high-dimensional real matrix algebra, then we obtain a polynomial time approximation algorithm for the permanent of A within ratio $(1+\epsilon)^n$ for arbitrarily small $\epsilon > 0$. More recently, Chien, Rasmussen, and Sinclair [5] showed that if we can compute $\det_n(\widehat{A})$ efficiently when each nonzero entry of $\widehat{A}$ is an independent, uniformly random basis element of the Clifford algebra CL_m with $m = O(\log n)$, then we have a *fully polynomial randomized approximation scheme* for the permanent.

Our results indicate that any attempt to implement these approaches will either require a determinant algorithm that cannot be cast as an ABP, or will need to use special statistical properties of the random matrix $\widehat{A}$.

6. Discussion.

As stated in the introduction, our main goal in this work is to better understand the nature of noncommutative computation and the role played by commutativity in the design of efficient algebraic algorithms. We have used the determinant as a vehicle for these investigations. In this final section, we briefly discuss some limitations of our results and some ideas for further work.

6.1. Limitations. Our current results do not fully describe the computational power of commutativity in computing determinants. While there remain some noncommutative $\mathbb{F}$-algebras $\mathcal{A}$ over which we have yet to determine $B_{\mathcal{A}}(\det_n)$, a more important limitation derives from the ABP model. As currently defined, an ABP sees an algebra only in terms of its polynomial identities and cannot exploit any additional structure it may have. We now show two examples of this: Barvinok's symmetrized determinant and the Dieudonné determinant.

For an $n \times n$ matrix A over an $\mathbb{F}$-algebra $\mathcal{A}$, the *symmetrized determinant* [3] is defined as $\mathrm{sdet}_n(A) = \frac{1}{n!}\sum_{\sigma\in\mathcal{S}_n}\sum_{\tau\in\mathcal{S}_n}\mathrm{sgn}(\sigma)\mathrm{sgn}(\tau)\prod_{i=1}^{n} a_{\sigma i,\tau i}$. Thus sdet_n can be viewed as the average of the standard (Cayley) determinant $\det_n$ over all $n!$ possible row orderings and is in fact weakly equivalent to $\det_n$. Therefore, for (say) $\mathcal{A} = \mathrm{Mat}_2(\mathbb{R})$ (so that the entries of A are 2×2 real matrices), Theorem 4.1 implies $B_{\mathcal{A}}(\mathrm{sdet}_n) \geq 2^n$; however, Barvinok has shown that sdet_n can in fact be computed in polynomial time.[7] The key idea used is to operate on the four entries of each 2×2 matrix separately, something an ABP cannot do.

If A is a matrix over a division algebra $\mathcal{A}$ (such as the quaternion algebra), we can define the *Dieudonné determinant* Ddet_n of A (see, e.g., [2]). This can be computed in polynomial time using Gaussian elimination, which is made possible by the presence of inverses in $\mathcal{A}$. While $\mathrm{Ddet}_n(A)$ and $\det_n(A)$ are not equal in general, they are similar enough for $\mathrm{Ddet}_n(A)$ to be useful in the permanent estimators discussed in section 5.3 (see [5]). Here, it is the ability to do division that takes us outside the ABP model.

Note that neither of these examples shows that the standard determinant $\det_n$ can be efficiently computed over a specific noncommutative algebra; however, they do demonstrate the importance of considering more general models of computation.

6.2. Some open questions. Within the ABP model discussed in this paper, a number of open questions suggest themselves. First, it would be interesting to complete our picture of which noncommutative algebras, if any, allow for polynomial-sized ABPs for the determinant. While our approach thus far appears ad hoc in that we have focused on specific examples of algebras, there may not in fact be too many classes of polynomial identities that we still need to examine. Note that our analysis of the degree-4 polynomial identity $t(x_1, x_2, x_3, x_4) = [x_1, x_2][x_3, x_4]$ already covers a large spectrum of identities. We feel that an important gap here in both our understanding and our proof technique is the effect of polynomial identities of degree 3, such as the standard identity s_3, on the rank of the matrices M_k. Second, it would be interesting to extend our investigation of the role of commutativity to functions other than the determinant.

Another interesting set of issues concerns the ABP model itself. To what extent can one strengthen the model while retaining similar lower bounds for determinant computation over (say) matrix algebras? One natural extension would allow an ABP computing over a matrix algebra to access the individual components of its input matrices; as we saw above, this is helpful in computing the symmetrized determinant sdet_n (though we do not know what happens for $\det_n$ itself). Similarly, in algebras with involution (such as the Clifford algebras), we might allow an ABP to use the involutions of its input variables. More ambitiously, we might consider more general models of computation, such as those capable of implementing Gaussian elimination.

Appendix. An $O(n^3)$-size ABP for the determinant over commutative algebras. As claimed in Theorem 2.4, we sketch without proof an ABP of size $O(n^3)$ for computing the determinant of an $n \times n$ matrix $A = (a_{ij})$ over any commutative algebra $\mathcal{A}$, based on an algorithm of Mahajan and Vinay [16].

[7] More generally, if $\mathcal{A}$ is of finite dimension r, then sdet_n can be computed in time $O(n^{r+3})$ [3].

We first think of A as a weighted directed graph in which the vertices are labeled $1, \dots, n$ and each edge (i, j) has weight a_{ij}. A *closed walk* on this graph is a path $(v_1, \dots, v_k)$ (starting from v_1 and reaching v_k before returning to v_1 along edge (v_k, v_1)) in which v_1 appears only once and is the vertex with smallest label on the path (also called the *head*). A *closed walk sequence* is an ordered sequence of closed walks whose total length is n and whose heads are in strictly increasing order. The *weight* of a closed walk sequence is the product of the weights of the edges it contains.

Let $\mathcal{C}_n$ denote the set of all closed walk sequences, and for each $C \in \mathcal{C}_n$, let $w(C)$ denote the weight of C and $\text{sgn}(C) = (-1)^{n+k}$, where k is the number of closed walks in C. Note that the sum of $\text{sgn}(C)w(C)$ over only those closed walk sequences that are cycle covers of the graph is exactly the determinant of A. Moreover, Mahajan and Vinay show that in fact $\det_n(A) = \sum_{C \in \mathcal{C}_n} \text{sgn}(C)w(C)$, where the sum is over *all* closed walk sequences. The proof of this fact relies on commutativity of the matrix entries a_{ij} to ensure that the contributions of closed walk sequences that do not correspond to cycle covers cancel. (Cancellations occur between closed walk sequences in which the same edges appear but in different orders.) Mahajan and Vinay then give a simple dynamic programming algorithm for computing this sum, which can be interpreted as an ABP as follows.

The ABP has depth n. A vertex at level i, $1 \le i \le n-1$, is labeled with a triple (p, h, v) with $p \in \{0, 1\}$, $h \in \{1, \dots, n\}$, and $v \in \{1, \dots, n\}$. The function computed at such a vertex (i.e., the function computed by the ABP having this vertex as its sink) is the sum of the weights of all valid length-i prefixes of closed walk sequences in which the parity of the number of closed walks completed so far is p, the head of the walk currently being constructed is h, and the current end vertex of this walk is v. There are edges going from level i to level $i+1$ as follows. Vertex (p, h, v) at level $i < n-1$ is connected to all valid (p, h, u) at level $i+1$ by an edge with label x_{vu} (corresponding to extending the current walk to vertex u), and to all valid $(1-p, h', h')$ by an edge with label x_{vh} (corresponding to completing the current closed walk and starting a new one with head h'). Finally, vertex (p, h, v) at level $n-1$ is connected to the sink (the only vertex at level n) by an edge with label $(-1)^{n+p+1}x_{vh}$ (corresponding to completing the last closed walk and incorporating the sign of the closed walk sequence). This ABP has size $O(n^3)$ ($O(n^2)$ vertices at each of $O(n)$ levels), and it is straightforward to check that it computes the above sum over all closed walk sequences.

Acknowledgments. We thank Vesselin Drensky for valuable help with the literature on polynomial identities, particularly for the proof of Theorem 5.4, and, as always, T. Y. Lam for patiently answering all our algebraic questions. We also thank Salil Vadhan and Avi Wigderson for helpful initial discussions.

REFERENCES

[1] S. A. Amitsur and J. Levitzki, *Minimal identities for algebras*, Proc. Amer. Math. Soc., 2 (1950), pp. 449–463.

[2] E. Artin, *Geometric Algebra*, Wiley-Interscience, New York, 1988.

[3] A. Barvinok, *New Permanent Estimators via Noncommutative Determinants*, preprint, University of Michigan, Ann Arbor, MI, 2000; available online from www.math.lsa.umich.edu/∼barvinok/papers.html.

[4] F. Benanti, J. Demmel, V. Drensky, and P. Koev, *Computational approach to polynomial identities of matrices—a survey*, in Ring Theory: Polynomial Identities and Combinatorial Methods, Lecture Notes in Pure and Appl. Math. 235, A. Giambruno, A. Regev, and M. Zaicev, eds., Dekker, New York, 2003, pp. 141–178.

[5] S. Chien, L. Rasmussen, and A. Sinclair, *Clifford algebras and approximating the permanent*, J. Comput. System Sci., 67 (2003), pp. 263–290.

[6] V. Drensky, *A minimal basis for the identities for a second-order matrix algebra over a field of characteristic* 0, Algebra Logic, 20 (1981), pp. 188–194.

[7] V. Drensky, *Free Algebras and PI-Algebras*, Springer-Verlag, Singapore, 1999.

[8] V. Drensky, *private communication*, 2004.

[9] D. Fadeev and V. Fadeeva, *Computational Methods in Linear Algebra*, Freeman, San Francisco, 1963.

[10] D. Grigoriev and M. Karpinski, *An exponential lower bound for depth* 3 *arithmetic circuits*, in Proceedings of the 30th Annual ACM Symposium on Theory of Computing, 1998, pp. 577–582.

[11] D. Grigoriev and A. Razborov, *Exponential lower bounds for depth* 3 *arithmetic circuits in algebras of functions over finite fields*, Appl. Algebra Engrg. Comm. Comput., 10 (2000), pp. 465–487.

[12] K. Kalorkoti, *A lower bound for the formula size of rational functions*, SIAM J. Comput., 14 (1985), pp. 678–687.

[13] A. Kemer, *Ideals of Identities of Associative Algebras*, Transl. Math. Monogr. 87, AMS, Providence, RI, 1991.

[14] T. Y. Lam, *Introduction to Quadratic Forms over Fields*, Grad. Stud. Math. 67, AMS, Providence, RI, 2004.

[15] T. Y. Lam, *A theorem of Burnside on matrix rings*, Amer. Math. Monthly, 105 (1998), pp. 651–653.

[16] M. Mahajan and V. Vinay, *Determinant: Combinatorics, algorithms, and complexity*, Chic. J. Theoret. Comput. Sci., 1997, Article 5.

[17] M. Mahajan and V. Vinay, *Determinant: Old algorithms, new insights*, SIAM J. Discrete Math., 12 (1999), pp. 474–490.

[18] N. Nisan, *Lower bounds for noncommutative computation*, in Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, 1991, pp. 410–418.

[19] N. Nisan and A. Wigderson *Lower bounds on arithmetic circuits via partial derivatives*, Comput. Complexity, 6 (1996), pp. 217–234.

[20] R. Raz, *Multi-linear formulas for permanent and determinant are of super-polynomial size*, in Proceedings of the 36th Annual ACM Symposium on Theory of Computing, 2004, pp. 633–641.

[21] R. Raz and A. Shpilka, *Deterministic polynomial identity testing in noncommutative models*, Comput. Complexity, 14 (2005), pp. 1–19.

[22] J.-P. Serre, *Linear Representations of Finite Groups*, Springer-Verlag, New York, 1977.

[23] L. Valiant, *Why is Boolean complexity theory difficult?*, in Boolean Function Complexity, London Math. Soc. Lecture Note Ser. 169, M. Paterson, ed., Cambridge University Press, Cambridge, UK, 1992, pp. 84–94.

[24] M. R. Vaughan-Lee, *Varieties of Lie algebras*, Quart. J. Math. Oxford Ser. (2), 21 (1970), pp. 297–308.

SIAM J. Comput.
Vol. 37, No. 1, pp. 267–302

WORST-CASE TO AVERAGE-CASE REDUCTIONS BASED ON GAUSSIAN MEASURES*

DANIELE MICCIANCIO† AND ODED REGEV‡

Abstract. We show that finding small solutions to random modular linear equations is at least as hard as approximating several lattice problems in the worst case within a factor almost linear in the dimension of the lattice. The lattice problems we consider are the shortest vector problem, the shortest independent vectors problem, the covering radius problem, and the guaranteed distance decoding problem (a variant of the well-known closest vector problem). The approximation factor we obtain is $n \log^{O(1)} n$ for all four problems. This greatly improves on all previous work on the subject starting from Ajtai's seminal paper [*Generating hard instances of lattice problems*, in Complexity of Computations and Proofs, Quad. Mat. 13, Dept. Math., Seconda Univ. Napoli, Caserta, Italy, 2004, pp. 1–32] up to the strongest previously known results by Micciancio [*SIAM J. Comput.*, 34 (2004), pp. 118–169]. Our results also bring us closer to the limit where the problems are no longer known to be in NP intersect coNP. Our main tools are Gaussian measures on lattices and the high-dimensional Fourier transform. We start by defining a new lattice parameter which determines the amount of Gaussian noise that one has to add to a lattice in order to get close to a uniform distribution. In addition to yielding quantitatively much stronger results, the use of this parameter allows us to simplify many of the complications in previous work. Our technical contributions are twofold. First, we show tight connections between this new parameter and existing lattice parameters. One such important connection is between this parameter and the length of the shortest set of linearly independent vectors. Second, we prove that the distribution that one obtains after adding Gaussian noise to the lattice has the following interesting property: the distribution of the noise vector when conditioning on the final value behaves in many respects like the original Gaussian noise vector. In particular, its moments remain essentially unchanged.

Key words. lattices, worst-case to average-case reductions, Gaussian measures

AMS subject classifications. 68Q25, 11H06

DOI. 10.1137/S0097539705447360

1. Introduction.

Lattice problems have received considerable attention as a potential source of computational hardness to be used in cryptography, after a breakthrough result of Ajtai [2] showing that if certain lattice problems are computationally hard to solve in the *worst case*, then *average-case* one-way functions (a fundamental cryptographic primitive) exist. Ajtai's one-way function is essentially the generalized subset sum function over the additive group of n-dimensional vectors modulo q: functions are described by m group elements $\mathbf{a}_1, \ldots, \mathbf{a}_m \in \mathbb{Z}_q^n$, and the associated function maps bit-string $x_1, \ldots, x_m \in \{0,1\}$ to $f_{\mathbf{A}}(x_1, \ldots, x_m) = \sum_i \mathbf{a}_i x_i$. The main worst-case lattice problem considered by Ajtai is that of finding a set of n linearly

[*] Received by the editors March 26, 2005; accepted for publication (in revised form) May 28, 2006; published electronically May 14, 2007. A preliminary version of this paper appeared in the *Proceedings of the* 45*th Annual IEEE Symposium on Foundations of Computer Science*, Rome, Italy, 2004, pp. 372–381. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

http://www.siam.org/journals/sicomp/37-1/44736.html

[†] Computer Science and Engineering Department, University of California-San Diego, La Jolla, CA 92093 (daniele@cs.ucsd.edu). This author's work was supported in part by NSF grants CCR-0093029 and CCF-0634909, and a Sloan Research Fellowship.

[‡] Department of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel. This author's work was supported by the Israel Science Foundation, by the Binational Science Foundation, by an Alon Fellowship, by the Army Research Office grant DAAD19-03-1-0082, and by NSF grant CCR-9987845.

independent lattice vectors in an arbitrary lattice of length within a polynomial (in n) factor from the shortest such set.[1] This problem in turn is related, using standard techniques, to various other lattice problems, such as approximating the length of the shortest nonzero lattice vector in the worst case, within factors polynomial in n.

No polynomial time algorithm is known to solve any of these worst-case problems, so it is reasonable to conjecture that the problems are hard for any polynomial approximation factor. Still, since the problems get easier and easier as the factor increases, it is theoretically interesting and practically important to determine the smallest factors for which the hardness of approximating these lattice problems in the worst case implies that the function $f_{\mathbf{A}}$ is one-way on the average. The factors implicit in Ajtai's proof are rather large: [9] estimates all these factors to be larger than n^8. In subsequent developments the factors have been improved, leading to the currently best known results of Micciancio [21]: the subset-sum function $f_{\mathbf{A}}$ is hard to invert (in fact, even collision resistant) on the average, provided that any of the following problems is hard in the worst case:

- computing a set of n linearly independent lattice vectors in an n-dimensional lattice of length within a factor[2] of $\tilde{O}(n^{2.5})$ from the shortest such set;
- approximating the length of the shortest nonzero vector in an n-dimensional lattice within a factor of $\tilde{O}(n^3)$;
- approximating the covering radius of an n-dimensional lattice within a factor of $\tilde{O}(n^{2.5})$;
- finding a lattice vector within distance at most $\tilde{O}(n^{2.5})$ times the covering radius from any given target point.

Micciancio [21] also showed that the above factors can be further reduced by $\sqrt{n}$ if certain sequences of "almost perfect" easily decodable lattices exist, and conjectured a reduction achieving factors as low as $\tilde{O}(n^{1.5})$. In a recent work of Regev [24], a similar result was shown based on worst-case instances of a problem known as the $\tilde{O}(n^{1.5})$-*unique* shortest vector problem. This problem is a special case of the shortest vector problem in which the lattices have a special structure (namely, their shortest vector is *unique*, in the sense that the next shortest linearly independent vector is longer than the shortest nonzero vector by $\tilde{O}(n^{1.5})$). Although the connection factor $\tilde{O}(n^{1.5})$ is better than the factors of [21], a major drawback of the reduction in [24] is that the unique shortest vector problem is potentially easier to solve than the shortest vector problem; in fact, it is not even known to be NP-hard for small constant approximation factors.[3]

Our results. We substantially improve all of the above results and prove that the subset-sum function $f_{\mathbf{A}}$ is hard to invert (and collision resistant) on the average provided any of the following problems is hard in the worst case:

- computing a set of n linearly independent lattice vectors in an n-dimensional lattice of length within a factor of $\tilde{O}(n)$ from the shortest such set;
- approximating the length of the shortest nonzero vector in an n-dimensional lattice within a factor of $\tilde{O}(n)$;

[1] The length of a finite set of vectors is defined as the length of the longest vector in the set. The problem can be defined with respect to any norm, but the Euclidean norm is the most common.

[2] A function $g(n)$ is in $\tilde{O}(f(n))$ if there exist constants $a, c \geq 0$ such that $g(n) \leq af(n) \log^c f(n)$ for all sufficiently large n.

[3] The main result of [24] is a lattice-based *encryption scheme*. This encryption scheme, as is the one in the original work of Ajtai and Dwork [3], is also based on the unique shortest vector problem. Constructing an *encryption scheme* based on other lattice problems such as the shortest vector problem is a major open problem.

- approximating the covering radius of an n-dimensional lattice within a factor of $\tilde{O}(n)$;
- finding a lattice vector within distance at most $\tilde{O}(n)$ times the covering radius from any given target point.

In other words, the connection factor is $\tilde{O}(n)$ for all four lattice problems. This proves Micciancio's conjecture [21] and in fact provides even better connection factors. Our results are significant for two reasons:

- On the technical side, we present a new approach to worst-case to average-case reductions for lattice problems, based on the use of Gaussian measures. The results in [21] were making an essentially optimal use of previous reduction techniques; the results presented in this paper require some new techniques that might be of independent interest. Another important technical contribution of this paper is the study of Gaussian distributions on lattices. These issues are discussed in subsection 1.1.
- On the theoretical side, our improvements bring us closer to factors for which lattice problems are not known to be in NP $\cap$ coNP. This is discussed in subsection 1.2.

1.1. Our techniques.

The reduction. In this paper, as in previous work [2, 12, 9, 21, 24], we consider the problem of reducing worst-case instances of lattice approximation problems (e.g., finding short lattice vectors) to the problem of finding small solutions to random linear equations with coefficients in $\mathbb{Z}_q^n$. So, in order to perform such a reduction, one needs to sample (almost uniformly at random) the group $\mathbb{Z}_q^n$ in a way that is somehow related to an underlying lattice problem (for an arbitrary lattice) as we now explain. The core of the reduction is a (polynomial time) sampling procedure that allows us to draw pairs consisting of a group element and a corresponding short "offset" vector (not necessarily in the lattice) having the following property: any (integer) solution to the homogeneous linear equation defined by the group elements maps the corresponding short offset vectors to a vector in the underlying lattice. The length of the resulting lattice vector depends on the size of the integer solution used to combine the short offset vectors. If we can find a *small* solution to the group equation (e.g., using the average-case oracle), then we can find a *short* lattice vector, essentially solving the underlying lattice problem. We remark that for the average-case oracle to work, the coefficients of the equation must be distributed almost uniformly at random in the group.

The above high level approach to worst-case to average-case reduction is common to all works, including this paper. The difference is in the way group elements (and corresponding short offset vectors) are sampled. Essentially all previous works were based on the following approach: given an arbitrary lattice $\mathcal{L}(\mathbf{B})$, consider a sufficiently large region of space C which is approximately equal to a hypercube of size ℓ (with vertices in $\mathcal{L}(\mathbf{B})$). Then divide each side into q equal parts. This results in q^n subcubes of size ℓ/q, each corresponding to a group element in $\mathbb{Z}_q^n$. Next we sample lattice points from $\mathcal{L}(\mathbf{B}) \cap C$ and for each sample consider the corresponding subcube and offset within the subcube (e.g., with respect to the center of the subcube). If each subcube contains approximately the same number of lattice points, then the induced distribution on group elements is almost uniform over $\mathbb{Z}_q^n$. The correctness of the reduction is based on the following two important properties of the sampling procedure:

- Each subregion should be small enough so that the offset vectors are short,

and the final output of the reduction is a short lattice vector.
- Each subregion should be large enough so that the number of lattice points in each region is about the same and the chosen group element is almost uniform in $\mathbb{Z}_q^n$.

These two contradicting requirements end up determining the connection factor obtained by the reduction.

In this paper we develop a new technique to generate random group elements that does not require starting from a large hypercube C. Instead of considering large regions of space and counting the number of lattice points in them, we simply start from a lattice point and add some Gaussian noise to it. Our goal is to use an amount of noise sufficiently large so that the resulting point (which does not belong to the lattice in general) is distributed almost uniformly in space.

Technically, we pick a random noise vector with a Gaussian distribution, and reduce it modulo the basis of the lattice, to obtain a vector distributed almost uniformly at random over the fundamental parallelepiped of the lattice. Next we divide the fundamental parallelepiped into q^n equal regions and use each of them to represent a group element in $\mathbb{Z}_q^n$. Notice that none of the regions contains any lattice point. Notice also that using this approach, it is not important that the regions have a nice (approximately hypercubic) shape: since all regions have the same volume, a reduced noise vector distibuted almost uniformly over the fundamental parallelepiped will induce an almost uniform distribution over $\mathbb{Z}_q^n$.

As an additional remark, we point out that the previous best reductions produced group elements whose distribution is only moderately close to uniform. In order to get almost uniformly distributed group elements, they generated a small (superlogarithmic) number of group elements and added them all up. Our technique avoids this complication since it directly gives group elements whose distribution is extremely close to uniform, and does not require adding up many samples. We believe that this fact, together with the fact that we do not need to start from a large cube, allows us to obtain a much cleaner and simpler reduction. The ideas and techniques presented in this paper have been recently used in [22] to obtain analogous improvements and simplifications for similar results about cyclic lattices.

Gaussian distributions. The use of Gaussian distributions in the study of lattices is standard in mathematics (see, for example, [5]). In computer science, they have recently been used in [8, 24, 1]. In [1], for example, Gaussian distributions are used to prove that certain lattice problems are in coNP.

We believe that a large part of our technical contribution is in the study of these Gaussian distributions. We start by defining the *smoothing parameter* of a lattice, a new lattice parameter with the following fundamental property:[4] if one picks a noise vector from a Gaussian distribution with radius at least as large as the smoothing parameter and reduces the noise vector modulo the fundamental parallelepiped of the lattice, then the resulting distribution is very close to uniform. We then relate this parameter to standard lattice parameters such as the length of the shortest dual vector and the length of the shortest set of independent vectors. The proof of the former is based on a lemma by Banaszczyk [5], while the proof of the latter is, to the best of our knowledge, novel.

We then go on to consider the discrete Gaussian distribution on a lattice. Let

[4] The actual definition of smoothing parameter involves the dual lattice and is rather technical. Here we state only a fundamental property of the smoothing parameter that conveys the intuition behind our definition. See Definition 3.1 for the actual definition.

$\mathbf{c}$ be any point in space. Let $\mathbf{y}$ be obtained by adding to $\mathbf{c}$ a vector chosen from a Gaussian distribution whose size is at least the smoothing parameter of the lattice. Then, consider the distribution of $\mathbf{y}$ conditioned on it being in the lattice (this will be made rigorous later). This distribution is illustrated in Figure 1. Essentially, it is a Gaussian distribution around $\mathbf{c}$ restricted to the lattice. Interestingly, we prove that this distribution behaves in many respects like the (continuous) Gaussian distribution around $\mathbf{c}$. For example, its center is very close to $\mathbf{c}$, and its average square distance from $\mathbf{c}$ is also very close to that of the continuous Gaussian distribution. From these two facts we can derive relatively easily all the properties needed for the worst-case to average-case reduction.

1.2. Complexity of lattice problems. Since many lattice problems are NP-hard to approximate within small factors, connections between the average-case and worst-case complexity of lattice problems can be regarded as progress toward the ambitious goal of constructing one-way functions based on the assumption that $\mathrm{P} \neq \mathrm{NP}$. Unfortunately, there is still a big gap between factors for which lattice problems are known to be NP-hard and those known to imply the existence of one-way functions. The strongest known hardness result (for the problems considered in this paper) is the NP-hardness of approximating the length of the shortest linearly independent set within any constant and, under the stronger assumption $\mathrm{NP} \not\subseteq \mathrm{DTIME}(2^{\mathrm{polylog}(n)})$, within $2^{(\log n)^{1-\epsilon}}$ for any $\epsilon > 0$ [6]. For the shortest vector problem, hardness within any constant approximation factor or factors of the form $2^{(\log n)^{1/2-\epsilon}}$ (for any $\epsilon > 0$) has been shown [16] under the assumption[5] that $\mathrm{NP} \neq \mathrm{RP}$ or $\mathrm{NP} \not\subseteq \mathrm{BPTIME}(2^{\mathrm{polylog}(n)})$, respectively. No hardness result (under deterministic or probabilistic reductions) is currently known for the covering radius problem, although the problem is conceivably hard. (See [14] for further discussion of the complexity of the covering radius problem.)

Beside the fact that all known hardness results are only for subpolynomial approximation factors, all three problems have been shown to be in coAM for $O(\sqrt{n/\log n})$ approximation factors [11, 14] (see also [1, 14] where the problems are shown to be in coNP for $O(\sqrt{n})$ factors), giving evidence[6] that the problems are not NP-hard within such factors. Still, one might conjecture that some of these problems are NP-hard to approximate for factors close to $\sqrt{n/\log n}$, say, $n^{1/2-\epsilon}$ for any $\epsilon > 0$.

The results in this paper, showing that there exist hard-on-average problems based on the inapproximability of lattice problems within $\tilde{O}(n)$, bring us closer to factors of $O(\sqrt{n})$, below which the lattice problems are not known to be in coNP, and therefore may be NP-hard. However, it is not clear how our techniques can be used to obtain factors below $\tilde{O}(n)$.

2. Preliminaries.

General. For any real x, $\lfloor x \rfloor$ denotes the largest integer not greater than x. For a vector $\mathbf{x} = (x_1, \ldots, x_n)$ we define $\lfloor \mathbf{x} \rfloor$ as $(\lfloor x_1 \rfloor, \ldots, \lfloor x_n \rfloor)$. We write log for the

[5] No true NP-hardness result (i.e., under deterministic polynomial time reductions) is currently known for SVP even in its exact version. However, [19] showed that if a certain number theoretic conjecture on the distribution of square-free smooth numbers holds true, then SVP is NP-hard (under deterministic polynomial time Karp reductions) for any factor $\gamma < \sqrt{2}$.

[6] Specifically, since the first two problems are in NP even in their exact version, they cannot be NP-hard to approximate within $O(\sqrt{n/\log n})$ (resp. $O(\sqrt{n})$) unless $\mathrm{NP} \subseteq \mathrm{coAM}$ (resp. $\mathrm{NP} = \mathrm{coNP}$.) For the covering radius problem the situation is more complicated because the exact version of the problem is not known to be in NP. See [11, 1] for further discussion of the implications of these results.

logarithm to the base 2, and $\log_q$ when the base q is any number possibly different from 2. We use $\omega(f(n))$ to denote the set of functions growing faster than $c \cdot f(n)$ for any $c > 0$. A function $\epsilon(n)$ is *negligible* if $\epsilon(n) < 1/n^c$ for any $c > 0$ and all sufficiently large n.

The n-dimensional Euclidean space is denoted $\mathbb{R}^n$. We use bold lower case letters (e.g., $\mathbf{x}$) to denote vectors, and bold upper case letters (e.g., $\mathbf{M}$) to denote matrices. The ith coordinate of $\mathbf{x}$ is denoted x_i. For a set $S \subseteq \mathbb{R}^n$, $\mathbf{x} \in \mathbb{R}^n$, and $a \in \mathbb{R}$, we let $S + \mathbf{x} = \{\mathbf{y} + \mathbf{x} \colon \mathbf{y} \in S\}$ denote the translate of S by $\mathbf{x}$, and $aS = \{a\mathbf{y} \colon \mathbf{y} \in S\}$ denote the scaling of S by a. The Euclidean norm (also known as the ℓ_2 norm) of a vector $\mathbf{x} \in \mathbb{R}^n$ is $\|\mathbf{x}\| = (\sum_i x_i^2)^{1/2}$, and the associated distance is $\operatorname{dist}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$. The distance function is extended to sets in the customary way: $\operatorname{dist}(\mathbf{x}, S) = \operatorname{dist}(S, \mathbf{x}) = \min_{\mathbf{y} \in S} \operatorname{dist}(\mathbf{x}, \mathbf{y})$. We often use matrix notation to denote sets of vectors. For example, matrix $\mathbf{S} \in \mathbb{R}^{n \times m}$ represents the set of n-dimensional vectors $\{\mathbf{s}_1, \ldots, \mathbf{s}_m\}$, where $\mathbf{s}_1, \ldots, \mathbf{s}_m$ are the columns of $\mathbf{S}$. We denote by $\|\mathbf{S}\|$ the maximum length of a vector in $\mathbf{S}$. The linear space spanned by a set of m vectors $\mathbf{S}$ is denoted $\operatorname{span}(\mathbf{S}) = \{\sum_i x_i \mathbf{s}_i \,:\, x_i \in \mathbb{R} \text{ for } 1 \le i \le m\}$. For any set of n linearly independent vectors $\mathbf{S}$, we define the half-open parallelepiped $\mathcal{P}(\mathbf{S}) = \{\sum_i x_i \mathbf{s}_i : 0 \le x_i < 1 \text{ for } 1 \le i \le n\}$. Finally, we denote by $\mathcal{B}$ the closed Euclidean ball of radius 1 around the origin, $\mathcal{B} = \{\mathbf{w} \in \mathbb{R}^n : \|\mathbf{w}\| \le 1\}$.

Statistical distance. Statistical distance is a measure of distance between two probability distributions and is a convenient tool in the analysis of randomized algorithms and reductions. Here we define it and state some simple facts that will be used in the rest of the paper. These facts are easily verified; for more details the reader is referred to [23, Chapter 8].

Definition 2.1. *We define the statistical distance between two discrete random variables X and Y over a (countable) set A as*

$$\Delta(X, Y) = \frac{1}{2} \sum_{a \in A} |\Pr\{X = a\} - \Pr\{Y = a\}|.$$

Similarly, for two continuous random variables X and Y over $\mathbb{R}^n$ with probability density functions T_1 and T_2, respectively, the statistical distance is defined as

$$\Delta(X, Y) = \frac{1}{2} \int_{\mathbb{R}^n} |T_1(r) - T_2(r)| dr.$$

One important fact that we use is that the statistical distance cannot increase by applying a (possibly randomized) function f, i.e.,

$$\Delta(f(X), f(Y)) \le \Delta(X, Y); \tag{1}$$

see, e.g., [23]. In particular, this implies that the acceptance probability of any algorithm on inputs from X differs from its acceptance probability on inputs from Y by at most $\Delta(X, Y)$. Another useful property of the statistical distance is the following. Let $X_1, \ldots, X_k$ and $Y_1, \ldots, Y_k$ be two lists of totally independent random variables. Then

$$\Delta((X_1, \ldots, X_k), (Y_1, \ldots, Y_k)) \le \sum_{i=1}^{k} \Delta(X_i, Y_i).$$

Lattices. We now describe some basic definitions related to lattices. For a more in-depth discussion, see [23]. An n-dimensional *lattice* is the set of all integer combinations

$$\left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i \colon x_i \in \mathbb{Z} \text{ for } 1 \leq i \leq n \right\}$$

of n linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$ in $\mathbb{R}^n$.[7] The set of vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$ is called a *basis* for the lattice. A basis can be represented by the matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n] \in \mathbb{R}^{n \times n}$ having the basis vectors as columns. The lattice generated by $\mathbf{B}$ is denoted $\mathcal{L}(\mathbf{B})$. Notice that $\mathcal{L}(\mathbf{B}) = \{\mathbf{B}\mathbf{x} \colon \mathbf{x} \in \mathbb{Z}^n\}$, where $\mathbf{B}\mathbf{x}$ is the usual matrix-vector multiplication.

For any lattice basis $\mathbf{B}$ and point $\mathbf{x}$, there exists a unique vector $\mathbf{y} \in \mathcal{P}(\mathbf{B})$ such that $\mathbf{y} - \mathbf{x} \in \mathcal{L}(\mathbf{B})$. This vector is denoted $\mathbf{y} = \mathbf{x} \bmod \mathbf{B}$, and it can be computed in polynomial time given $\mathbf{B}$ and $\mathbf{x}$. The dual of a lattice Λ is the set

$$\Lambda^* = \{\mathbf{x} \colon \forall \mathbf{y} \in \Lambda \ \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}\}$$

of all vectors that have integer scalar product ($\langle \mathbf{x}, \mathbf{y} \rangle = \sum_i x_i y_i$) with all lattice vectors. The dual of a lattice is a lattice, and if $\Lambda = \mathcal{L}(\mathbf{B})$ is the lattice generated by basis $\mathbf{B}$, then $\mathbf{B}^* = (\mathbf{B}^T)^{-1}$ is a basis for the dual lattice, where $\mathbf{B}^T$ is the transpose of $\mathbf{B}$. A sublattice of $\mathcal{L}(\mathbf{B})$ is a lattice $\mathcal{L}(\mathbf{S})$ such that $\mathcal{L}(\mathbf{S}) \subseteq \mathcal{L}(\mathbf{B})$. The *determinant* of a lattice $\det(\mathcal{L}(\mathbf{B}))$ is the (n-dimensional) volume of the fundamental parallelepiped $\mathcal{P}(\mathbf{B})$ and is given by $|\det(\mathbf{B})|$.

The *minimum distance* of a lattice Λ, denoted $\lambda_1(\Lambda)$, is the minimum distance between any two distinct lattice points, and equals the length of the shortest nonzero lattice vector:

$$\begin{aligned} \lambda_1(\Lambda) &= \min\{\operatorname{dist}(\mathbf{x}, \mathbf{y}) : \mathbf{x} \neq \mathbf{y} \in \Lambda\} \\ &= \min\{\|\mathbf{x}\| : \mathbf{x} \in \Lambda \setminus \{\mathbf{0}\}\} \ . \end{aligned}$$

This definition can be generalized to define the ith successive minimum as the smallest λ_i such that $\lambda_i \mathcal{B}$ contains i linearly independent lattice points:

$$\lambda_i(\Lambda) = \min\{r \colon \dim(\operatorname{span}(\Lambda \cap r\mathcal{B})) \geq i\}.$$

Another important constant associated to a lattice is the *covering radius* $\nu(\Lambda)$, defined as

$$\nu(\Lambda) = \max_{\mathbf{x} \in \mathbb{R}^n} \{\operatorname{dist}(\mathbf{x}, \Lambda)\}.$$

We often abuse notation and write $\lambda_1(\mathbf{B})$ instead of $\lambda_1(\mathcal{L}(\mathbf{B}))$ and similarly for other lattice parameters.

Lattice problems. We consider the following lattice problems. For simplicity, we consider some of our problems in their promise version.[8] It is easy to see that a

[7] Strictly speaking, this is the definition of a *full-rank* lattice. Since only full-rank lattices are used in this paper, all definitions are restricted to the full-rank case.

[8] Promise problems are a generalization of decision problems where one is asked whether a given input satisfies one of two mutually exclusive properties. Unlike decision problems, these two properties are not necessarily exhaustive. The problem is, under the promise that the given input satisfies one of the two conditions, to tell which of the two properties is satisfied. If the input satisfies neither property, then any answer is acceptable.

solution to any of the promise problems below implies a solution to the corresponding optimization problem (that is, the problem that asks for an approximation to the corresponding lattice parameter, e.g., λ_1). The reader is referred to [23] for further discussion of these lattice problems. The following definitions are parameterized by a positive (and typically monotone) real valued function $\gamma\colon \mathbb{Z}^+ \to \mathbb{R}^+$ of the lattice dimension.

Definition 2.2 (shortest vector problem). *An input to* GapSVP$_\gamma$ *is a pair* $(\mathbf{B}, d)$*, where* $\mathbf{B}$ *is an* n*-dimensional lattice basis and* d *is a rational number. In* Yes *inputs* $\lambda_1(\mathbf{B}) \leq d$ *and in* No *inputs* $\lambda_1(\mathbf{B}) > \gamma(n) \cdot d$*.*

Definition 2.3 (closest vector problem). *An input to* GapCVP$_\gamma$ *is a triple* $(\mathbf{B}, \mathbf{t}, d)$*, where* $\mathbf{B}$ *is an* n*-dimensional lattice basis,* $\mathbf{t}$ *is a target vector, and* d *is a rational number. In* Yes *inputs* $\operatorname{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B})) \leq d$ *and in* No *inputs* $\operatorname{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B})) > \gamma(n) \cdot d$*.*

Definition 2.4 (covering radius problem). *An input to* GapCRP$_\gamma$ *is a pair* $(\mathbf{B}, d)$*, where* $\mathbf{B}$ *is an* n*-dimensional lattice basis and* d *is a rational number. In* Yes *inputs* $\nu(\mathbf{B}) \leq d$ *and in* No *inputs* $\nu(\mathbf{B}) > \gamma(n) \cdot d$*.*

The remaining lattice problems are given in their search version.

Definition 2.5 (shortest independent vectors problem). *An input to* SIVP$_\gamma$ *is an* n*-dimensional lattice basis* $\mathbf{B}$*. The goal is to output a set of* n *linearly independent lattice vectors* $\mathbf{S} \subset \mathcal{L}(\mathbf{B})$ *such that* $\|\mathbf{S}\| \leq \gamma(n) \cdot \lambda_n(\mathbf{B})$*, where* $\|\mathbf{S}\|$ *is the maximum length of a vector in* $\mathbf{S}$*.*

A generalization of SIVP is the following somewhat less standard lattice problem.

Definition 2.6 (generalized independent vectors problem). *An input to* GIVP$_\gamma^\phi$ *is an* n*-dimensional lattice basis* $\mathbf{B}$*. The goal is to output a set of* n *linearly independent lattice vectors* $\mathbf{S} \subset \mathcal{L}(\mathbf{B})$ *such that* $\|\mathbf{S}\| \leq \gamma(n) \cdot \phi(\mathbf{B})$*.*

In the above, ϕ denotes any arbitrary function on lattices. Choosing $\phi = \lambda_n$ results in the SIVP. In this paper, we usually take ϕ to be the smoothing parameter, defined in the next section.

Definition 2.7 (guaranteed distance decoding). *An input to* GDD$_\gamma^\phi$ *is an* n*-dimensional lattice basis* $\mathbf{B}$ *and a target point* $\mathbf{t}$*. The goal is to output a lattice point* $\mathbf{x} \in \mathcal{L}(\mathbf{B})$ *such that* $\operatorname{dist}(\mathbf{t}, \mathbf{x}) \leq \gamma(n) \cdot \phi(\mathbf{B})$*.*

In this problem, we usually take $\phi = \nu$ to be the covering radius of the lattice. Notice that for any lattice basis $\mathbf{B}$ and target $\mathbf{t} \in \mathbb{R}^n$, there is always a lattice point within distance $\nu(\mathbf{B})$ of $\mathbf{t}$. The GDD$_\gamma^\nu$ problem can be seen as a variant of the CVP in which the quality of the solution is measured with respect to the worst possible distance $\max_{\mathbf{x} \in \mathbb{R}^n} \operatorname{dist}(\mathbf{x}, \mathcal{L}(\mathbf{B}))$ instead of the distance of the given target $\operatorname{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B}))$.

Gaussian measures. For any vectors $\mathbf{c}, \mathbf{x}$ and any $s > 0$, let

$$\rho_{s,\mathbf{c}}(\mathbf{x}) = e^{-\pi\|(\mathbf{x}-\mathbf{c})/s\|^2}$$

be a Gaussian function centered in $\mathbf{c}$ scaled by a factor of s. The total measure associated to $\rho_{s,\mathbf{c}}$ is $\int_{\mathbf{x}\in\mathbb{R}^n} \rho_{s,\mathbf{c}}(\mathbf{x})d\mathbf{x} = s^n$. Therefore, we can define the (continuous) Gaussian distribution around $\mathbf{c}$ with parameter s by its probability density function

$$\forall \mathbf{x} \in \mathbb{R}^n, \; D_{s,\mathbf{c}}(\mathbf{x}) = \frac{\rho_{s,\mathbf{c}}(\mathbf{x})}{s^n}.$$

It can be seen that the expected square distance from $\mathbf{c}$ of a vector chosen from this distribution is $ns^2/(2\pi)$. So, intuitively, one can think of $D_{s,\mathbf{c}}$ as a sphere of radius $s\sqrt{n/(2\pi)}$ centered around $\mathbf{c}$.

Notice that $D_{s,\mathbf{c}}$ can be expressed as the sum of n orthogonal 1-dimensional Gaussian distributions, and each of them can be efficiently approximated with arbitrary precision using standard techniques. Thus, the distribution $D_{s,\mathbf{c}}$ can be efficiently approximated. For simplicity, in this paper we work with real numbers and assume we can sample from $D_{s,\mathbf{c}}$ exactly. In practice, when only finite precision is available, $D_{s,\mathbf{c}}$ can be approximated by picking a fine grid and choosing points from the grid with probability approximately proportional to $D_{s,\mathbf{c}}$. All our arguments can be made rigorous by selecting a sufficiently fine grid.

When $\mathbf{c}$ and s are not specified, we assume that they are the origin and 1, respectively. Functions are extended to sets in the usual way; e.g., $\rho_{s,\mathbf{c}}(A) = \sum_{\mathbf{x}\in A} \rho_{s,\mathbf{c}}(\mathbf{x})$ for any countable set A.

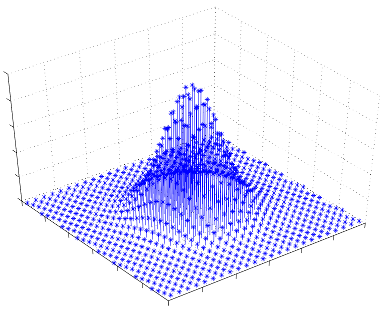


FIG. 1. *A discrete Gaussian distribution.*

For any vector $\mathbf{c}$, real $s > 0$, and lattice Λ, define the probability distribution $D_{\Lambda,s,\mathbf{c}}$ over Λ by

$$\forall \mathbf{x} \in \Lambda,\ D_{\Lambda,s,\mathbf{c}}(\mathbf{x}) = \frac{D_{s,\mathbf{c}}(\mathbf{x})}{D_{s,\mathbf{c}}(\Lambda)} = \frac{\rho_{s,\mathbf{c}}(\mathbf{x})}{\rho_{s,\mathbf{c}}(\Lambda)}.$$

We refer to $D_{\Lambda,s,\mathbf{c}}$ as a discrete Gaussian distribution (see Figure 1) and as before, we sometimes omit s or $\mathbf{c}$. We will later use the following connection between $D_{s,\mathbf{c}}$ and $D_{\Lambda,s,\mathbf{c}}$: if $\mathbf{x}$ is distributed according to $D_{s,\mathbf{c}}$ and we condition on $\mathbf{x} \in \Lambda$, the conditional distribution of $\mathbf{x}$ is $D_{\Lambda,s,\mathbf{c}}$. To see why this is true, recall that our vector $\mathbf{x}$ is in fact chosen from some very fine grid;[9] then, the probability of obtaining some grid point $\mathbf{x}$ in a sample from $D_{s,\mathbf{c}}$ is very close to $\alpha D_{s,\mathbf{c}}(\mathbf{x})$, where α is the volume of one cell in our grid, whereas the probability of $\mathbf{x} \in \Lambda$ is very close to $\alpha D_{s,\mathbf{c}}(\Lambda)$. All our arguments can be made rigorous by working with a fine enough grid.

We will show that for a large enough s, $D_{\Lambda,s,\mathbf{c}}$ behaves in many respects like the continuous Gaussian distribution $D_{s,\mathbf{c}}$. In particular, vectors distributed according to $D_{\Lambda,s,\mathbf{c}}$ have an average value very close to $\mathbf{c}$ and expected squared distance from $\mathbf{c}$ very close to $s^2 n/2\pi$ (for vectors chosen from $D_{s,\mathbf{c}}$, these quantities are exactly $\mathbf{c}$ and $s^2 n/2\pi$). In fact, we define a new lattice parameter that tells us how big s has to be in order for this to happen. We name this parameter the *smoothing parameter*. We then relate this parameter to other lattice parameters such as the length of the shortest vector in the dual lattice and the length of the shortest maximal set of independent vectors.

[9] Although not needed in this paper, one can also define the conditional probability on the continuous random variables directly. This requires some care as it involves conditioning on an event of probability zero.

Fourier transform. We briefly review some of the important properties of the Fourier transform. For a more precise and in-depth treatment, see, e.g., [10]. The Fourier transform of a function $h : \mathbb{R}^n \mapsto \mathbb{R}$ is defined by $\hat{h}(\mathbf{w}) = \int_{\mathbb{R}^n} h(\mathbf{x}) e^{-2\pi i \langle \mathbf{x}, \mathbf{w} \rangle} d\mathbf{x}$. From the definition we can obtain several useful formulas; first, if h is defined by $h(\mathbf{x}) = g(\mathbf{x} + \mathbf{v})$ for some function g and vector $\mathbf{v}$, then

$$\hat{h}(\mathbf{w}) = e^{2\pi i \langle \mathbf{v}, \mathbf{w} \rangle} \hat{g}(\mathbf{w}). \tag{2}$$

Similarly, if h is defined by $h(\mathbf{x}) = e^{2\pi i \langle \mathbf{x}, \mathbf{v} \rangle} g(\mathbf{x})$ for some function g and vector $\mathbf{v}$, then

$$\hat{h}(\mathbf{w}) = \hat{g}(\mathbf{w} - \mathbf{v}). \tag{3}$$

Also, if we denote by $h^{\mathbf{u}}$ the derivative of h in the direction of some unit vector $\mathbf{u}$, then its Fourier transform is

$$\widehat{h^{\mathbf{u}}}(\mathbf{w}) = 2\pi i \langle \mathbf{u}, \mathbf{w} \rangle \cdot \hat{h}(\mathbf{w}). \tag{4}$$

Another important fact is that the Gaussian is its own Fourier transform, i.e., $\hat{\rho} = \rho$. More generally, for any $s > 0$ it holds that $\widehat{\rho_s} = s^n \rho_{1/s}$. We use the following formulation of the Poisson summation formula.

LEMMA 2.8. *For any lattice Λ and any*[10] *function $f : \mathbb{R}^n \to \mathbb{C}$, $f(\Lambda) = \det(\Lambda^*) \hat{f}(\Lambda^*)$, where $\hat{f}$ denotes the Fourier transform of f.*

An immediate application of the Poisson summation formula is the fact that the Gaussian measure $\rho_{s,\mathbf{c}}(\Lambda)$ is maximized when the center is a lattice point $\mathbf{c} \in \Lambda$.

LEMMA 2.9. *For any lattice Λ, positive real $s > 0$, and vector $\mathbf{c}$, $\rho_{s,\mathbf{c}}(\Lambda) \le \rho_s(\Lambda)$.*

Proof. Using Lemma 2.8 twice and (2) we get

$$\begin{aligned} \rho_{s,\mathbf{c}}(\Lambda) &= \det(\Lambda^*) \widehat{\rho_{s,\mathbf{c}}}(\Lambda^*) \\ &= \det(\Lambda^*) \sum_{\mathbf{y} \in \Lambda^*} \widehat{\rho_{s,\mathbf{c}}}(\mathbf{y}) \\ &= \det(\Lambda^*) \sum_{\mathbf{y} \in \Lambda^*} e^{-2\pi i \langle \mathbf{c}, \mathbf{y} \rangle} \widehat{\rho_s}(\mathbf{y}) \\ &\le \det(\Lambda^*) \sum_{\mathbf{y} \in \Lambda^*} \widehat{\rho_s}(\mathbf{y}) = \rho_s(\Lambda), \end{aligned}$$

where we used that $\widehat{\rho_s} = s^n \rho_{1/s}$ is a positive function. ∎

We will also use the following lemma by Banaszczyk.

LEMMA 2.10 (see [5, Lemma 1.5]). *For any $c > 1/\sqrt{2\pi}$, n-dimensional lattice Λ, and vector $\mathbf{v} \in \mathbb{R}^n$,*

$$\rho(\Lambda \setminus c\sqrt{n}\mathcal{B}) < C^n \cdot \rho(\Lambda), \tag{5}$$

$$\rho((\Lambda + \mathbf{v}) \setminus c\sqrt{n}\mathcal{B}) < 2C^n \cdot \rho(\Lambda), \tag{6}$$

where $C = c\sqrt{2\pi e} \cdot e^{-\pi c^2} < 1$.

[10] For this formula to hold, f needs to satisfy certain niceness assumptions. These assumptions always hold in our applications. See [10] for more details.

Sum of independent vectors. We conclude this section with a simple lemma which will be used in section 5 to bound the length of the sum of Gaussian random variables. The lemma essentially shows that when summing m independent random variables, the expected length of the sum grows with $\sqrt{m}$ and not m. (As an example to illustrate the use of the lemma, consider the case $\epsilon = 0$ and $\mathbf{z} = (1, \dots, 1)$.)

LEMMA 2.11. *Let $\mathbf{v}_1, \dots, \mathbf{v}_m$ be m vectors chosen independently from probability distributions $V_1, \dots, V_m$ such that $\mathrm{Exp}[\|\mathbf{v}_i\|^2] \le l$ and $\|\mathrm{Exp}[\mathbf{v}_i]\|^2 \le \epsilon$ for every $i = 1, \dots, m$. Then, for any $\mathbf{z} \in \mathbb{R}^m$, the expected squared norm of $\sum \mathbf{v}_i z_i$ is at most $\mathrm{Exp}[\|\sum_{i=1}^m \mathbf{v}_i z_i\|^2] \le (l + \epsilon \cdot m)\|\mathbf{z}\|^2$.*

Proof. By linearity of expectation and inequality $\sum_i |z_i| \le \sqrt{m}\|\mathbf{z}\|$, we get

$$\begin{aligned}\mathrm{Exp}\left[\Big\|\sum_i \mathbf{v}_i z_i\Big\|^2\right] &= \sum_{i,j} z_i z_j \,\mathrm{Exp}[\langle \mathbf{v}_i, \mathbf{v}_j\rangle] \\ &= \sum_i z_i^2 \,\mathrm{Exp}[\|\mathbf{v}_i\|^2] + \sum_{i \ne j} z_i z_j \langle \mathrm{Exp}[\mathbf{v}_i], \mathrm{Exp}[\mathbf{v}_j]\rangle \\ &\le \|\mathbf{z}\|^2 l + \Big(\sum_i |z_i|\Big)^2 \epsilon \\ &\le \|\mathbf{z}\|^2 (l + \epsilon m). \qquad \square\end{aligned}$$

3. The smoothing parameter. In this section we define a new lattice parameter related to Gaussian measures on lattices. We name it the *smoothing parameter.*

DEFINITION 3.1. *For an n-dimensional lattice Λ, and positive real $\epsilon > 0$, we define its smoothing parameter $\eta_\epsilon(\Lambda)$ to be the smallest s such that $\rho_{1/s}(\Lambda^* \setminus \{\mathbf{0}\}) \le \epsilon$.*

Notice that $\rho_{1/s}(\Lambda^* \setminus \{\mathbf{0}\})$ is a continuous and strictly decreasing function of s such that $\lim_{s \to 0} \rho_{1/s}(\Lambda^* \setminus \{\mathbf{0}\}) = \infty$ and $\lim_{s \to \infty} \rho_{1/s}(\Lambda^* \setminus \{\mathbf{0}\}) = 0$. Thus, the parameter $\eta_\epsilon(\Lambda)$ is well defined for any $\epsilon > 0$, and $\epsilon \mapsto \eta_\epsilon(\Lambda)$ is the inverse function of $s \mapsto \rho_{1/s}(\Lambda^* \setminus \{\mathbf{0}\})$. In particular, $\eta_\epsilon(\Lambda)$ is also a continuous and strictly decreasing function of ϵ.

In this paper we are mostly interested in sequences of lattices Λ_n (in increasing dimension n) and the corresponding smoothing parameters $\eta_{\epsilon(n)}(\Lambda_n)$, where $\epsilon(n)$ is some negligible function of n. Thus, $\eta_{\epsilon(n)}(\Lambda_n)$ is the smallest s such that a Gaussian measure on the dual lattice Λ_n^* with parameter $1/s$ gives all but a negligible amount of its weight to the origin for some negligible function $\epsilon(n)$ of the lattice dimension.

The motivation for this definition (and the name "smoothing parameter") is presented in Lemma 4.1. Intuitively, it says that if we start from a uniformly random lattice point in Λ and perturb it by a Gaussian of radius $\eta_\epsilon(\Lambda)$, then the resulting distribution is $\epsilon/2$ close to uniform on the entire space.[11,12] The next two lemmas relate the smoothing parameter to some standard lattice parameters.

LEMMA 3.2. *For any n-dimensional lattice Λ, $\eta_\epsilon(\Lambda) \le \sqrt{n}/\lambda_1(\Lambda^*)$, where $\epsilon = 2^{-n}$.*

Proof. We use Lemma 2.10 with $c = 1$ and $C = \sqrt{2\pi e} \cdot e^{-\pi} < 1/4$. By separating the right hand side of (5) as the sum over points in $\sqrt{n}\mathcal{B}$ and over points outside $\sqrt{n}\mathcal{B}$

[11] In fact, no uniform probability distribution can be defined over a lattice (or other countably infinite set) or over the entire space. Formally, in order to define this property we follow [18] and capture the intuition of "starting from a random lattice point" by working modulo the lattice. See section 4 for details and [18] for more motivations and explanations about working modulo the lattice.

[12] In fact, a stronger property holds: at any point, the density function of the resulting distribution is within $(1 \pm \epsilon)$ of that of the uniform distribution. Moreover, it can be shown that this stronger property is equivalent to the assumption $s \ge \eta_\epsilon(B)$.

and rearranging, we obtain that for any lattice Λ,

$$\rho(\Lambda \setminus \sqrt{n}\mathcal{B}) < \frac{C^n}{1-C^n}\rho(\Lambda \cap \sqrt{n}\mathcal{B}).$$

Now, let s be such that $s > \sqrt{n}/\lambda_1(\Lambda^*)$. We have

$$\rho_{1/s}(\Lambda^*\setminus\{\mathbf{0}\}) = \rho(s\Lambda^*\setminus\{\mathbf{0}\}) = \rho(s\Lambda^*\setminus\sqrt{n}\mathcal{B}) < \frac{C^n}{1-C^n}\rho(s\Lambda^*\cap\sqrt{n}\mathcal{B}) = \frac{C^n}{1-C^n} < 2^{-n},$$

where we used that the shortest vector in $s\Lambda^*$ is longer than $\sqrt{n}$, and therefore $s\Lambda^* \setminus \sqrt{n}\mathcal{B} = s\Lambda^* \setminus \{\mathbf{0}\}$ and $s\Lambda^* \cap \sqrt{n}\mathcal{B} = \{\mathbf{0}\}$. □

Lemma 3.3. *For any n-dimensional lattice Λ and positive real $\epsilon > 0$,*

$$\eta_\epsilon(\Lambda) \le \sqrt{\frac{\ln(2n(1+1/\epsilon))}{\pi}} \cdot \lambda_n(\Lambda).$$

In particular, for any superlogarithmic function $\omega(\log n)$, there exists a negligible function $\epsilon(n)$ such that $\eta_\epsilon(\Lambda) \le \sqrt{\omega(\log n)} \cdot \lambda_n(\Lambda)$.

Proof. Let $s = \sqrt{\frac{\ln(2(1+1/\epsilon)n)}{\pi}} \cdot \lambda_n(\Lambda)$. Our goal is to show that $\rho_{1/s}(\Lambda^* \setminus \{\mathbf{0}\}) \le \epsilon$. The idea is to show that for any vector $\mathbf{v} \in \Lambda$ of length at most $\lambda_n(\Lambda)$, almost all the contribution to $\rho_{1/s}(\Lambda^*)$ comes from those points in Λ^* that lie on the hyperplane orthogonal to $\mathbf{v}$. Therefore, if we take n linearly independent vectors of length at most $\lambda_n(\Lambda)$, almost all the contribution to $\rho_{1/s}(\Lambda^*)$ must come from the intersection of the corresponding hyperplanes, which is simply the origin. Details follow.

Let $\mathbf{v}_1, \ldots, \mathbf{v}_n$ be a set of n linearly independent vectors in Λ each of length at most $\lambda_n(\Lambda)$. Without loss of generality, we can assume that each $\mathbf{v}_i$ is primitive; i.e., for any $k > 1$, $\mathbf{v}_i/k$ is not in Λ. Define the set $S_{i,j} \subseteq \Lambda^*$ as the set of all points in Λ^* whose inner product with $\mathbf{v}_i$ is $j \in \mathbb{Z}$. Note that for any fixed i, the $S_{i,j}$'s form a partition of Λ^*, and that each $S_{i,j}$ is a translation of $S_{i,0}$. Moreover, since $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \Lambda$ are linearly independent, any nonzero vector in Λ^* must have a nonzero integer inner product with at least one of them, and hence $\Lambda^* \setminus \{\mathbf{0}\} = \bigcup_i(\Lambda^* \setminus S_{i,0})$.

For any index i let $\mathbf{u}_i = \mathbf{v}_i/\|\mathbf{v}_i\|^2$ be a vector of length $1/\|\mathbf{v}_i\| \ge 1/\lambda_n(\Lambda)$ in the same direction as $\mathbf{v}_i$. For all j,

$$\rho_{1/s}(S_{i,j}) = e^{-\pi\|js\mathbf{u}_i\|^2}\rho_{1/s}(S_{i,j} - j\mathbf{u}_i).$$

Now, $S_{i,j} - j\mathbf{u}_i$ is simply a shift of the set $S_{i,0}$. In other words, there exists some vector $\mathbf{w}$ (which is orthogonal to $\mathbf{u}_i$) such that $S_{i,j} - j\mathbf{u}_i = S_{i,0} - \mathbf{w}$. Therefore, by Lemma 2.9,

$$\rho_{1/s}(S_{i,j} - j\mathbf{u}_i) = \rho_{1/s}(S_{i,0} - \mathbf{w}) = \rho_{1/s,\mathbf{w}}(S_{i,0}) \le \rho_{1/s}(S_{i,0}).$$

Using $\|\mathbf{u}_i\| \ge 1/\lambda_n(\Lambda)$ and the bound $\sum_{j\neq 0} x^{-j^2} \le 2\sum_{j>0} x^{-j} = 2/(x-1)$ (valid for all $x > 1$), we get

$$\begin{aligned}\rho_{1/s}(\Lambda^* \setminus S_{i,0}) &= \sum_{j\neq 0}\rho_{1/s}(S_{i,j})\\ &\le \sum_{j\neq 0} e^{-\pi(s/\lambda_n)^2 j^2}\rho_{1/s}(S_{i,0})\\ &\le \frac{2}{e^{\pi(s/\lambda_n)^2}-1}\rho_{1/s}(S_{i,0})\\ &= \frac{2}{e^{\pi(s/\lambda_n)^2}-1}(\rho_{1/s}(\Lambda^*) - \rho_{1/s}(\Lambda^* \setminus S_{i,0})).\end{aligned}$$

Solving for $\rho_{1/s}(\Lambda^* \setminus S_{i,0})$, we get

$$\rho_{1/s}(\Lambda^* \setminus S_{i,0}) \leq \frac{2}{e^{\pi(s/\lambda_n)^2}+1}\rho_{1/s}(\Lambda^*).$$

Since ρ is positive,

$$\rho_{1/s}(\Lambda^* \setminus \{\mathbf{0}\}) \leq \sum_i \rho_{1/s}(\Lambda^* \setminus S_{i,0}) \leq \frac{2n}{e^{\pi(s/\lambda_n)^2}+1}\rho_{1/s}(\Lambda^*).$$

Finally, using $\rho_{1/s}(\Lambda^*) = 1 + \rho_{1/s}(\Lambda^* \setminus \{\mathbf{0}\})$ and solving for $\rho_{1/s}(\Lambda^* \setminus \{\mathbf{0}\})$, we get

$$\rho_{1/s}(\Lambda^* \setminus \{\mathbf{0}\}) \leq \frac{2n}{e^{\pi(s/\lambda_n)^2}+1-2n} < \frac{2n}{e^{\pi(s/\lambda_n)^2}-2n} = \epsilon$$

by our choice of s. □

4. Properties of Gaussian distributions. In this section we prove several properties of Gaussian distributions related to lattices. Our first lemma below justifies the name given to the smoothing parameter.

LEMMA 4.1. *For any $s > 0$, $\mathbf{c} \in \mathbb{R}^n$, and lattice $\mathcal{L}(\mathbf{B})$, the statistical distance between $D_{s,\mathbf{c}} \bmod \mathcal{P}(\mathbf{B})$ and the uniform distribution over $\mathcal{P}(\mathbf{B})$ is at most $\frac{1}{2}\rho_{1/s}(\mathcal{L}(\mathbf{B})^* \setminus \{\mathbf{0}\})$. In particular, for any $\epsilon > 0$ and any $s \geq \eta_\epsilon(\mathbf{B})$, the statistical distance is at most*

$$\Delta(D_{s,\mathbf{c}} \bmod \mathcal{P}(\mathbf{B}), U(\mathcal{P}(\mathbf{B}))) \leq \epsilon/2.$$

Proof. Let Y be the density function of the distribution over $\mathcal{P}(\mathbf{B})$ defined by $(D_{s,\mathbf{c}} \bmod \mathcal{P}(\mathbf{B}))$:

$$Y(\mathbf{x}) = \frac{1}{s^n}\sum_{\mathbf{y}\in\mathcal{L}(\mathbf{B})} \rho_{s,\mathbf{c}}(\mathbf{x}+\mathbf{y}) = \frac{1}{s^n}\rho_{s,\mathbf{c}-\mathbf{x}}(\mathcal{L}(\mathbf{B})).$$

By (2), the Fourier transform of $\rho_{s,\mathbf{c}-\mathbf{x}}$ at point $\mathbf{w}$ is $e^{2\pi i\langle \mathbf{x}-\mathbf{c},\mathbf{w}\rangle}s^n\rho_{1/s}(\mathbf{w})$. Hence, using Lemma 2.8,

$$\begin{aligned} Y(\mathbf{x}) &= \det(\mathcal{L}(\mathbf{B})^*) \sum_{\mathbf{w}\in\mathcal{L}(\mathbf{B})^*} e^{2\pi i\langle \mathbf{x}-\mathbf{c},\mathbf{w}\rangle}\rho_{1/s}(\mathbf{w}) \\ &= \det(\mathcal{L}(\mathbf{B})^*)\left(1 + \sum_{\mathbf{w}\in\mathcal{L}(\mathbf{B})^*\setminus\{\mathbf{0}\}} e^{2\pi i\langle \mathbf{x}-\mathbf{c},\mathbf{w}\rangle}\rho_{1/s}(\mathbf{w})\right). \end{aligned}$$

The density function of the uniform distribution over $\mathcal{P}(\mathbf{B})$ is $U(\mathbf{x}) = 1/\mathrm{vol}(\mathcal{P}(\mathbf{B})) = \det(\mathcal{L}(\mathbf{B})^*)$. Therefore the statistical distance between Y and U is

$$\begin{aligned} \Delta(Y,U) &= \frac{1}{2}\int_{\mathbf{x}\in\mathcal{P}(\mathbf{B})} |Y(\mathbf{x}) - U(\mathbf{x})|d\mathbf{x} \\ &\leq \frac{1}{2}\mathrm{vol}(\mathcal{P}(\mathbf{B}))\cdot \max_{\mathbf{x}\in\mathcal{P}(\mathbf{B})} |Y(\mathbf{x}) - \det(\mathcal{L}(\mathbf{B})^*)| \\ &= \frac{1}{2}\mathrm{vol}(\mathcal{P}(\mathbf{B}))\cdot\det(\mathcal{L}(\mathbf{B})^*)\cdot \max_{\mathbf{x}\in\mathcal{P}(\mathbf{B})}\left|\sum_{\mathbf{w}\in\mathcal{L}(\mathbf{B})^*\setminus\{\mathbf{0}\}} e^{2\pi i\langle \mathbf{x}-\mathbf{c},\mathbf{w}\rangle}\rho_{1/s}(\mathbf{w})\right| \\ &\leq \frac{1}{2}\cdot\rho_{1/s}(\mathcal{L}(\mathbf{B})^*\setminus\{\mathbf{0}\}), \end{aligned}$$

where the last inequality follows by the triangle inequality (and can in fact be replaced by an equality). □

Our second lemma shows that when s is large enough, some statistical properties of the discrete Gaussian distribution $D_{\Lambda,s,\mathbf{c}}$ are very close to those of the continuous Gaussian distribution $D_{s,\mathbf{c}}$.

LEMMA 4.2. *For any n-dimensional lattice Λ, point $\mathbf{c} \in \mathbb{R}^n$, unit vector $\mathbf{u}$, and reals $0 < \epsilon < 1$, $s \geq 2\eta_\epsilon(\Lambda)$,*

$$\left| \operatorname*{Exp}_{\mathbf{x}\sim D_{\Lambda,s,\mathbf{c}}}[\langle \mathbf{x}-\mathbf{c}, \mathbf{u}\rangle] \right| \leq \frac{\epsilon s}{1-\epsilon},$$

$$\left| \operatorname*{Exp}_{\mathbf{x}\sim D_{\Lambda,s,\mathbf{c}}}[\langle \mathbf{x}-\mathbf{c}, \mathbf{u}\rangle^2] - \frac{s^2}{2\pi} \right| \leq \frac{\epsilon s^2}{1-\epsilon}.$$

Proof. For any positive real $s > 0$, define $\Lambda' = \Lambda/s$, $\mathbf{c}' = \mathbf{c}/s$. Notice that, for any $\mathbf{x}$,

$$\Pr\{D_{\Lambda,s,\mathbf{c}} = s\mathbf{x}\} = \frac{\rho_{s,\mathbf{c}}(s\mathbf{x})}{\rho_{s,\mathbf{c}}(\Lambda)} = \frac{\rho_{\mathbf{c}'}(\mathbf{x})}{\rho_{\mathbf{c}'}(\Lambda')} = \Pr\{D_{\Lambda',\mathbf{c}'} = \mathbf{x}\};$$

i.e., the distribution $D_{\Lambda,s,\mathbf{c}}$ is equal to $D_{\Lambda',\mathbf{c}'}$ scaled by a factor of s. Therefore, it is enough to prove the lemma for $s = 1$. The general case follows by scaling the lattice by a factor of s.

In the rest of the proof, we assume $s = 1$. We want to estimate the quantity $\operatorname{Exp}_{\mathbf{x}\sim D_{\Lambda,\mathbf{c}}}[\langle \mathbf{x}-\mathbf{c}, \mathbf{u}\rangle^j]$ for $j = 1, 2$. Without loss of generality, assume that $\mathbf{u}$ is the vector $(1, 0, \dots, 0)$, and define the functions

$$g_j(\mathbf{x}) = (x_1 - c_1)^j \cdot \rho_{\mathbf{c}}(\mathbf{x}),$$

where x_1 and c_1 denote the first coordinate of $\mathbf{x}$ and $\mathbf{c}$, respectively. Notice that

$$\operatorname*{Exp}_{\mathbf{x}\sim D_{\Lambda,\mathbf{c}}}[\langle \mathbf{x}-\mathbf{c}, \mathbf{u}\rangle^j] = \operatorname*{Exp}_{\mathbf{x}\sim D_{\Lambda,\mathbf{c}}}[(x_1 - c_1)^j] = \frac{g_j(\Lambda)}{\rho_{\mathbf{c}}(\Lambda)}.$$

Applying Poisson's summation formula (Lemma 2.8) to the numerator and denominator, the above fraction can be rewritten as

$$\operatorname*{Exp}_{\mathbf{x}\sim D_{\Lambda,\mathbf{c}}}[\langle \mathbf{x}-\mathbf{c}, \mathbf{u}\rangle^j] = \frac{\det(\Lambda^*) \cdot \widehat{g_j}(\Lambda^*)}{\det(\Lambda^*) \cdot \widehat{\rho_{\mathbf{c}}}(\Lambda^*)} = \frac{\widehat{g_j}(\Lambda^*)}{\widehat{\rho_{\mathbf{c}}}(\Lambda^*)}. \tag{7}$$

The Fourier transform $\widehat{\rho_{\mathbf{c}}}$ is easily computed using (2):

$$\widehat{\rho_{\mathbf{c}}}(\mathbf{y}) = \rho(\mathbf{y}) e^{-2\pi i \langle \mathbf{y}, \mathbf{c}\rangle}.$$

In particular, $\widehat{\rho_{\mathbf{c}}}(\mathbf{0}) = 1$, $|\widehat{\rho_{\mathbf{c}}}(\mathbf{y})| = \rho(\mathbf{y})$, and

$$|\widehat{\rho_{\mathbf{c}}}(\Lambda^*)| = \left| 1 + \sum_{\mathbf{y}\in\Lambda^*\setminus\{\mathbf{0}\}} \widehat{\rho_{\mathbf{c}}}(\mathbf{y}) \right| \geq 1 - \rho(\Lambda^* \setminus \{\mathbf{0}\}) \geq 1 - \epsilon, \tag{8}$$

where the last inequality uses $\eta_\epsilon(\Lambda) \leq \frac{1}{2} \leq 1$.

We evaluate the Fourier transform $\widehat{g_j}$ (for $j = 1, 2$) as follows. For any $j \geq 0$, let

$$\rho_{\mathbf{c}}^{(j)}(\mathbf{x}) = \left(\frac{\partial}{\partial x_1}\right)^j \rho_{\mathbf{c}}(\mathbf{x})$$

be the jth partial derivative of $\rho_{\mathbf{c}}(\mathbf{x})$ with respect to x_1. It is easy to see that

$$\rho_{\mathbf{c}}^{(1)}(\mathbf{x}) = -2\pi(x_1 - c_1)\rho_{\mathbf{c}}(\mathbf{x}),$$
$$\rho_{\mathbf{c}}^{(2)}(\mathbf{x}) = (4\pi^2(x_1 - c_1)^2 - 2\pi)\rho_{\mathbf{c}}(\mathbf{x}).$$

Taking linear combinations of the previous equations, we can express the g_j functions as

$$g_1 = -\frac{1}{2\pi}\rho_{\mathbf{c}}^{(1)},$$
$$g_2 = \frac{1}{4\pi^2}\rho_{\mathbf{c}}^{(2)} + \frac{1}{2\pi}\rho_{\mathbf{c}}.$$

Using $\widehat{\rho_{\mathbf{c}}^{(j)}}(\mathbf{y}) = (2\pi i y_1)^j \widehat{\rho_{\mathbf{c}}}(\mathbf{y})$ (see (4)) and the linearity of the Fourier transform, we get

$$\widehat{g_1}(\mathbf{y}) = -iy_1\widehat{\rho_{\mathbf{c}}}(\mathbf{y}), \tag{9}$$

$$\widehat{g_2}(\mathbf{y}) = \left(\frac{1}{2\pi} - y_1^2\right)\widehat{\rho_{\mathbf{c}}}(\mathbf{y}). \tag{10}$$

We are now ready to evaluate expression (7). For $j = 1$, using (9) and (8), we get

$$\left|\underset{\mathbf{x}\sim D_{\Lambda,\mathbf{c}}}{\mathrm{Exp}}[\langle \mathbf{x} - \mathbf{c}, \mathbf{u}\rangle]\right| \leq \frac{\sum_{\mathbf{y}\in\Lambda^*} |y_1| \cdot |\widehat{\rho_{\mathbf{c}}}(\mathbf{y})|}{1 - \epsilon}.$$

We use $|y_1| \leq \sqrt{\|\mathbf{y}\|^2} \leq e^{\|\mathbf{y}\|^2/2}$ and $|\widehat{\rho_{\mathbf{c}}}(\mathbf{y})| = \rho(\mathbf{y})$ to bound the numerator:

$$\begin{aligned}\sum_{\mathbf{y}\in\Lambda^*} |y_1||\widehat{\rho_{\mathbf{c}}}(\mathbf{y})| &= \sum_{\mathbf{y}\in\Lambda^*\setminus\{\mathbf{0}\}} |y_1| \cdot \rho(\mathbf{y}) \\ &\leq \sum_{\mathbf{y}\in\Lambda^*\setminus\{\mathbf{0}\}} e^{\|\mathbf{y}\|^2/2} \cdot e^{-\pi\|\mathbf{y}\|^2} \\ &\leq \sum_{\mathbf{y}\in\Lambda^*\setminus\{\mathbf{0}\}} e^{-\pi\|\mathbf{y}/2\|^2} = \rho_2(\Lambda^* \setminus \{\mathbf{0}\}) \leq \epsilon,\end{aligned}$$

where the last inequality uses $\eta_\epsilon(\Lambda) \leq \frac{1}{2}$. This completes the proof for $j = 1$.

For $j = 2$, combining (7), (8), (10), and $|\widehat{\rho_{\mathbf{c}}}(\mathbf{y})| = \rho(\mathbf{y})$, we get

$$\begin{aligned}\left|\underset{\mathbf{x}\sim D_{\Lambda,\mathbf{c}}}{\mathrm{Exp}}[\langle \mathbf{x} - \mathbf{c}, \mathbf{u}\rangle^2] - \frac{1}{2\pi}\right| &= \frac{|\sum_{\mathbf{y}\in\Lambda^*} y_1^2 \cdot \widehat{\rho_{\mathbf{c}}}(\mathbf{y})|}{\widehat{\rho_{\mathbf{c}}}(\Lambda^*)} \\ &\leq \frac{\sum_{\mathbf{y}\in\Lambda^*} y_1^2 \cdot \rho(\mathbf{y})}{1 - \rho(\Lambda^* \setminus \{\mathbf{0}\})}.\end{aligned}$$

This time we use $y_1^2 \leq \|\mathbf{y}\|^2 \leq e^{\|\mathbf{y}\|^2}$ to bound the numerator:

$$\begin{aligned}\sum_{\mathbf{y}\in\Lambda^*} y_1^2 \cdot \rho(\mathbf{y}) &\leq \sum_{\mathbf{y}\in\Lambda^*\setminus\{\mathbf{0}\}} e^{\|\mathbf{y}\|^2} \cdot e^{-\pi\|\mathbf{y}\|^2} \leq \sum_{\mathbf{y}\in\Lambda^*\setminus\{\mathbf{0}\}} e^{-\pi\|\mathbf{y}/2\|^2} \\ &= \rho_2(\Lambda^* \setminus \{\mathbf{0}\}) \leq \epsilon. \qquad \square\end{aligned}$$

As a corollary, we obtain the following lemma.

LEMMA 4.3. *For any n-dimensional lattice Λ, vector $\mathbf{c} \in \mathbb{R}^n$, and reals $0 < \epsilon < 1$, $s \geq 2\eta_\epsilon(\Lambda)$, we have*

$$\left\| \operatorname*{Exp}_{\mathbf{x} \sim D_{\Lambda,s,\mathbf{c}}} [\mathbf{x} - \mathbf{c}] \right\|^2 \leq \left(\frac{\epsilon}{1-\epsilon} \right)^2 s^2 n,$$

$$\operatorname*{Exp}_{\mathbf{x} \sim D_{\Lambda,s,\mathbf{c}}} [\|\mathbf{x} - \mathbf{c}\|^2] \leq \left(\frac{1}{2\pi} + \frac{\epsilon}{1-\epsilon} \right) s^2 n.$$

Proof. Take any orthonormal basis $\mathbf{u}_1, \ldots, \mathbf{u}_n$. Using Lemma 4.2, we get

$$\left\| \operatorname*{Exp}_{\mathbf{x} \sim D_{\Lambda,s,\mathbf{c}}} [\mathbf{x} - \mathbf{c}] \right\|^2 = \sum_{i=1}^{n} \left(\operatorname*{Exp}_{\mathbf{x} \sim D_{\Lambda,s,\mathbf{c}}} [\langle \mathbf{x} - \mathbf{c}, \mathbf{u}_i \rangle] \right)^2 \leq ns^2 \cdot \left(\frac{\epsilon}{1-\epsilon} \right)^2$$

and

$$\operatorname*{Exp}_{\mathbf{x} \sim D_{\Lambda,s,\mathbf{c}}} [\|\mathbf{x} - \mathbf{c}\|^2] = \sum_{i=1}^{n} \operatorname*{Exp}_{\mathbf{x} \sim D_{\Lambda,s,\mathbf{c}}} [\langle \mathbf{x} - \mathbf{c}, \mathbf{u}_i \rangle^2] \leq ns^2 \cdot \left(\frac{1}{2\pi} + \frac{\epsilon}{1-\epsilon} \right). \qquad \square$$

The remaining lemmas describe some additional properties of the discrete Gaussian distribution. These lemmas are used only in our GAPSVP result of subsection 5.4.

LEMMA 4.4. *For any n-dimensional lattice Λ, vector $\mathbf{c} \in \mathbb{R}^n$, and reals $0 < \epsilon < 1$, $s \geq \eta_\epsilon(\Lambda)$, we have*

$$\Pr_{\mathbf{x} \sim D_{\Lambda,s,\mathbf{c}}} \{\|\mathbf{x} - \mathbf{c}\| > s\sqrt{n}\} \leq \frac{1+\epsilon}{1-\epsilon} \cdot 2^{-n}.$$

Proof. As in the proof of Lemma 4.2, it is enough to prove the lemma for $s = 1$. We can write

$$\Pr_{\mathbf{x} \sim D_{\Lambda,\mathbf{c}}} \{\|\mathbf{x} - \mathbf{c}\| > \sqrt{n}\} = \frac{\rho((\Lambda - \mathbf{c}) \setminus \sqrt{n}\mathcal{B})}{\rho_{\mathbf{c}}(\Lambda)}.$$

By Lemma 2.10 with $c = 1$, the numerator is at most $2^{-n}\rho(\Lambda)$. By the Poisson summation formula (Lemma 2.8),

$$\begin{aligned} \rho_{\mathbf{c}}(\Lambda) &= \det(\Lambda^*) \widehat{\rho_{\mathbf{c}}}(\Lambda^*) \\ &= \det(\Lambda^*) \sum_{\mathbf{y} \in \Lambda^*} \widehat{\rho_{\mathbf{c}}}(\mathbf{y}) \\ &= \det(\Lambda^*) \sum_{\mathbf{y} \in \Lambda^*} e^{-2\pi i \langle \mathbf{c}, \mathbf{y} \rangle} \widehat{\rho}(\mathbf{y}) \\ &= \det(\Lambda^*)(1 + \delta), \end{aligned}$$

where $|\delta| \leq |\rho(\Lambda^* \setminus \{\mathbf{0}\})| \leq \epsilon$. Therefore $\rho_{\mathbf{c}}(\Lambda) \geq \det(\Lambda^*)(1-\epsilon)$, $\rho(\Lambda) \leq \det(\Lambda^*)(1+\epsilon)$, and $2^{-n}\rho(\Lambda)/\rho_{\mathbf{c}}(\Lambda) \leq 2^{-n}\frac{1+\epsilon}{1-\epsilon}$. $\square$

LEMMA 4.5. *Let Λ be an n-dimensional lattice, $\mathbf{c}, \mathbf{v}$ be two points in $\mathbb{R}^n$, $0 < \epsilon < 1$, and $s \geq \eta_\epsilon(\Lambda)$ such that* $\operatorname{dist}(\mathbf{v}, \Lambda^*) \geq \sqrt{n}/s$. *Then,*

$$\left| \operatorname*{Exp}_{\mathbf{x} \sim D_{\Lambda,s,\mathbf{c}}} [e^{2\pi i \langle \mathbf{x}, \mathbf{v} \rangle}] \right| \leq \frac{1+\epsilon}{1-\epsilon} \cdot 2^{-n}.$$

Proof. Define $\Lambda' = \Lambda/s$, $\mathbf{c}' = \mathbf{c}/s$, and $\mathbf{v}' = s\mathbf{v}$. As in the proof of Lemma 4.2, the distribution $D_{\Lambda,s,\mathbf{c}}$ is equal to $D_{\Lambda',\mathbf{c}'}$ scaled by a factor of s. Therefore, it is enough to prove the lemma for the case $s = 1$.

Define the function

$$g(\mathbf{x}) = e^{2\pi i\langle \mathbf{x},\mathbf{v}\rangle} \cdot \rho_{\mathbf{c}}(\mathbf{x})$$

and notice that

$$\operatorname*{Exp}_{\mathbf{x}\sim D_{\Lambda,\mathbf{c}}}[e^{2\pi i\langle \mathbf{x},\mathbf{v}\rangle}] = \frac{g(\Lambda)}{\rho_{\mathbf{c}}(\Lambda)}.$$

Applying Poisson's summation formula (Lemma 2.8) to the numerator and denominator, the above fraction can be rewritten as

$$\operatorname*{Exp}_{\mathbf{x}\sim D_{\Lambda,\mathbf{c}}}[e^{2\pi i\langle \mathbf{x},\mathbf{v}\rangle}] = \frac{\det(\Lambda^*)\cdot \widehat{g}(\Lambda^*)}{\det(\Lambda^*)\cdot \widehat{\rho_{\mathbf{c}}}(\Lambda^*)} = \frac{\widehat{g}(\Lambda^*)}{\widehat{\rho_{\mathbf{c}}}(\Lambda^*)}. \tag{11}$$

As in the proof of Lemma 4.2, we have $\widehat{\rho_{\mathbf{c}}}(\mathbf{y}) = \rho(\mathbf{y})e^{-2\pi i\langle \mathbf{y},\mathbf{c}\rangle}$ and $|\widehat{\rho_{\mathbf{c}}}(\Lambda^*)| \geq 1-\rho(\Lambda^*\setminus\{\mathbf{0}\})$. By (3), the Fourier transform of g is given by

$$\widehat{g}(\mathbf{y}) = \widehat{\rho_{\mathbf{c}}}(\mathbf{y}-\mathbf{v}) = \rho(\mathbf{y}-\mathbf{v})e^{-2\pi i\langle \mathbf{y}-\mathbf{v},\mathbf{c}\rangle}.$$

Combined with (11), we obtain

$$\left|\operatorname*{Exp}_{\mathbf{x}\sim D_{\Lambda,\mathbf{c}}}[e^{2\pi i\langle \mathbf{x},\mathbf{v}\rangle}]\right| \leq \frac{\rho(\Lambda^*-\mathbf{v})}{1-\rho(\Lambda^*\setminus\{\mathbf{0}\})}.$$

Since $\operatorname{dist}(\mathbf{v},\Lambda^*) \geq \sqrt{n}$, Lemma 2.10 with $c = 1$ implies that

$$\rho(\Lambda^*-\mathbf{v}) \leq 2^{-n}\rho(\Lambda^*) = 2^{-n}(1+\rho(\Lambda^*\setminus\{\mathbf{0}\})),$$

so we have

$$\left|\operatorname*{Exp}_{\mathbf{x}\sim D_{\Lambda,\mathbf{c}}}[e^{2\pi i\langle \mathbf{x},\mathbf{v}\rangle}]\right| \leq 2^{-n}\frac{1+\rho(\Lambda^*\setminus\{\mathbf{0}\})}{1-\rho(\Lambda^*\setminus\{\mathbf{0}\})}. \qquad \square$$

Using the lemma, we obtain the following easy corollary.

COROLLARY 4.6. *Let* Λ *be an* n*-dimensional lattice,* $\mathbf{w},\mathbf{c},\mathbf{v}\in\mathbb{R}^n$*,* $0<\epsilon<1$*, and* $s \geq \eta_\epsilon(\Lambda)$ *such that* $\operatorname{dist}(\mathbf{v},\Lambda^*) \geq \sqrt{n}/s$*. Then,*

$$\left|\operatorname*{Exp}_{\mathbf{x}\sim D_{\Lambda,s,\mathbf{c}}}[\cos(2\pi\langle \mathbf{x}+\mathbf{w},\mathbf{v}\rangle)]\right| \leq \frac{1+\epsilon}{1-\epsilon}\cdot 2^{-n}.$$

Proof.

$$\begin{aligned}\left|\operatorname*{Exp}_{\mathbf{x}\sim D_{\Lambda,s,\mathbf{c}}}[\cos(2\pi\langle \mathbf{x}+\mathbf{w},\mathbf{v}\rangle)]\right| &= \left|\Re\Big(\operatorname*{Exp}_{\mathbf{x}\sim D_{\Lambda,s,\mathbf{c}}}[e^{2\pi i\langle \mathbf{x}+\mathbf{w},\mathbf{v}\rangle}]\Big)\right| \\ &\leq \left|\operatorname*{Exp}_{\mathbf{x}\sim D_{\Lambda,s,\mathbf{c}}}[e^{2\pi i\langle \mathbf{x}+\mathbf{w},\mathbf{v}\rangle}]\right| \\ &= \left|\operatorname*{Exp}_{\mathbf{x}\sim D_{\Lambda,s,\mathbf{c}}}[e^{2\pi i\langle \mathbf{x},\mathbf{v}\rangle}]\right|. \qquad \square\end{aligned}$$

5. Worst-case to average-case connection. In this section we show that if various lattice problems are hard to solve in the worst case, then a certain computational problem is hard to solve on the average. We start in subsection 5.1 with a description of the average-case problem. We then describe our reductions. Following [21, 22], the reductions are performed in two steps. First, in subsection 5.2, we present a reduction from an intermediate worst-case lattice problem to the average-case problem. This is the core of our proof. Then, in subsection 5.3, we show that the intermediate worst-case lattice problem is at least as hard as various other computational problems on lattices, such as SIVP and GapCRP. We remark that the intermediate worst-case problem is introduced to present the worst-case to average-case reduction in a simpler setting where the worst-case algorithm makes a single call to the average-case oracle. This allows for a cleaner and simpler probabilistic analysis, and it is well worth the effort of introducing one additional and perhaps artificial problem. Work prior to [21, 22] reduced standard worst-case lattice problems (like SIVP) directly to the average-case problem by making (polynomially) many random calls to the average-case oracle, resulting in an overall more complex probabilistic argument.

In subsection 5.4 we present our reduction from $\mathrm{GapSVP}_{\tilde{O}(n)}$ to the average-case problem. The proof of this result requires some additional machinery and relies on the results proved in subsections 5.2 and 5.3 as well as techniques from [1]. We remark that a weaker result can be derived directly from the results in subsection 5.3. Namely, using standard reductions between lattice problems (see [23, Theorem 7.12]), our $\tilde{O}(n)$ approximation to SIVP immediately implies an $\tilde{O}(n^2)$ approximation to GapSVP. Hence, subsection 5.4 is needed only in order to reduce the approximation factor to $\tilde{O}(n)$.

5.1. The average-case problem. Our average-case problem is the problem of finding small nonzero solutions to random linear systems of modular equations.

Definition 5.1. *The* small integer solution *problem* SIS *(in the ℓ_2 norm) is as follows: given an integer q, a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, and a real β, find a nonzero integer vector $\mathbf{z} \in \mathbb{Z}^m \setminus \{\mathbf{0}\}$ such that $\mathbf{Az} = \mathbf{0} \bmod q$ and $\|\mathbf{z}\| \leq \beta$.*

Equivalently, the SIS problem asks to find a vector $\mathbf{z} \in \Lambda_q(\mathbf{A}) \setminus \{\mathbf{0}\}$ with $\|\mathbf{z}\| \leq \beta$, where

$$\Lambda_q(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m \colon \mathbf{Az} = \mathbf{0} \bmod q\}$$

is the set of all integer solutions to the system of linear equations modulo q defined by matrix $\mathbf{A}$.

In the definition of SIS it is implicitly assumed that a solution of length $\|\mathbf{z}\| \leq \beta$ exists (for otherwise the problem is trivially hard). The following lemma gives sufficient conditions under which SIS instances are guaranteed to have a solution.

Lemma 5.2. *For any q, $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, and $\beta \geq \sqrt{m}q^{n/m}$, the* SIS *instance $(q, \mathbf{A}, \beta)$ admits a solution; i.e., there exists a vector $\mathbf{z} \in \mathbb{Z}^m \setminus \{\mathbf{0}\}$ such that $\mathbf{Az} = \mathbf{0} \bmod q$ and $\|\mathbf{z}\| \leq \beta$.*

Proof. The proof is by the pigeonhole principle. Consider all vectors $\mathbf{z} \in \mathbb{Z}^m$ with coordinates in $\{0, \ldots, q^{n/m}\}$. There are more than q^n such vectors, and hence there must exist two such vectors $\mathbf{z}_1 \neq \mathbf{z}_2$ for which $\mathbf{Az}_1 = \mathbf{Az}_2 \bmod q$. Then, $\mathbf{z}_1 - \mathbf{z}_2 \neq \mathbf{0}$ satisfies $\mathbf{A}(\mathbf{z}_1 - \mathbf{z}_2) = \mathbf{0} \bmod q$ and, moreover, $\|\mathbf{z}_1 - \mathbf{z}_2\| \leq \sqrt{m}q^{n/m}$ since all its coordinates are between $-q^{n/m}$ and $q^{n/m}$. □

We want to study the average-case complexity of the SIS problem when $\beta \geq \sqrt{m}q^{n/m}$ satisfies the condition in Lemma 5.2 and SIS instances $(q, \mathbf{A}, \beta)$ are guaranteed to have a solution. In order to define probability ensembles over SIS instances, it is convenient to use the number of equations n as a security parameter and consider families of SIS instances indexed by functions $q(n)$, $m(n)$, and $\beta(n)$ that express the other parameters in terms of n.

DEFINITION 5.3. *For any functions $q(n)$, $m(n)$, and $\beta(n)$, let*

$$\mathrm{SIS}_{q,m,\beta} = \{(q(n), U(\mathbb{Z}_{q(n)}^{n\times m(n)}), \beta(n))\}_n$$

be the probability ensemble over SIS *instances $(q(n), \mathbf{A}, \beta(n))$, where $\mathbf{A}$ is chosen uniformly at random among all $n \times m(n)$ integer matrices modulo $q(n)$. When $\beta(n) = \sqrt{m(n)}q(n)^{n/m(n)}$ is the bound specified in Lemma* 5.2, *the parameter $\beta(n)$ is often omitted, and we simply write* $\mathrm{SIS}_{q,m}$.

Notice that for the instances of $\mathrm{SIS}_{q,m,\beta}$ to be of size polynomial in n, the number of variables must be a polynomially bounded function $m(n) = n^{O(1)}$, but $q(n)$ and $\beta(n)$ can be exponentially large. However, we will be mostly interested in instances where $q(n)$ and $\beta(n)$ are also polynomially bounded functions of the security parameter n. Moreover, we typically choose values of q and m satisfying $q(n)^{n/m(n)} = O(1)$, so that $\beta(n) = \sqrt{m(n)}q(n)^{n/m(n)} = O(\sqrt{m(n)})$. In the next two subsections we show that for an appropriate choice of parameters q, m, and β, solving $\mathrm{SIS}_{q,m,\beta}$ on the average is as hard as solving worst-case instances of several standard lattice problems such as SIVP and GAPCRP. The reduction from GAPSVP is shown in subsection 5.4. For technical reasons, in that reduction we need to consider a variant of the SIS problem, defined below, which extends SIS with the additional requirement that the solution vector must contain at least one odd coordinate.

DEFINITION 5.4. *The* SIS' *problem (in the ℓ_2 norm) is as follows: given an integer q, a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, and a real β, find an integer vector $\mathbf{z} \in \mathbb{Z}^m \setminus 2\mathbb{Z}^m$ such that $\mathbf{A}\mathbf{z} = \mathbf{0} \bmod q$ and $\|\mathbf{z}\| \leq \beta$.*

The distribution ensemble $\mathrm{SIS}'_{q,m,\beta} = \{(q(n), \mathbf{A}, \beta(n))\}_n$ is defined analogously to $\mathrm{SIS}_{q,m,\beta}$ by choosing matrix $\mathbf{A} \in \mathbb{Z}_{q(n)}^{n\times m(n)}$ uniformly at random. Similarly, when $\beta(n) = \sqrt{m(n)}q(n)^{n/m(n)}$, we omit the parameter β and simply write $\mathrm{SIS}'_{q,m}$. Clearly, any solution to $\mathrm{SIS}'_{q,m,\beta}$ is also a solution to $\mathrm{SIS}_{q,m,\beta}$ because $\mathbf{0} \notin \mathbb{Z}^m \setminus 2\mathbb{Z}^m$. The next lemma shows that when the modulus $q(n)$ is odd, $\mathrm{SIS}'_{q,m,\beta}$ is not any harder than $\mathrm{SIS}_{q,m,\beta}$.

LEMMA 5.5. *For any odd integer $q \in 2\mathbb{Z} + 1$ and* SIS' *instance $I = (q, \mathbf{A}, \beta)$, if I has a solution as an instance of* SIS, *then it also has a solution as an instance of* SIS'. *Moreover, there is a polynomial time algorithm that on input a solution to an* SIS *instance I, outputs a solution to the same* SIS' *instance I.*

Proof. Assume q is odd, and let $\mathbf{z}$ be a solution to SIS instance $(q, \mathbf{A}, \beta)$; i.e., assume $\mathbf{z} \in \mathbb{Z}^m \setminus \{\mathbf{0}\}$, $\mathbf{A}\mathbf{z} = \mathbf{0} \bmod q$, and $\|\mathbf{z}\| \leq \beta$. Compute the largest power i such that 2^i divides all the coordinates of $\mathbf{z}$, and output $\mathbf{z}/2^i$. Since $\mathbf{z}$ is nonzero, i is well defined and can be easily computed. Moreover, $\mathbf{z}/2^i$ has at least one odd coordinate, and since the modulus $q(n)$ is odd, $\mathbf{z}/2^i$ satisfies $\mathbf{A}(\mathbf{z}/2^i) = \mathbf{0} \bmod q$. □

We end this subsection with two simple observations on the average-case hardness of SIS. These observations are not used in the following subsections and can be safely skipped at first reading.

First, observe that for any $\mathbf{A}$, $\Lambda_q(\mathbf{A})$ forms a lattice. Therefore, the SIS problem is closely related to the shortest vector problem (SVP) on lattices of the form $\Lambda_q(\mathbf{A})$. More specifically, finding shortest nonzero vectors in a random lattice $\Lambda_q(\mathbf{A})$ is at least as hard as solving $\mathrm{SIS}_{q,m}$ on the average. So, all our results can be formulated as reductions from solving various lattice problems (including GapSVP$_\gamma$ for factors $\gamma(n) = \tilde{O}(n)$) in the worst case to solving SVP on the average (for random lattices of the form $\Lambda_q(\mathbf{A})$).

Next, we observe that SIS can be reduced to the problem of finding collisions for an appropriately defined family of hash functions. For any $q(n), m(n)$, and $d(n)$, define the family of functions

$$\mathcal{H}_{q,m,d} = \{f_{\mathbf{A}} \colon \{0, \ldots, d(n) - 1\}^{m(n)} \to \mathbb{Z}^n_{q(n)} \mid \mathbf{A} \in \mathbb{Z}^{n \times m(n)}_{q(n)}\},$$

where n is a security parameter and $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q(n)$. A typical choice of parameters is $q(n) = n^{O(1)}$, $d(n) = 2$, and $m(n) > n \log_2 q(n) = \Theta(n \log n)$. A collision is a pair of distinct inputs $\mathbf{x} \neq \mathbf{y}$ (both in the domain $\{0, \ldots, d(n) - 1\}^{m(n)}$ of $f_{\mathbf{A}}$) that are mapped to the same output $f_{\mathbf{A}}(\mathbf{x}) = f_{\mathbf{A}}(\mathbf{y})$. Notice that if $m(n) > n \log_{d(n)} q(n)$, then the domain $\{0, \ldots, d(n) - 1\}^{m(n)}$ is larger than the range $\mathbb{Z}^n_{q(n)}$, and, by the pigeonhole principle, the functions $f_{\mathbf{A}}$ are guaranteed to have collisions $(\mathbf{x}, \mathbf{y})$. We argue that these collisions are computationally hard to find when $\mathbf{A}$ is chosen at random. Observe that if $(\mathbf{x}, \mathbf{y})$ is a collision for $f_{\mathbf{A}}$, then $\mathbf{z} = \mathbf{x} - \mathbf{y} \in \Lambda_q(\mathbf{A}) \setminus \{\mathbf{0}\}$ is a nonzero lattice vector of length at most $\beta(n) = (d(n) - 1)\sqrt{m(n)}$. So, finding collisions on the average when $\mathbf{A}$ is chosen uniformly at random is at least as hard as solving random instances of SIS$_{q,m,\beta}$ for the same value of $q(n)$ and $m(n)$, and $\beta(n) = (d(n) - 1)\sqrt{m(n)}$. This gives collision resistant hash functions that are provably secure based on the worst-case intractability assumption of lattice approximation problems (e.g., SIVP$_\gamma$, GapCRP$_\gamma$, GapSVP$_\gamma$) for approximation factors $\gamma(n) = \tilde{O}(n)$ almost linear in the dimension of the lattice.

5.2. Incremental guaranteed distance decoding. In this section we show that solving SIS on the average with nonnegligible probability is at least as hard as solving worst-case instances of the following IncGDD problem (originally introduced in [22] in a slightly different form). We remind the reader that we introduce IncGDD for the sole purpose of simplifying the worst-case to average-case reduction. In particular, we will show that IncGDD can be solved (in the worst case) by making a single call to the average-case SIS oracle, resulting in a simpler probabilistic analysis compared to reductions that make several oracle calls. In the next subsection we show that several other more interesting lattice problems (like SIVP and GapCRP) can be solved in the worst case by making many calls to an IncGDD oracle. Although these reductions require the solution of several IncGDD instances, they are conceptually easier to analyze because they are standard worst-case to worst-case reductions.

Definition 5.6 (incremental guaranteed distance decoding). *An input to the problem* IncGDD$^\phi_{\gamma,g}$ *is an n-dimensional lattice basis* $\mathbf{B}$*, a set of n linearly independent vectors* $\mathbf{S} \subset \mathcal{L}(\mathbf{B})$*, a target point* $\mathbf{t}$*, and a real* $r > \gamma(n) \cdot \phi(\mathbf{B})$*. The goal is to output a lattice vector* $\mathbf{s} \in \mathcal{L}(\mathbf{B})$ *such that* $\|\mathbf{s} - \mathbf{t}\| \leq (\|\mathbf{S}\|/g) + r$*.*

In other words, the IncGDD problem asks to find a lattice vector within distance $(\|\mathbf{S}\|/g) + r$ from the given target. One possible choice of parameters is, for example, $g = 4$, $\gamma(n) = \sqrt{n}/2$, and $\phi = \lambda_n$. Often, $\|\mathbf{S}\|$ is much larger than r, so the dominant part in the distance bound is $\|\mathbf{S}\|/g$, or $\|\mathbf{S}\|/4$ for our choice of parameters. Notice that using the nearest plane algorithm [4] one can always find (in polynomial time) a lattice point within distance $(\sqrt{n}/2)\|\mathbf{S}\|$ from any target. Here we are trying to do much better than that. However, it is not always possible to find a lattice vector within distance $\|\mathbf{S}\|/4$ of a given target vector: for example, consider the integer lattice $\mathbb{Z}^n$ generated by the identity matrix $\mathbf{B} = \mathbf{I}$. If we choose the set $\mathbf{S} = \mathbf{I}$ and the target point $\mathbf{t} = (1/2, \ldots, 1/2)$, then there is no lattice point at distance strictly less than $\sqrt{n}/2 = (\sqrt{n}/2)\|\mathbf{S}\|$ from the target. The r term in the distance bound of the IncGDD problem is introduced to guarantee the existence of a solution. For example, using the above choice of parameters, we get $r > \gamma(n)\phi(\mathbf{B}) = (\sqrt{n}/2)\lambda_n(\mathbf{B})$, and a lattice point within this distance always exists by the nearest plane algorithm. To summarize, one can think of IncGDD as asking to find a lattice point within distance roughly $\|\mathbf{S}\|/g$ from the target, provided that $\|\mathbf{S}\|$ is not too small.

We now give a high-level overview of the reduction. Our goal is to reduce worst-case instances of IncGDD to random instances of SIS. In other words, we want to solve an IncGDD instance $(\mathbf{B}, \mathbf{S}, \mathbf{t}, r)$ with the help of an oracle $\mathcal{F}$ that on input a random matrix $\mathbf{A}$, returns with nonnegligible probability a short nonzero integer vector $\mathbf{z}$ such that $\mathbf{Az} = \mathbf{0}$. To fix some parameters, assume that we want to reduce IncGDD with, say, $g = 4$ (we ignore γ and ϕ in this discussion) to SIS with $q(n) = n^4$, $m(n) = n \log n$, and $\beta(n) = n$ (it is easy to check that for large enough n, this choice satisfies the conditions in Lemma 5.2). For now, let us make two simplifying assumptions: the target vector $\mathbf{t}$ is the origin $\mathbf{0}$, and $\mathbf{S} = \mathbf{B}$. Although the former assumption makes the IncGDD instance trivial ($\mathbf{0} \in \mathcal{L}(\mathbf{B})$ is always a solution), it helps in explaining the main ideas in the reduction. We will later indicate how to avoid these assumptions.

With these assumptions in place, we can describe a simplified form of the reduction. At the core of the reduction is a sampling procedure $\mathcal{S}$. This procedure generates a pair $(\mathbf{c}, \mathbf{y})$, where $\mathbf{c}$ is distributed uniformly in $\mathcal{P}(\mathbf{B})$ and $\mathbf{y} \in \mathcal{L}(\mathbf{B})$ is a lattice vector close to $\mathbf{c}$. The reduction starts by applying the sampling procedure m times to obtain m pairs $(\mathbf{c}_1, \mathbf{y}_1), \ldots, (\mathbf{c}_m, \mathbf{y}_m)$. We then partition the parallelepiped $\mathcal{P}(\mathbf{B})$ into q^n smaller parallelepipeds, naturally corresponding to elements of $\mathbb{Z}_q^n$. For each $\mathbf{c}_i$, let $\tilde{\mathbf{c}}_i$ be the "lower-left" corner of the parallelepiped of $\mathbf{c}_i$, that is, $\tilde{\mathbf{c}}_i = \mathbf{B}\lfloor q \cdot \mathbf{B}^{-1}\mathbf{c}_i \rfloor / q$. Notice that the distance between $\mathbf{c}_i$ and $\tilde{\mathbf{c}}_i$ is at most $n\|\mathbf{B}\|/q = \|\mathbf{B}\|/n^3$. Next, let $\mathbf{a}_i \in \mathbb{Z}_q^n$ be the group element corresponding to the parallelepiped that contains $\mathbf{c}_i$. More precisely, we define $\mathbf{a}_i = \lfloor q \cdot \mathbf{B}^{-1}\mathbf{c}_i \rfloor \bmod q$. Since each $\mathbf{c}_i$ is uniformly distributed in $\mathcal{P}(\mathbf{B})$, each $\mathbf{a}_i$ is uniformly distributed in $\mathbb{Z}_q^n$. We can therefore apply the oracle $\mathcal{F}$ to the matrix $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_m]$ to find a small combination of the $\mathbf{a}_i$ that sums to zero in $\mathbb{Z}_q^n$. That is, we find a vector $\mathbf{z}$ such that $\mathbf{Az} = \mathbf{0}$ and $\|\mathbf{z}\|_1 \leq \sqrt{m}\|\mathbf{z}\|_2 \leq \sqrt{m}\beta \leq n^2$. Crucially, the same combination applied to $\tilde{\mathbf{c}}_i$ yields a lattice vector: if we denote by $\tilde{\mathbf{C}}$ the matrix $[\tilde{\mathbf{c}}_1, \ldots, \tilde{\mathbf{c}}_m]$, we see that $\tilde{\mathbf{C}}\mathbf{z} \in \mathcal{L}(\mathbf{B})$. We complete the argument by noting that the vector $\mathbf{Cz}$ is close to both $\tilde{\mathbf{C}}\mathbf{z}$ and $\mathbf{Yz}$ (where $\mathbf{C}$ and $\mathbf{Y}$ are defined similarly to $\tilde{\mathbf{C}}$). Since the latter two vectors are lattice vectors, we obtain that $(\tilde{\mathbf{C}} - \mathbf{Y})\mathbf{z}$ is a lattice vector close to $\mathbf{0}$, as required. In slightly more detail, it turns out that the dominant part in the distance between $\tilde{\mathbf{C}}\mathbf{z}$ and $\mathbf{Yz}$ is typically that between $\mathbf{Cz}$ and $\tilde{\mathbf{C}}\mathbf{z}$. By a triangle inequality, this distance is at most $\|\mathbf{z}\|_1\|\mathbf{B}\|/n^3 \leq \|\mathbf{B}\|/n \ll \|\mathbf{B}\|/4$; hence we obtain a solution to IncGDD.

Let us indicate how to avoid the two simplifying assumptions we have made. First, IncGDD asks not for a lattice vector close to the origin but for a lattice vector close to a given target $\mathbf{t}$. This is taken care of by modifying the sampling procedure so that it outputs a pair $(\mathbf{c}, \mathbf{y})$, where $\mathbf{y}$ is close to $\mathbf{c} + \mathbf{t}'$ (instead of $\mathbf{c}$) where $\mathbf{t}'$ is now an input to the sampling procedure. By carefully choosing the vectors $\mathbf{t}'$ used in each of the m applications of the sampling procedure, we can guarantee that with some reasonable probability, the output of the reduction will be a vector close to $\mathbf{t}$. The second issue to consider is that in general, $\mathbf{S}$ is not equal to $\mathbf{B}$ and, typically, $\|\mathbf{S}\| \ll \|\mathbf{B}\|$. This is taken care of by first mapping the vectors $\mathbf{c}_i$ to the parallelepiped $\mathcal{P}(\mathbf{S})$ and then partitioning $\mathcal{P}(\mathbf{S})$ into q^n smaller parallelepipeds, as we did before with $\mathcal{P}(\mathbf{B})$. This makes the dominant distance roughly $\|\mathbf{S}\|/q$, as required. The mapping requires some care, as we want to map the uniform distribution over $\mathcal{P}(\mathbf{B})$ to the uniform distribution over $\mathcal{P}(\mathbf{S})$. Finally, let us mention that although we have ignored so far the distance between $\mathbf{Cz}$ and $\mathbf{Yz}$, this distance ends up determining the approximation factor achieved by the reduction. Because of this, in the reduction below we will make an effort to give a good bound on this distance.

We can now describe the reduction in more detail. We start with the sampling procedure $\mathcal{S}$. This procedure takes as input a lattice $\mathbf{B}$ and two additional parameters $\mathbf{t}$ and s. Provided s is not too small, the output of the procedure is a pair of vectors $(\mathbf{c}, \mathbf{y})$ with the following properties. The distribution of $\mathbf{c}$ is very close to uniform on $\mathcal{P}(\mathbf{B})$. The vector $\mathbf{y}$ is a lattice vector distributed according to a discrete Gaussian distribution with parameter s around $\mathbf{t} + \mathbf{c}$. Since s is typically small, we can think of the procedure as outputting a uniform vector $\mathbf{c} \in \mathcal{P}(\mathbf{B})$ and a lattice vector $\mathbf{y}$ close to $\mathbf{c} + \mathbf{t}$.

LEMMA 5.7 (sampling lemma). *There is a probabilistic polynomial time algorithm $\mathcal{S}(\mathbf{B}, \mathbf{t}, s)$ that on input an n-dimensional lattice $\mathbf{B} \in \mathbb{R}^{n\times n}$, a vector $\mathbf{t} \in \mathbb{R}^n$, and a real $s \geq \eta_\epsilon(\mathbf{B})$ (for some $\epsilon > 0$), outputs a pair of vectors $(\mathbf{c}, \mathbf{y}) \in \mathcal{P}(\mathbf{B}) \times \mathcal{L}(\mathbf{B})$ such that*

- *the distribution of vector $\mathbf{c}$ is within statistical distance $\Delta(\mathbf{c}, U(\mathcal{P}(\mathbf{B}))) \leq \epsilon/2$ from the uniform distribution over $\mathcal{P}(\mathbf{B})$;*
- *for any $\hat{\mathbf{c}} \in \mathcal{P}(\mathbf{B})$, the conditional distribution of $\mathbf{y}$ given $\mathbf{c} = \hat{\mathbf{c}}$ is $D_{\mathcal{L}(\mathbf{B}),s,(\mathbf{t}+\hat{\mathbf{c}})}$.*

Proof. The sampling procedure $\mathcal{S}(\mathbf{B}, \mathbf{t}, s)$ is the following:

1. Generate a noise vector $\mathbf{r}$ with probability density $D_{s,\mathbf{t}}$.
2. Output $\mathbf{c} = -\mathbf{r} \bmod \mathcal{P}(\mathbf{B})$ and $\mathbf{y} = \mathbf{r} + \mathbf{c}$.

For the first property, notice that by Lemma 4.1 and $s \geq \eta_\epsilon(\mathbf{B})$, the statistical distance between the distribution of $\mathbf{c}$ and the uniform distribution is at most

$$\begin{aligned}\Delta(\mathbf{c}, U(\mathcal{P}(\mathbf{B}))) &= \Delta(-D_{s,\mathbf{t}} \bmod \mathcal{P}(\mathbf{B}), U(\mathcal{P}(\mathbf{B})))\\ &= \Delta(D_{s,-\mathbf{t}} \bmod \mathcal{P}(\mathbf{B}), U(\mathcal{P}(\mathbf{B})))\\ &\leq \epsilon/2.\end{aligned}$$

For the second property, fix any $\hat{\mathbf{c}} \in \mathcal{P}(\mathbf{B})$. Then, by definition, the distribution of $\mathbf{r} + \hat{\mathbf{c}}$ is $D_{s,\mathbf{t}+\hat{\mathbf{c}}}$. Conditioning on $\mathbf{c} = \hat{\mathbf{c}}$ is the same as conditioning on $\mathbf{r} + \hat{\mathbf{c}} \in \mathcal{L}(\mathbf{B})$. As discussed in section 2, the distribution of $\mathbf{r} + \hat{\mathbf{c}}$ conditioned on $\mathbf{r} + \hat{\mathbf{c}} \in \mathcal{L}(\mathbf{B})$ is $D_{\mathcal{L}(\mathbf{B}),s,\mathbf{t}+\hat{\mathbf{c}}}$, as required. □

Next, we describe a procedure which we call the combining procedure $\mathcal{A}$. This procedure is the heart of the worst-case to average-case reduction. It maps the vectors $\mathbf{c}_i$ to vectors in the parallelepiped $\mathcal{P}(\mathbf{S})$ and group elements $\mathbf{a}_i$, and then applies the oracle $\mathcal{F}$. At first reading, we suggest skipping the proof of the lemma and jumping directly to Theorem 5.9.

LEMMA 5.8 (combining procedure). *There is a probabilistic polynomial time oracle algorithm $\mathcal{A}^{\mathcal{F}}(\mathbf{B}, \mathbf{S}, \mathbf{C}, q)$ that on input an n-dimensional lattice $\mathbf{B} \in \mathbb{R}^{n\times n}$, a full-rank sublattice $\mathbf{S} \subset \mathcal{L}(\mathbf{B})$, m vectors $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_m] \in \mathcal{P}(\mathbf{B})^m$, and a positive integer q, makes a single oracle call $\mathcal{F}(\mathbf{A}) = \mathbf{z}$ (with $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$) and outputs a vector $\mathbf{x} \in \mathbb{R}^n$ such that*

- *if the input matrix $\mathbf{C} \in \mathcal{P}(\mathbf{B})^m$ is distributed uniformly at random, then the query matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ is also uniformly distributed;*
- *if the oracle's answer $\mathbf{z} = \mathcal{F}(\mathbf{A})$ is in $\Lambda_q(\mathbf{A})$, then the output vector $\mathbf{x}$ belongs to the lattice $\mathcal{L}(\mathbf{B})$;*
- *the distance between the output vector $\mathbf{x}$ and $\mathbf{Cz}$ is at most $\sqrt{m}n\|\mathbf{S}\| \cdot \|\mathbf{z}\|/q$.*

Proof. The procedure $\mathcal{A}^{\mathcal{F}}(\mathbf{B}, \mathbf{S}, \mathbf{C}, q)$ is the following (see also the box labelled $\mathcal{A}^{\mathcal{F}}$ in Figure 2).

1. Generate m uniformly random lattice vectors $\mathbf{v}_i \in \mathcal{L}(\mathbf{B}) \bmod \mathcal{P}(\mathbf{S})$ (this can be done using standard techniques; see for example [21, Proposition 2.9]).
2. Define the matrix $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_m]$, where $\mathbf{w}_i = \mathbf{v}_i + \mathbf{c}_i \bmod \mathcal{P}(\mathbf{S})$ for all $i = 1, \dots, m$.
3. Define the query $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_m]$, where $\mathbf{a}_i = \lfloor q \cdot \mathbf{S}^{-1}\mathbf{w}_i \rfloor \in \mathbb{Z}_q^n$ for all $i = 1, \dots, m$.
4. Invoke the oracle $\mathcal{F}$ on input $\mathbf{A}$ to obtain an integer vector $\mathbf{z} = \mathcal{F}(\mathbf{A})$.
5. Output the vector $\mathbf{x} = (\mathbf{C} - \mathbf{W} + \mathbf{S}\mathbf{A}/q)\mathbf{z}$.

We now prove the first property. We start by noting that if $\mathbf{c}$ is uniformly distributed in $\mathcal{P}(\mathbf{B})$ and $\mathbf{v}$ is chosen uniformly from the vectors in $\mathcal{L}(\mathbf{B}) \bmod \mathcal{P}(\mathbf{S})$, then $\mathbf{c} + \mathbf{v} \bmod \mathcal{P}(\mathbf{S})$ is distributed uniformly in $\mathcal{P}(\mathbf{S})$. This holds since the sets $(\mathbf{v} + \mathcal{P}(\mathbf{B})) \bmod \mathcal{P}(\mathbf{S})$ for all $\mathbf{v} \in \mathcal{L}(\mathbf{B}) \bmod \mathcal{P}(\mathbf{S})$ form a partition of $\mathcal{P}(\mathbf{S})$ into sets of equal volume. Thus, we see that if $\mathbf{C} \in \mathcal{P}(\mathbf{B})^m$ is distributed uniformly, then $\mathbf{W}$ is distributed uniformly in $\mathcal{P}(\mathbf{S})^m$. From this, it easily follows that $\mathbf{A}$ is distributed uniformly in $\mathbb{Z}_q^{n \times m}$, as required.

We now prove the second property. Assume $\mathbf{z} \in \Lambda_q(\mathbf{A})$, and consider the output vector

$$\mathbf{x} = (\mathbf{C} - \mathbf{W} + \mathbf{S}\mathbf{A}/q)\mathbf{z} = \sum_{i=1}^{m} (\mathbf{c}_i - \mathbf{w}_i) z_i + \mathbf{S}(\mathbf{A}\mathbf{z}/q).$$

Notice that for any $i = 1, \dots, m$ the vector

$$\mathbf{c}_i - \mathbf{w}_i = ((\mathbf{c}_i + \mathbf{v}_i) - \mathbf{w}_i) - \mathbf{v}_i$$

belongs to the lattice $\mathcal{L}(\mathbf{B})$ because $\mathbf{c}_i + \mathbf{v}_i \equiv \mathbf{w}_i$ modulo $\mathcal{L}(\mathbf{S}) \subseteq \mathcal{L}(\mathbf{B})$ and $\mathbf{v}_i \in \mathcal{L}(\mathbf{B})$. Also, $\mathbf{A}\mathbf{z}/q$ is an integer vector because $\mathbf{A}\mathbf{z} = \mathbf{0} \bmod q$. This proves that $\mathbf{x}$ belongs to the lattice $\mathcal{L}(\mathbf{B})$ because it is an integer linear combination of lattice vectors.

For the third property, we bound the distance between $\mathbf{x}$ and $\mathbf{C}\mathbf{z}$ as follows:

$$\begin{aligned} \|\mathbf{x} - \mathbf{C}\mathbf{z}\| &= \left\| \sum_{i=1}^{m} (\mathbf{w}_i - (\mathbf{S}/q)\mathbf{a}_i) z_i \right\| \\ &= \frac{1}{q} \left\| \mathbf{S} \sum_{i=1}^{m} (\mathbf{u}_i - \lfloor \mathbf{u}_i \rfloor) z_i \right\|, \end{aligned}$$

where $\mathbf{u}_i = q\mathbf{S}^{-1}\mathbf{w}_i$. Since for each i, all entries of $\mathbf{u}_i - \lfloor \mathbf{u}_i \rfloor$ are bounded by 1, the vector $\sum_{i=1}^{m} (\mathbf{u}_i - \lfloor \mathbf{u}_i \rfloor) z_i$ has all entries bounded by $\sum_i |z_i| \le \sqrt{m}\|\mathbf{z}\|$. It follows by the triangle inequality that

$$\left\| \mathbf{S} \sum_{i=1}^{m} (\mathbf{u}_i - \lfloor \mathbf{u}_i \rfloor) z_i \right\| \le n\sqrt{m}\|\mathbf{z}\|\|\mathbf{S}\|$$

and $\|\mathbf{x} - \mathbf{C}\mathbf{z}\| \le n\sqrt{m}\|\mathbf{z}\|\|\mathbf{S}\|/q$. □

We are now ready to reduce IncGDD to SIS using the procedures $\mathcal{S}$ and $\mathcal{A}$ from the previous lemmas. In Theorem 5.9, all parameters are implicitly assumed to have

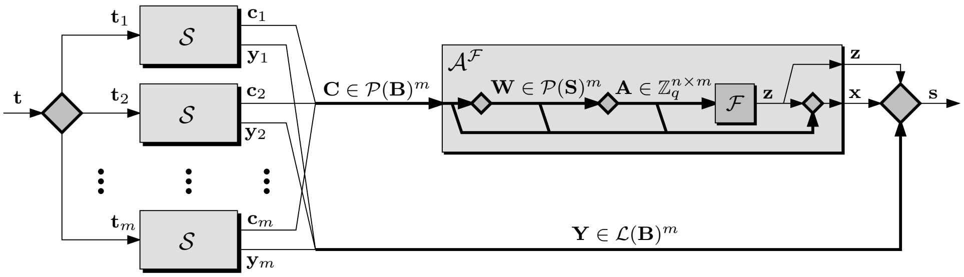


FIG. 2. *A diagram of the worst-case to average-case reduction.*

bit-size polynomial in the security parameter, i.e., $g(n), q(n) = 2^{n^{O(1)}}$. In fact, in most of the applications of this theorem considered in this paper, the parameters will be smaller, typically polynomial in n. For example, one can take, say, $g = 8$ to be a constant, $q(n) = n^3$ (or any other sufficiently large polynomial), $m(n) = n \log n$, and $\beta(n) = \sqrt{m(n)} q(n)^{n/m(n)} = 8\sqrt{n \log n}$ the bound from Lemma 5.2 so that $\mathrm{SIS}_{q,m,\beta}$ is guaranteed to admit a solution. It is easy to check that $q(n)$ satisfies the condition in the theorem below, yielding approximation factor $\gamma(n) = \beta(n)\sqrt{n} = 8n\sqrt{\log n} = O(n\sqrt{\log n})$.

THEOREM 5.9. *For any function $g(n) > 0$, polynomially bounded functions $m(n), \beta(n) = n^{O(1)}$, negligible function $\epsilon(n) = n^{-\omega(1)}$, and $q(n) \geq g(n) n \sqrt{m(n)} \beta(n)$, there is a probabilistic polynomial time reduction from solving $\mathrm{INCGDD}^{\eta_\epsilon}_{\gamma,g}$ for $\gamma(n) = \beta(n)\sqrt{n}$ on n-dimensional instances in the worst case to solving $\mathrm{SIS}_{q,m,\beta}$ on the average with nonnegligible probability.*

Proof. Many of our parameters depend on n; for notational convenience, we often omit this dependency and write m instead of $m(n)$, and similarly for $\gamma, \beta, g, q, \epsilon$, and δ. Let $\mathcal{F}$ be an oracle that solves $\mathrm{SIS}_{q,m,\beta}$ on the average. In other words, we assume that on input a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, the oracle call $\mathcal{F}(\mathbf{A})$ returns a nonzero vector $\mathbf{z} \in \Lambda_q(\mathbf{A})$ of length at most $\|\mathbf{z}\| \leq \beta$ with some nonnegligible probability $\delta(n) = n^{-O(1)}$.

On input an $\mathrm{INCGDD}^{\eta_\epsilon}_{\gamma,g}$ instance $(\mathbf{B}, \mathbf{S}, \mathbf{t}, r)$, the reduction performs the following operations (see Figure 2 for a high-level overview). The goal of the first step is to "guess" how to choose the vectors $\mathbf{t}_i$ given to the sampling procedure. As we shall see later, with reasonable probability, this guess causes the output of the reduction to be a lattice vector close to $\mathbf{t}$.

1. Pick an index $j \in M = \{1, \ldots, m\}$ and integer $\alpha \in B = \{-\beta, \ldots, -1, 1, \ldots, \beta\}$ uniformly at random. For each $i \in M$, define the vector
$$\mathbf{t}_i = \begin{cases} -\mathbf{t}/\alpha & \text{if } i = j, \\ \mathbf{0} & \text{otherwise.} \end{cases}$$
2. For each $i = 1, \ldots, m$, compute the pair
$$(\mathbf{c}_i, \mathbf{y}_i) = \mathcal{S}(\mathbf{B}, \mathbf{t}_i, 2r/\gamma)$$
using the sampling procedure of Lemma 5.7, each time with independent randomness.
3. Define the matrices $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_m]$ and $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_m]$.

4. Finally, call the combining algorithm $\mathcal{A}^{\mathcal{F}}(\mathbf{B}, \mathbf{S}, \mathbf{C}, q) = \mathbf{x}$ using $\mathcal{F}$ as an oracle, and output the vector $\mathbf{s} = \mathbf{x} - \mathbf{Y}\mathbf{z}$, where $\mathbf{z} = \mathcal{F}(\mathbf{A})$ is the answer returned by $\mathcal{F}$ to $\mathcal{A}$'s query $\mathbf{A}$.

We want to bound from below the success probability of the reduction, i.e., the probability that the output vector $\mathbf{s}$ satisfies $\mathbf{s} \in \mathcal{L}(\mathbf{B})$ and $\|\mathbf{s} - \mathbf{t}\| \leq (\|\mathbf{S}\|/g) + r$. We start by finding some "good" values for j and α. To this end, consider the output $\mathbf{z}' = \mathcal{F}(\mathbf{A}')$ of the oracle on a *uniformly random* input $\mathbf{A}' \in \mathbb{Z}_q^{n \times m}$. For each $j' \in M, \alpha' \in B$ denote by $\delta_{j',\alpha'}$ the probability that this output $\mathbf{z}'$ satisfies

$$(z'_{j'} = \alpha') \wedge (\mathbf{z}' \in \Lambda_q(\mathbf{A}') \setminus \{\mathbf{0}\}) \wedge (\|\mathbf{z}'\| \leq \beta).$$

Since any nonzero vector $\mathbf{z} \in \mathbb{Z}^m$ with $\|\mathbf{z}\| \leq \beta$ must have at least one coordinate in the set B,

$$\sum_{j' \in M, \alpha' \in B} \delta_{j',\alpha'} \geq \delta.$$

Hence, there must exist some $j' \in M, \alpha' \in B$ for which $\delta_{j',\alpha'} \geq \delta/2\beta m$ and is thus nonnegligible. In the rest of the proof we consider the execution of the reduction for any fixed values of j and α and show that its success probability is at least $\delta_{j,\alpha}/3$, up to a negligible term. In particular, for $j = j', \alpha = \alpha'$, the reduction is successful with a nonnegligible probability. This would complete the proof since the event $j = j', \alpha = \alpha'$ happens with probability $1/2\beta m$, which is nonnegligible.

So in the rest of the proof, fix some values of j and α. Let H be the event

$$H \overset{def}{=} (z_j = \alpha) \wedge (\mathbf{z} \in \Lambda_q(\mathbf{A}) \setminus \{\mathbf{0}\}) \wedge (\|\mathbf{z}\| \leq \beta), \tag{12}$$

where $\mathbf{A}$ and $\mathbf{z}$ are the random variables that appear in the reduction. In other words, H is the event that oracle $\mathcal{F}$ is successful and the values j and α satisfy the desired condition $z_j = \alpha$. We now show that the input $\mathbf{C}$ to the combining procedure is very close to uniform and that this implies that H happens with probability very close to $\delta_{j,\alpha}$. We will later see that conditioned on H, the reduction succeeds with probability $1/3$.

Each column $\mathbf{c}_i$ of $\mathbf{C}$ is chosen by running the sampling algorithm $\mathcal{S}(\mathbf{B}, \mathbf{t}_i, s) = (\mathbf{c}_i, \mathbf{y}_i)$ for some vector $\mathbf{t}_i \in \mathbb{R}^n$ and $s = 2r/\gamma > 2\eta_\epsilon(\mathbf{B})$. It follows from Lemma 5.7 that for each i, the statistical distance between $\mathbf{c}_i$ and the uniform distribution over $\mathcal{P}(\mathbf{B})$ is at most $\epsilon/2$. Since the vectors $\mathbf{c}_i$ are independent, we have

$$\Delta(\mathbf{C}, U(\mathcal{P}(\mathbf{B})^m)) \leq \sum_{i=1}^{m} \Delta(\mathbf{c}_i, U(\mathcal{P}(\mathbf{B}))) \leq \epsilon m/2. \tag{13}$$

By Lemma 5.8, on input a uniformly random matrix $\mathbf{C}' \in \mathcal{P}(\mathbf{B})^m$, the distribution of the query $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ asked by $\mathcal{A}^{\mathcal{F}}(\mathbf{B}, \mathbf{S}, \mathbf{C}', q)$ is also uniform. Therefore, on a uniform input $\mathbf{C}'$, the vector $\mathbf{z} = \mathcal{F}(\mathbf{A})$ obtained by the combining procedure satisfies the conditions in (12) with probability $\delta_{j,\alpha}$. By (13) and the properties of the statistical distance, it follows that the event H holds with probability at least $\delta_{j,\alpha} - \epsilon m/2$, which is $\delta_{j,\alpha}$ up to a negligible term.

To complete the proof, we show that the success probability of the reduction conditioned on H is at least $1/3$. We in fact show the stronger fact that the success probability of the reduction is at least $1/3$ conditioned on any fixed values of $\mathbf{C}$, the oracle query $\mathbf{A}$, and the answer $\mathbf{z} = \mathcal{F}(\mathbf{A})$ for which H is satisfied. So in the following, fix some j, α, $\mathbf{C}$, $\mathbf{A}$, and $\mathbf{z}$ for which $z_j = \alpha$, $\mathbf{z} \in \Lambda_q(\mathbf{A}) \setminus \{\mathbf{0}\}$ and $\|\mathbf{z}\| \leq \beta$. In particular, if we define $\mathbf{T} = [\mathbf{t}_1, \ldots, \mathbf{t}_m]$, we get $\mathbf{Tz} = -\mathbf{t}$. We know from Lemma 5.8 that the vector $\mathbf{x} = \mathcal{A}^{\mathcal{F}}(\mathbf{B}, \mathbf{S}, \mathbf{C}, q)$ belongs to the lattice $\mathcal{L}(\mathbf{B})$ and is within distance $n\sqrt{m}\|\mathbf{z}\|\|\mathbf{S}\|/q \leq \|\mathbf{S}\|/g$ from $\mathbf{Cz}$. We also know from Lemma 5.7 that the vectors $\mathbf{y}_i$ are distributed independently according to $D_{\mathcal{L}(\mathbf{B}),s,(\mathbf{c}_i+\mathbf{t}_i)}$, where $s = 2r/\gamma > 2\eta_\epsilon$. Since $\mathbf{x}$ and $\mathbf{y}_1, \ldots, \mathbf{y}_m$ are all lattice vectors and $\mathbf{z} \in \mathbb{Z}^m$, it also holds that $\mathbf{s} = \mathbf{x} - \mathbf{Yz}$ belongs to the lattice $\mathcal{L}(\mathbf{B})$. We need to compute the probability that $\|\mathbf{s} - \mathbf{t}\| \leq (\|\mathbf{S}\|/g) + r$. By the triangle inequality,

$$\|\mathbf{s} - \mathbf{t}\| \leq \|\mathbf{x} - \mathbf{Cz}\| + \|(\mathbf{C} - \mathbf{Y})\mathbf{z} - \mathbf{t}\| \leq \frac{\|\mathbf{S}\|}{g} + \|(\mathbf{Y} - (\mathbf{C} + \mathbf{T}))\mathbf{z}\|.$$

Thus, all we have to do is to bound the probability that $\|(\mathbf{Y} - (\mathbf{C} + \mathbf{T}))\mathbf{z}\| \leq r$. By Lemma 4.3, since each vector $\mathbf{y}_i$ is distributed according to $D_{\mathcal{L}(\mathbf{B}),s,(\mathbf{c}_i+\mathbf{t}_i)}$, we have

$$\operatorname{Exp}[\|\mathbf{y}_i - (\mathbf{c}_i + \mathbf{t}_i)\|^2] \leq \left(\frac{1}{2\pi} + \frac{\epsilon}{1-\epsilon}\right) s^2 n$$

and

$$\|\operatorname{Exp}[\mathbf{y}_i - (\mathbf{c}_i + \mathbf{t}_i)]\|^2 \leq \left(\frac{\epsilon}{1-\epsilon}\right)^2 s^2 n.$$

Since the vectors $\mathbf{y}_1, \ldots, \mathbf{y}_m$ are chosen independently, we can apply Lemma 2.11 and get

$$\begin{aligned} \operatorname{Exp}\left[\left\|\sum_{i=1}^{m}(\mathbf{y}_i - (\mathbf{c}_i + \mathbf{t}_i))z_i\right\|^2\right] &\leq \left(\frac{1}{2\pi} + \frac{\epsilon}{1-\epsilon} + \left(\frac{\epsilon}{1-\epsilon}\right)^2 m\right) \|\mathbf{z}\|^2 s^2 n \\ &\leq \frac{\|\mathbf{z}\|^2 s^2 n}{6} \end{aligned}$$

for all sufficiently large n. Finally, using $\|\mathbf{z}\| \leq \beta$, $s = 2r/\gamma$, and $\gamma = \beta\sqrt{n}$, we get

$$\operatorname{Exp}[\|(\mathbf{Y} - (\mathbf{C} + \mathbf{T}))\mathbf{z}\|^2] \leq \frac{\|\mathbf{z}\|^2 s^2 n}{6} \leq \frac{2}{3} r^2$$

and, by Markov inequality, we get

$$\Pr\{\|(\mathbf{Y} - (\mathbf{C} + \mathbf{T}))\mathbf{z}\| > r\} = \Pr\{\|(\mathbf{Y} - (\mathbf{C} + \mathbf{T}))\mathbf{z}\|^2 > r^2\} \leq \frac{2}{3}.$$

This proves that the conditional probability of $\|\mathbf{s} - \mathbf{t}\| \leq (\|\mathbf{S}\|/g) + r$ is at least $1/3$ and completes the proof. □

5.3. Other worst-case problems. In section 5.2 we have shown that solving SIS on the average is at least as hard as solving IncGDD in the worst case. In this section we prove that solving SIS on the average is at least as hard as solving many other standard lattice problems, such as SIVP and GapCRP. These results are obtained as corollaries to Theorem 5.9 using straightforward worst-case to worst-case reductions among lattice problems. We now describe three such reductions. The first two are taken from [22] and for completeness, we include a sketch of their proof.

Lemma 5.10. *For any $\gamma(n) \geq 1$ and any ϕ, there exists a reduction from* $\mathrm{GIVP}^{\phi}_{8\gamma}$ *to* $\mathrm{IncGDD}^{\phi}_{\gamma,8}$.

Proof. Given a basis $\mathbf{B}$, our goal is to construct a set of n linearly independent vectors $\mathbf{S}$ of length $\|\mathbf{S}\| \leq 8\gamma(n)\phi(\mathbf{B})$. We do this by an iterative process. Initially, we set $\mathbf{S} = \mathbf{B}$. At each step, we identify the longest vector in $\mathbf{S}$, say $\mathbf{s}_i$. We then take $\mathbf{t}$ to be a vector orthogonal to $\mathbf{s}_1, \ldots, \mathbf{s}_{i-1}, \mathbf{s}_{i+1}, \ldots, \mathbf{s}_n$ of length $\|\mathbf{S}\|/2$. We apply the IncGDD oracle with the instance $(\mathbf{B}, \mathbf{S}, \mathbf{t}, \|\mathbf{S}\|/8)$. If it fails, we abort and output $\mathbf{S}$. Otherwise, we obtain a lattice vector $\mathbf{u}$ within distance at most $(\|\mathbf{S}\|/8) + \|\mathbf{S}\|/8 = \|\mathbf{S}\|/4$ from $\mathbf{t}$. Notice that $\|\mathbf{u}\| \leq 3\|\mathbf{S}\|/4$ and that it is linearly independent from the vectors in $\mathbf{s}_1, \ldots, \mathbf{s}_{i-1}, \mathbf{s}_{i+1}, \ldots, \mathbf{s}_n$. We then replace $\mathbf{s}_i$ with $\mathbf{u}$ and repeat the process.

Notice that when the oracle call fails, it must be the case that $\|\mathbf{S}\|/8 \leq \gamma(n)\phi(\mathbf{B})$, and hence $\|\mathbf{S}\| \leq 8\gamma(n)\phi(\mathbf{B})$, as required. Moreover, it is not difficult to argue that this procedure terminates after a polynomial number of steps. For instance, one can note that $\log \Pi_i \|\mathbf{s}_i\|$ decreases by a constant at each step, and that its initial value is polynomial in the input size. □

LEMMA 5.11. *For any $\gamma(n) \geq 1$ and any ϕ, there exists a reduction from $\mathrm{GDD}_{3\gamma}^{\phi}$ to $\mathrm{IncGDD}_{\gamma,8}^{\phi}$.*

Proof. Given a basis $\mathbf{B}$ and a vector $\mathbf{t}$, our goal is to find a lattice vector within distance $3\gamma(n)\phi(\mathbf{B})$ of $\mathbf{t}$. First, we apply the reduction in Lemma 5.10 to obtain a set $\mathbf{S}$ of n linearly independent vectors of length at most $\|\mathbf{S}\| \leq 8\gamma(n)\phi(\mathbf{B})$. We then search for a value r for which an oracle call with $(\mathbf{B}, \mathbf{S}, \mathbf{t}, r/2)$ fails but an oracle call with $(\mathbf{B}, \mathbf{S}, \mathbf{t}, r)$ succeeds. Since the former oracle call fails, it must be the case that $r \leq 2\gamma(n)\phi(\mathbf{B})$. The latter oracle call yields a lattice vector within distance $\|\mathbf{S}\|/8 + r \leq \gamma(n)\phi(\mathbf{B}) + 2\gamma(n)\phi(\mathbf{B}) = 3\gamma(n)\phi(\mathbf{B})$, as required. □

LEMMA 5.12. *For any $\gamma(n)$, there exists a randomized reduction from GapCRP_{γ} to $\mathrm{GDD}_{\gamma/4}^{\lambda_n}$.*

Proof. Let $(\mathbf{B}, d)$ be an instance of GapCRP_{γ}. The reduction picks a point $\mathbf{t} \in \mathcal{P}(\mathbf{B})$ uniformly at random and then calls the $\mathrm{GDD}_{\gamma/4}^{\lambda_n}$ oracle with the instance $(\mathbf{B}, \mathbf{t})$ to obtain a lattice vector $\mathbf{x} \in \mathcal{L}(\mathbf{B})$ within distance $(\gamma/4)\lambda_n(\mathbf{B})$ from the target $\mathbf{t}$. If $\|\mathbf{t} - \mathbf{x}\| \leq \gamma d/2$, then we accept; otherwise we reject. If $\nu(\mathbf{B}) \leq d$, then

$$\|\mathbf{t} - \mathbf{x}\| \leq \gamma\lambda_n(\mathbf{B})/4 \leq \gamma\nu(\mathbf{B})/2 \leq \gamma d/2,$$

where we use that for any lattice Λ, $\nu(\Lambda) \geq \lambda_n(\Lambda)/2$ [23, Theorem 7.9]. So, YES instances are always accepted. On the other hand, assume that $\nu(\mathbf{B}) > \gamma d$. In [14] it is shown that a random $\mathbf{t}$ chosen as above satisfies $\mathrm{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B})) \geq \nu(\mathbf{B})/2$ with probability at least 1/2. Hence, $\mathrm{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B})) > \gamma d/2$ with probability 1/2, and NO instances are rejected with probability 1/2. □

By combining these reductions with Theorem 5.9, we obtain several useful corollaries. The first relates GIVP to SIS and we give it here in its most general form, as it will later be used in the GapSVP reduction.

COROLLARY 5.13. *For any polynomially bounded functions $\beta(n), m(n) = n^{O(1)}$, any negligible function $\epsilon(n)$, and any $q(n) \geq 8n\sqrt{m(n)}\beta(n)$, there is a probabilistic polynomial time reduction from solving $\mathrm{GIVP}_{\gamma}^{\eta_\epsilon}$ in the worst case with $\gamma(n) = 8\beta(n)\sqrt{n}$ to solving $\mathrm{SIS}_{q,m,\beta}$ on the average with nonnegligible probability.*

For the remaining corollaries, it is helpful to specialize Theorem 5.9 to the $\mathrm{SIS}_{q,m}$ problem where solutions are guaranteed to exist. This is done by choosing $\beta(n) = \sqrt{m(n)}q(n)^{n/m(n)}$. Observe that for this value of β, we have that the condition $q(n) \geq g(n)n\sqrt{m(n)}\beta(n)$ is equivalent to $q(n) \geq (g(n)nm(n))^{1+n/(m(n)-n)}$.

COROLLARY 5.14. *For any function $g(n) > 0$, polynomially bounded function $m(n) = n^{O(1)}$, negligible function $\epsilon(n) = n^{-\omega(1)}$, and $q(n) \geq (g(n)nm(n))^{1+n/(m(n)-n)}$,* *there is a probabilistic polynomial time reduction from solving* $\text{IncGDD}_{\gamma,g}^{\eta_\epsilon}$ *for* $\gamma(n) = \sqrt{nm(n)} \cdot q(n)^{n/m(n)}$ *on n-dimensional instances in the worst case to solving* $\text{SIS}_{q,m}$ *on the average with nonnegligible probability.*

We continue with some other connections. For simplicity, from now on we consider a specific choice of parameters. Other choices can be handled similarly.

Theorem 5.15. *For any* $m(n) = \Theta(n \log n)$*, there exists some* $q(n) = O(n^2 \log n)$ *and* $\gamma(n) = O(n\sqrt{\log n})$ *such that for any negligible function* $\epsilon(n)$*, solving* $\text{SIS}_{q,m}$ *on the average with nonnegligible probability is at least as hard as solving any of the following worst-case problems:*

- $\text{GIVP}_\gamma^{\eta_\epsilon}$*,*
- $\text{GDD}_\gamma^{\eta_\epsilon}$*.*

Proof. Notice that for any $m(n) = \Theta(n \log n)$ there exists a $q(n) = O(n^2 \log n)$ that satisfies the conditions in Corollary 5.14 with $g(n) = 8$ a constant. This yields a solution to $\text{IncGDD}_{\gamma,8}^{\eta_\epsilon}$ for some $\gamma(n) = O(n\sqrt{\log n})$. It remains to apply Lemmas 5.10 and 5.11. □

Theorem 5.16. *For any* $m(n) = \Theta(n \log n)$ *there exists a* $q(n) = O(n^2 \log n)$ *such that for any function* $\gamma(n) = \omega(n \log n)$*, solving* $\text{SIS}_{q,m}$ *on the average with nonnegligible probability is at least as hard as solving any of the following worst-case problems:*

- SIVP_γ *(or equivalently,* $\text{GIVP}_\gamma^{\lambda_n}$*),*
- $\text{GDD}_\gamma^{\lambda_n}$*,*
- GapCRP_γ*.*

Proof. Let $\alpha(n)$ be any function (e.g., $\alpha(n) = \sqrt{\gamma(n)/n \log n}$) such that $\alpha(n) = \omega(1)$ and $\gamma(n) = \omega(\alpha(n) n \log n)$. By Lemma 3.3, there exists a negligible $\epsilon(n)$ for which $\eta_\epsilon(\Lambda) \le \alpha(n)\sqrt{\log n}\lambda_n(\Lambda)$ holds for any lattice. Hence, the first two claims follow from Theorem 5.15 for some approximation factor $O(\alpha(n) n \log n) < \gamma(n)$. The third claim follows from the second claim together with Lemma 5.12. □

We complete this section with a discussion of *nonadaptive* reductions. These are reductions in which the oracle queries do not depend on the answers to previous queries and hence can be performed all at once. It is known that unless the polynomial hierarchy collapses, no average-case problem can be shown to be NP-hard under nonadaptive reductions. See [7] and references therein for a more accurate description of these results. Here, we observe that our reductions can be made nonadaptive with only a slight worsening of the approximation factors obtained.

Lemma 5.17. *For any functions* $g(n), \gamma(n)$ *such that* $\gamma(n) < n^c$ *for some* $c > 0$*, there exists a nonadaptive reduction from* $\text{GDD}_{\gamma'}^{\lambda_n}$ *to* $\text{IncGDD}_{\gamma,g}^{\lambda_n}$*, where* $\gamma'(n) = (2^n/g(n)) + 2\gamma(n)$*.*

Proof. Given a lattice $\mathbf{B}$ and a target $\mathbf{t}$, we want to find a lattice vector close to $\mathbf{t}$. Using the LLL lattice reduction algorithm [17], we can efficiently compute a basis $\mathbf{S}$ of $\mathcal{L}(\mathbf{B})$ such that $\|\mathbf{S}\| \le 2^n \lambda_n(\mathbf{B})$. Let $\tilde{\lambda}_n = \|\mathbf{S}\|/2^n$ and notice that $\tilde{\lambda}_n \le \lambda_n(\mathbf{B}) \le 2^n \tilde{\lambda}_n$. The reduction then calls the IncGDD oracle on input $(\mathbf{B}, \mathbf{S}, \mathbf{t}, 2^i \cdot \tilde{\lambda}_n)$ for $i = 0, 1, \dots, \lceil n + c \log n \rceil$ and outputs the lattice vector closest to $\mathbf{t}$ among the vectors returned.

Let i be the smallest index such that $2^i \tilde{\lambda}_n > \gamma(n)\lambda_n(\mathbf{B})$. Such an i exists since $2^{n+c\log n}\tilde{\lambda}_n = n^c \cdot 2^n \tilde{\lambda}_n > \gamma(n)\lambda_n(\mathbf{B})$. Notice that $2^i \tilde{\lambda}_n \le 2\gamma(n)\lambda_n(\mathbf{B})$. It follows that the lattice vector returned by the IncGDD oracle on input $(\mathbf{B}, \mathbf{S}, \mathbf{t}, 2^i \tilde{\lambda}_n)$ is within distance

$$\frac{\|\mathbf{S}\|}{g(n)} + 2^i \tilde{\lambda}_n \le \frac{2^n \lambda_n(\mathbf{B})}{g(n)} + 2\gamma(n)\lambda_n(\mathbf{B}) = \left(\frac{2^n}{g(n)} + 2\gamma(n)\right)\lambda_n(\mathbf{B})$$

from the target $\mathbf{t}$, as required. □

LEMMA 5.18. *For any $\gamma(n)$, there exists a nonadaptive reduction from* SIVP_γ *to* $\mathrm{GDD}^{\lambda_n}_{\gamma/4\sqrt{n}}$.

Proof. Let $\mathbf{B}$ be some instance of SIVP_γ. Using the LLL lattice reduction algorithm [17], we can efficiently compute a basis $\mathbf{S}$ of the same lattice such that $\|\mathbf{S}\| \le 2^n \lambda_n(\mathbf{B})$. Notice that if $\gamma \ge 2^n$, we can simply output $\mathbf{S}$, so in the following assume $\gamma < 2^n$. Let $\tilde{\lambda}_n = 2^{-n-1}\|\mathbf{S}\|$ and notice that $2\tilde{\lambda}_n \le \lambda_n(\mathbf{B}) \le 2^{n+1}\tilde{\lambda}_n$. Let $\mathbf{e}_1, \dots, \mathbf{e}_n$ be some orthonormal set of vectors. The reduction calls the $\mathrm{GDD}^{\lambda_n}_{\gamma/4\sqrt{n}}$ oracle on input $(\mathbf{B}, 2^i\tilde{\lambda}_n\mathbf{e}_j)$ for $i = 0, \dots, 2n-1$ and $j = 1, \dots, n$. For $i = 0, \dots, 2n-1$, let $\mathbf{S}_i$ denote the set of n vectors returned by the oracle on queries corresponding to i. Among the sets $\mathbf{S}_i$ that contain n linearly independent vectors, the reduction outputs the one that minimizes $\|\mathbf{S}_i\|$. We need to prove that there exists an index i such that the vectors $\mathbf{S}_i$ are linearly independent and $\|\mathbf{S}_i\| \le \gamma\lambda_n(\mathbf{B})$.

Let $i \in \{0, \dots, 2n-1\}$ be the smallest index such that $2^i\tilde{\lambda}_n > \gamma\lambda_n(\mathbf{B})/4$. Notice that such an i exists and that $2^i\tilde{\lambda}_n \le \gamma\lambda_n(\mathbf{B})/2$. Each column of $\mathbf{S}_i$ is within distance $\gamma\lambda_n(\mathbf{B})/4\sqrt{n}$ from the corresponding vector $2^i\tilde{\lambda}_n\mathbf{e}_j$. Since the length of the latter is strictly greater than $\gamma\lambda_n(\mathbf{B})/4$, it follows that the columns of $\mathbf{S}_i$ are linearly independent (see, e.g., [20]). Finally, by the triangle inequality, each vector in $\mathbf{S}_i$ has length at most

$$2^i\tilde{\lambda}_n + \gamma\lambda_n(\mathbf{B})/4\sqrt{n} \le \gamma\lambda_n(\mathbf{B})/2 + \gamma\lambda_n(\mathbf{B})/4\sqrt{n} \le \gamma\lambda_n(\mathbf{B}). \qquad \square$$

THEOREM 5.19. *There exist functions $q(n) = 2^{O(n)}$ and $m(n) = n^{O(1)}$ such that for any function $\alpha(n) = \omega(\sqrt{\log n})$, solving* $\mathrm{SIS}_{q,m}$ *on the average with nonnegligible probability is at least as hard (via nonadaptive reductions) as solving any of the following worst-case problems:*

- $\mathrm{GDD}^{\lambda_n}_\gamma$ *for some $\gamma(n) = O(n^{1.5}\alpha(n))$,*
- GAPCRP_γ *for some $\gamma(n) = O(n^{1.5}\alpha(n))$,*
- SIVP_γ *for some $\gamma(n) = O(n^2\alpha(n))$.*

Proof. By Lemma 3.3, we can choose a negligible function $\epsilon(n)$ such that for any lattice Λ, $\eta_\epsilon(\Lambda) \le \alpha(n)\lambda_n(\Lambda)$. Let $q(n) = n^3 2^n$, $m(n) = n^2$, and $g(n) = 2^n/4$. Notice that this choice satisfies the hypothesis in Corollary 5.14. Moreover, notice that the reduction in Theorem 5.9 is nonadaptive since it makes only one oracle query. Therefore, by Corollary 5.14, there is a nonadaptive reduction from solving worst-case instances of $\mathrm{INCGDD}^{\eta_\epsilon}_{\gamma,g}$ with $\gamma(n) \le 4n^{1.5}$ to solving $\mathrm{SIS}_{q,m}$ on the average with nonnegligible probability. By our choice of ϵ, this is also a reduction from $\mathrm{INCGDD}^{\lambda_n}_{\gamma',g}$, where $\gamma'(n) = \gamma(n)\alpha(n)$.

The first claim follows from Lemma 5.17. The second claim follows directly from the first together with Lemma 5.12. The only thing to notice is that the reduction in that lemma is nonadaptive since it makes only one oracle call. The third follows similarly with the use of Lemma 5.18. □

5.4. Shortest vector problem. In this subsection we reduce GAPSVP to SIS′. Let us first recall Hoeffding's inequality [15], which states the following. Let $X_1, \dots, X_N$ be N independent random variables, such that for all i, $X_i \in [a, b]$. Then $S_N = \sum_i X_i$ satisfies

$$\Pr\{S_N \ge \mathrm{Exp}[S_N] + N\epsilon\} \le e^{-N\epsilon^2/(b-a)^2}. \tag{14}$$

We will also need the following lemma from [1]. For completeness, we include its proof in the appendix.

Lemma 5.20 (see [1, Lemma 6.2]). *Let σ, K, ℓ be some positive numbers and let D be a distribution on $\mathbb{R}^n$ such that for any fixed unit vector $\mathbf{u}$,*

$$\operatorname*{Exp}_{\mathbf{w}\sim D}[\langle \mathbf{u}, \mathbf{w}\rangle^2] \le \ell^2$$

and, moreover,

$$\Pr_{\mathbf{w}\sim D}\{\|\mathbf{w}\| \ge K\ell\} \le \sigma.$$

Let $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_N]$ be a matrix obtained by picking each column independently at random according to distribution $\mathbf{w}_i \sim D$. Then, with probability at least $1 - e^{-N/K^4}(4\sqrt{n}K^2)^n - N\sigma$ (over the choice of matrix $\mathbf{W}$) the maximum eigenvalue of the $n \times n$ matrix $\mathbf{W}\mathbf{W}^T$ is at most $3N\ell^2$.

We now define a variant of the closest vector problem that will be used as an intermediate step in our reduction from GapSVP to SIS′.

Definition 5.21. *An input to GapCVP$'_\gamma$ is a triple $(\mathbf{B}, \mathbf{t}, d)$, where $\mathbf{B}$ is an n-dimensional lattice basis, $\mathbf{t}$ is a target vector, and d is a rational number. In* Yes *inputs $\operatorname{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B})) \le d$. In* No *inputs $\lambda_1(\mathbf{B}) > \gamma(n) \cdot d$ and for any odd $k \in \mathbb{Z}$, $\operatorname{dist}(k\mathbf{t}, \mathcal{L}(\mathbf{B})) > \gamma(n) \cdot d$.*

The difference between GapCVP′ and the standard problem GapCVP is that when the target is far from the lattice, it also holds that any odd multiple of the target is far and the minimum distance of the lattice is large. In [13] it is shown that there is a polynomial time reduction from GapSVP$_\gamma$ to GapCVP$_\gamma$. We observe that the reduction given in [13] is also a reduction from GapSVP$_\gamma$ to GapCVP$'_\gamma$, as shown in the following lemma.

Lemma 5.22. *For any approximation factor $\gamma(n)$, there is a polynomial time reduction from GapSVP$_\gamma$ to GapCVP$'_\gamma$.*

Proof. In [13] it is shown that for any γ, there is a deterministic Cook reduction from GapSVP$_\gamma$ to GapCVP$_\gamma$ (see also [23]). Here we observe that the same reduction can be used as a reduction from GapSVP$_\gamma$ to GapCVP$'_\gamma$. To see this, it suffices to know that on input GapSVP$_\gamma$ instance $(\mathbf{B}, d)$, all the GapCVP$_\gamma$ calls made by the reduction have the form $(\mathbf{B}_i, \mathbf{b}_i, d)$, where $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ and $\mathbf{B}_i = [\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}, 2\mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_n]$. Moreover, the reduction outputs Yes if and only if any of the calls is answered Yes. Since GapCVP$_\gamma$ and GapCVP$'_\gamma$ have the same set of Yes instances, if the reduction is guaranteed to output Yes given a GapCVP$_\gamma$ oracle (e.g., when the input is a Yes instance), then it outputs Yes also when given access to a GapCVP$'_\gamma$ oracle. Now, let us consider the case when the input $(\mathbf{B}, d)$ is a No instance, and therefore all calls made by the reduction would receive a No answer from a GapCVP$_\gamma$ oracle. Notice that for any odd integer k, $\operatorname{dist}(k\mathbf{b}_i, \mathcal{L}(\mathbf{B}_i)) = \operatorname{dist}(\mathbf{b}_i, \mathcal{L}(\mathbf{B}_i))$ because $2\mathbf{b}_i \in \mathcal{L}(\mathbf{B}_i)$. Moreover, $\lambda_1(\mathbf{B}_i) \ge \lambda_1(\mathbf{B}) > \gamma d$. So, if $(\mathbf{B}_i, \mathbf{b}_i, d)$ is a No instance of GapCVP$_\gamma$, then it is also a No instance of GapCVP$'_\gamma$. Therefore all calls made by the reduction receive a No answer by the GapCVP$'_\gamma$ oracle as well, and the final output of the reduction is No. □

We now show how to solve GapCVP$'_\gamma$ in the worst case given access to an oracle that solves SIS′ on the average. By Lemma 5.22 this also implies a reduction from GapSVP$_\gamma$ to SIS′ (or SIS when the modulus q is odd).

Theorem 5.23. *For any polynomially bounded functions $\beta(n), m(n), q(n) = n^{O(1)}$, with $q(n) \ge 4\sqrt{m(n)}n^{1.5}\beta(n)$ and $\gamma(n) = 14\pi\sqrt{n}\beta(n)$, there is a probabilistic polynomial time reduction from solving GapCVP$'_\gamma$ in the worst case to solving SIS$'_{q,m,\beta}$ on the average with nonnegligible probability.*

In particular, for any $m(n) = \Theta(n \log n)$, there exist $q(n) = O(n^{2.5} \log n)$ and $\gamma(n) = O(n\sqrt{\log n})$, such that solving $\mathrm{SIS}'_{q,m}$ on the average is at least as hard as solving $\textsc{GapSVP}_\gamma$ *in the worst case.*

Proof. We adopt the notation of Theorem 5.9 and omit the dependence on n for the parameters $m, \gamma, \beta, q, \epsilon$, and δ. Let $\mathcal{F}$ be an oracle solving $\mathrm{SIS}'_{q,m,\beta}$ on the average with nonnegligible probability δ. Namely, $\mathcal{F}$ is an oracle that on input a random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ returns a vector $\mathbf{z} = \mathcal{F}(\mathbf{A}) \in \Lambda_q(\mathbf{A}) \setminus 2\mathbb{Z}^m$ of length at most β with probability δ. Notice that since $\mathbf{0} \notin \Lambda_q(\mathbf{A}) \setminus 2\mathbb{Z}^m$, oracle $\mathcal{F}$ also solves $\mathrm{SIS}_{q,m,\beta}$ with probability at least δ. We want to use $\mathcal{F}$ to solve $\textsc{GapCVP}'_\gamma$. The main idea is to use the NP verifier for (the complement of) $\textsc{GapCVP}$ presented in [1] as a routine for solving $\textsc{GapCVP}'$. To be able to do this, we need to be able to generate a good witness to that verifier. Such a witness is given by a set of short vectors sampled from the discrete Gaussian distribution in the dual lattice. Luckily, we can generate such a witness by using the sampling procedure and the combining procedure given in subsection 5.2 (together with $\mathcal{F}$). In fact, to be able to use these procedures, we need a reasonably short set of linearly independent vectors $\mathbf{S}$. We obtain such a set by using Corollary 5.13.[13]

We start by describing the NP verifier of [1]. For our purposes, it is best to think of this NP verifier as an algorithm, call it $\mathcal{V}$. The input to $\mathcal{V}$ consists of a lattice $\mathbf{B}$, a vector $\mathbf{t}$, a number $d > 0$, and a sequence of vectors $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_N]$ in $\mathcal{L}(\mathbf{B})^*$, where $N = n^3 m^3$. The algorithm $\mathcal{V}(\mathbf{B}, \mathbf{t}, d, \mathbf{W})$ performs the following three tests:

(a) Check that for all $i = 1, \ldots, N$, $\mathbf{w}_i \in \mathcal{L}(\mathbf{B})^*$.

(b) Check that $f_{\mathbf{W}}(\mathbf{t}) < 1/2$, where $f_{\mathbf{W}}$ is the function

$$f_{\mathbf{W}}(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} \cos(2\pi\langle \mathbf{x}, \mathbf{w}_i\rangle).$$

(c) Check that the largest eigenvalue of the $n \times n$ positive semidefinite matrix $\mathbf{W}\mathbf{W}^T$ is at most $N/(2\pi d)^2$.

If all three tests are satisfied, then $\mathcal{V}$ outputs YES; otherwise it outputs NO. It is shown in [1] that if $\mathrm{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B})) \le d$ then $\mathcal{V}(\mathbf{B}, \mathbf{t}, d, \mathbf{W})$ is guaranteed to output NO (for any matrix $\mathbf{W}$), while if $\mathrm{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B})) > c\sqrt{n}d$ (for some absolute constant c), then there exists a matrix $\mathbf{W}$ that makes $\mathcal{V}$ output YES.

We now describe our $\textsc{GapCVP}'$ reduction. From now on, fix $\epsilon(n) = 2^{-n}$. First, using Corollary 5.13 (with $\mathcal{F}$ as an oracle), we obtain a set of n linearly independent vectors $\mathbf{S}$ in $\mathcal{L}(\mathbf{B})^*$ such that $\|\mathbf{S}\| \le 8\beta\sqrt{n}\eta_\epsilon(\mathbf{B}^*)$. Define

$$s = \frac{2\sqrt{n}}{\gamma d}.$$

Consider the following procedure $\mathcal{W}(\mathbf{B}, \mathbf{S})$:

1. Run the sampling procedure $\mathcal{S}(\mathbf{B}^*, \mathbf{0}, s)$ of Lemma 5.7 on input a basis $\mathbf{B}^*$ of the dual lattice $\mathcal{L}(\mathbf{B})^*$. This procedure is run m times to generate m pairs of vectors $(\mathbf{c}_i, \mathbf{y}_i)$ and define the matrices $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_m]$ and $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_m]$.
2. Run the combining procedure $\mathcal{A}^{\mathcal{F}}(\mathbf{B}^*, \mathbf{S}, \mathbf{C}, q)$ of Lemma 5.8 with the oracle $\mathcal{F}$. Let $\mathbf{A}$ be the query asked by $\mathcal{A}$, and $\mathbf{z} = \mathcal{F}(\mathbf{A})$ the answer returned by the oracle.

[13] We remark that we could also use any polynomially longer set $\mathbf{S}$ with only a minor effect on our results.

3. If $\mathbf{z}$ is not a valid solution to SIS$'$ instance $(q, \mathbf{A}, \beta)$, then $\mathcal{W}$ aborts the computation with no output. Otherwise, let $\mathbf{x}$ be the vector returned by $\mathcal{A}$, and output the vector $\mathbf{w} = \mathbf{x} - \mathbf{Y}\mathbf{z}$.

We apply $\mathcal{W}(\mathbf{B}, \mathbf{S})$ nN/δ times, each time with independent randomness. If the number of nonaborting runs of $\mathcal{W}$ is less than N, then the reduction terminates immediately with output YES. Otherwise, let $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_N]$ be the vectors returned by the first N nonaborting runs of $\mathcal{W}$, and call $\mathcal{V}(\mathbf{B}, \mathbf{t}, d, \mathbf{W})$. If $\mathcal{V}$ says YES, the reduction outputs NO; otherwise, the reduction outputs YES. This completes the description of the reduction.

By the properties of $\mathcal{V}$, it is clear that whenever $\operatorname{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B})) \leq d$, the reduction correctly outputs YES (either because the number of nonaborting runs of $\mathcal{W}$ is less than N, or because $\mathcal{V}(\mathbf{B}, \mathbf{t}, d, \mathbf{W})$ outputs NO). For completeness, let us sketch the proof that $\mathcal{V}$ outputs NO whenever $\operatorname{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B})) \leq d$. Assume that the distance of $\mathbf{t}$ from $\mathcal{L}(\mathbf{B})$ is at most d and assume that tests (a) and (c) are satisfied. We show that test (b) must fail, and therefore $\mathcal{V}$ outputs NO. First, by the definition of $f_{\mathbf{W}}$ and the assumption that test (a) accepts, we have that $f_{\mathbf{W}}$ is periodic modulo $\mathcal{L}(\mathbf{B})$. Moreover, since the largest eigenvalue of $\mathbf{W}\mathbf{W}^T$ is bounded by $N/(2\pi d)^2$, we have that $\|\mathbf{W}^T\mathbf{x}\|^2 \leq N\|\mathbf{x}\|^2/(2\pi d)^2$ for any vector $\mathbf{x}$. Let $\tau(\mathbf{t})$ denote the lattice vector closest to $\mathbf{t}$. Notice that $\|\mathbf{t} - \tau(\mathbf{t})\| \leq d$. Since $f_{\mathbf{W}}$ is periodic modulo the lattice, $f_{\mathbf{W}}(\mathbf{t}) = f_{\mathbf{W}}(\mathbf{t} - \tau(\mathbf{t}))$. It thus suffices to prove that $f_{\mathbf{W}}(\mathbf{t} - \tau(\mathbf{t})) \geq 1/2$. Using the inequality $\cos x \geq 1 - x^2/2$ (valid for any $x \in \mathbb{R}$) we get

$$\begin{aligned} f_{\mathbf{W}}(\mathbf{t} - \tau(\mathbf{t})) &= \frac{1}{N}\sum_{i=1}^{N} \cos(2\pi\langle \mathbf{t} - \tau(\mathbf{t}), \mathbf{w}_i\rangle) \\ &\geq 1 - \frac{4\pi^2}{2N}\sum_{i=1}^{N} \langle \mathbf{t} - \tau(\mathbf{t}), \mathbf{w}_i\rangle^2 \\ &= 1 - \frac{2\pi^2}{N}\|\mathbf{W}^T(\mathbf{t} - \tau(\mathbf{t}))\|^2 \\ &\geq 1 - \frac{\|\mathbf{t} - \tau(\mathbf{t})\|^2}{2d^2} \geq \frac{1}{2}. \end{aligned}$$

It remains to show that the reduction outputs the correct answer when the input is a NO instance, i.e., when $\lambda_1(\mathbf{B}) > \gamma d$ and $\operatorname{dist}(k\mathbf{t}, \mathcal{L}(\mathbf{B})) > \gamma d$ for any odd $k \in \mathbb{Z}$. In order for our reduction to output NO, two conditions must be satisfied: at least N calls to $\mathcal{W}$ succeed and $\mathcal{V}$ outputs YES. Let us first show that after $n \cdot N/\delta$ calls to $\mathcal{W}$, we obtain at least N vectors with high probability. By Lemma 3.2, we have that

$$\eta_\epsilon(\mathbf{B}^*) \leq \frac{\sqrt{n}}{\lambda_1(\mathbf{B})} < \frac{\sqrt{n}}{\gamma d} = \frac{s}{2} \tag{15}$$

and hence s satisfies $s > 2\eta_\epsilon(\mathbf{B}^*)$. Therefore, by Lemma 5.7, the pairs $(\mathbf{c}_i, \mathbf{y}_i)$ computed by the sampling procedure $\mathcal{S}(\mathbf{B}^*, \mathbf{0}, s)$ satisfy that $\mathbf{c}_i$ is within distance $\epsilon/2$ from the uniform distribution. It follows that $\mathbf{C}$ is within distance $m\epsilon/2$ from the uniform distribution over $\mathcal{P}(\mathbf{B}^*)^m$. Hence, by Lemma 5.8, the query $\mathbf{A}$ given to the oracle by the combining procedure $\mathcal{A}^{\mathcal{F}}(\mathbf{B}^*, \mathbf{S}, \mathbf{C}, q)$ is within negligible distance $\epsilon m/2$ from the uniform distribution, and the oracle returns a vector $\mathbf{z}$ such that $\mathbf{z} \in \Lambda_q(\mathbf{A}) \setminus 2\mathbb{Z}^m$ and $\|\mathbf{z}\| \leq \beta$ with probability at least $\delta - \epsilon m/2 > \delta/2$ for all sufficiently large n. Thus, the probability that out of $n \cdot N/\delta$ calls to $\mathcal{W}$ fewer than N are successful is at most $N(1 - \delta/2)^{n/\delta} \leq Ne^{-n/2} < 2^{-n/2}$.

It remains to show that $\mathcal{V}$ outputs YES with high probability. The proof of this is based on [1]. However, in [1], it is only shown that there *exists* a good matrix $\mathbf{W}$ that makes $\mathcal{V}$ output YES. Here, we have to argue that the $\mathbf{W}$ given by $\mathcal{W}$ is good with high probability.

First, we observe that test (a) is always satisfied since $\mathcal{W}(\mathbf{B}, \mathbf{S})$ is guaranteed to output vectors in $\mathcal{L}(\mathbf{B})^*$. Indeed, $\mathbf{x}, \mathbf{y}_1, \ldots, \mathbf{y}_m \in \mathcal{L}(\mathbf{B})^*$, and hence $\mathbf{x} - \mathbf{Yz}$ also belongs to the lattice $\mathcal{L}(\mathbf{B})^*$. In the rest of the proof, we show that tests (b) and (c) are satisfied with high probability. To this end, notice that each vector $\mathbf{w}$ in $\mathbf{W}$ is distributed independently according to the distribution D defined as the output of $\mathcal{W}(\mathbf{B}, \mathbf{S})$ conditioned on a nonaborting run.

Consider test (b). Our goal is to show that $\Pr\{f_{\mathbf{W}}(\mathbf{t}) \geq 1/2\}$ is small. Below, we will show that

$$\left| \operatorname*{Exp}_{\mathbf{w}}[\cos(2\pi\langle \mathbf{t}, \mathbf{w}\rangle)] \right| \leq 2^{-n+1}, \tag{16}$$

where $\mathbf{w}$ is distributed according to D. By Hoeffding's bound (14), this would imply that $f_{\mathbf{W}}(\mathbf{t}) \geq 1/2$ with probability at most $e^{-N(\frac{1}{2} - \mathrm{Exp}[f_{\mathbf{W}}(\mathbf{t})])^2/4} \leq e^{-N(\frac{1}{2} - 2^{-n+1})^2/4} = 2^{-\Omega(N)}$. We now prove (16). We in fact show that it is true even when we condition on any fixed values of $\mathbf{C}$, $\mathbf{A}$, and $\mathbf{z} \in \Lambda_q(\mathbf{A}) \setminus 2\mathbb{Z}^m$. Furthermore, we condition on any fixed values of $\mathbf{y}_1, \ldots, \mathbf{y}_{j-1}, \mathbf{y}_{j+1}, \ldots, \mathbf{y}_m$, where j is some index for which z_j is odd. Notice that the only randomness left is in $\mathbf{y}_j$, which, by Lemma 5.7, is distributed according to $D_{\mathcal{L}(\mathbf{B})^*, s, \mathbf{c}_j}$. Hence, a sample $\mathbf{w} = \mathbf{x} - \mathbf{Yz}$ can be written as $-z_j(\hat{\mathbf{w}} + \mathbf{y}_j)$ for some fixed vector $\hat{\mathbf{w}}$. Notice that

$$\cos(2\pi\langle \mathbf{t}, \mathbf{w}\rangle) = \cos(2\pi\langle \mathbf{t}, -z_j(\hat{\mathbf{w}} + \mathbf{y}_j)\rangle) = \cos(2\pi\langle -z_j\mathbf{t}, \hat{\mathbf{w}} + \mathbf{y}_j\rangle).$$

By Corollary 4.6, (15), and $\mathrm{dist}(-z_j\mathbf{t}, \mathcal{L}(\mathbf{B})) > \gamma d = 2\sqrt{n}/s$, we obtain

$$\left| \operatorname*{Exp}_{\mathbf{y}_j}[\cos(2\pi\langle \mathbf{t}, \mathbf{w}\rangle)] \right| = \left| \operatorname*{Exp}_{\mathbf{y}_j}[\cos(2\pi\langle \hat{\mathbf{w}} + \mathbf{y}_j, -z_j\mathbf{t}\rangle)] \right| \leq \frac{1+\epsilon}{1-\epsilon} \cdot 2^{-n} \leq 2^{-n+1}.$$

In the rest of the proof we show that test (c) is satisfied with high probability. We do this by applying Lemma 5.20 with $\ell = 2s\beta$, $N = n^3m^3$, $\sigma = N2^{-n}(1+\epsilon)/(1-\epsilon)$, $K = \sqrt{n \cdot m}$, and the distribution D. Assuming the hypothesis in that lemma holds, we get that the maximum eigenvalue of matrix $\mathbf{WW}^T$ is bounded by

$$3N\ell^2 = 12Ns^2\beta^2 = \frac{48Nn\beta^2}{\gamma^2 d^2} < \frac{N}{(2\pi d)^2}$$

except possibly with probability

$$e^{-N/K^4}(4\sqrt{n}K^2)^n + N\sigma \leq (4e^{-m}n^{1.5}m)^n + N^2 2^{-n+1} \leq 2^{-n/2}.$$

Therefore, the probability that test (c) fails is exponentially small. It remains to check that the hypothesis of Lemma 5.20 is satisfied, i.e., that

$$\Pr_{\mathbf{w}}\{\|\mathbf{w}\| \geq 2\sqrt{n \cdot m}s\beta\} \leq \sigma \tag{17}$$

and for any unit vector $\mathbf{u}$,

$$\operatorname*{Exp}_{\mathbf{w}}[\langle \mathbf{u}, \mathbf{w}\rangle^2] \leq 4s^2\beta^2. \tag{18}$$

In the following, we show that (17) and (18) are true even when we condition on any fixed values of $\mathbf{C}$, $\mathbf{A}$, and $\mathbf{z}$. The only randomness left is in $\mathbf{y}_1, \ldots, \mathbf{y}_m$ where each $\mathbf{y}_i$ is distributed according to $D_{\mathcal{L}(\mathbf{B})^*,s,\mathbf{c}_i}$.

We first prove (17). We can write a vector $\mathbf{w}$ produced by $\mathcal{W}$ as

$$\mathbf{w} = \mathbf{x} - \mathbf{Y}\mathbf{z} = (\mathbf{x} - \mathbf{C}\mathbf{z}) - (\mathbf{Y} - \mathbf{C})\mathbf{z}.$$

By Lemma 5.8 and (15), the norm of the first term is at most

$$\begin{aligned}\|\mathbf{x} - \mathbf{C}\mathbf{z}\| &\le \frac{\sqrt{m}n\|\mathbf{S}\| \cdot \|\mathbf{z}\|}{q} \\ &\le \frac{8\sqrt{m} \cdot n^{1.5}\eta_\epsilon(\mathbf{B}^*) \cdot \beta^2}{q} \\ &\le 2\beta\eta_\epsilon(\mathbf{B}^*) < s\beta\end{aligned}$$

with probability 1. By Lemma 4.4, for every i, the probability that $\|\mathbf{y}_i - \mathbf{c}_i\| > s\sqrt{n}$ is at most $2^{-n}(1+\epsilon)/(1-\epsilon) = \sigma/N$. Hence, by union bound and triangle inequality, the norm of the second term is bounded by

$$\|(\mathbf{Y} - \mathbf{C})\mathbf{z}\| \le s\sqrt{n}\sum_{i=1}^{m} |z_i| \le s\sqrt{n}\sqrt{m}\beta$$

with probability at least $1-\sigma$. It follows that with probability at least $1-\sigma$ the norm of $\mathbf{w}$ is bounded by $s\beta + \sqrt{nm}s\beta < 2\sqrt{nm}s\beta$ for all sufficiently large n, proving (17).

Next, we prove (18). Fix some unit vector $\mathbf{u}$ and let us bound the expected value of $\langle \mathbf{u}, \mathbf{w}\rangle^2$. Using the inequality $(a-b)^2 \le 2a^2 + 2b^2$ (valid for all $a, b \in \mathbb{R}$), we can write

$$\begin{aligned}\langle \mathbf{u}, \mathbf{w}\rangle^2 &= (\langle \mathbf{u}, \mathbf{x} - \mathbf{C}\mathbf{z}\rangle - \langle \mathbf{u}, (\mathbf{Y} - \mathbf{C})\mathbf{z}\rangle)^2 \\ &\le 2\langle \mathbf{u}, \mathbf{x} - \mathbf{C}\mathbf{z}\rangle^2 + 2\langle \mathbf{u}, (\mathbf{Y} - \mathbf{C})\mathbf{z}\rangle^2 \\ &\le 2\|\mathbf{x} - \mathbf{C}\mathbf{z}\|^2 + 2\langle \mathbf{u}, (\mathbf{Y} - \mathbf{C})\mathbf{z}\rangle^2.\end{aligned}$$

Using Lemma 4.2, we obtain that for all $i = 1, \ldots, m$,

$$|\operatorname{Exp}[\langle \mathbf{u}, \mathbf{y}_i - \mathbf{c}_i\rangle]| \le \frac{\epsilon s}{1-\epsilon}, \tag{19}$$

$$\operatorname{Exp}[\langle \mathbf{u}, \mathbf{y}_i - \mathbf{c}_i\rangle^2] \le \Big(\frac{1}{2\pi} + \frac{\epsilon}{1-\epsilon}\Big)s^2. \tag{20}$$

Using (19), (20), and Lemma 2.11 with $\mathbf{v}_i = \langle \mathbf{u}, \mathbf{y}_i - \mathbf{c}_i\rangle$ as 1-dimensional vectors, we obtain

$$\begin{aligned}\operatorname{Exp}[\langle \mathbf{u}, (\mathbf{Y} - \mathbf{C})\mathbf{z}\rangle^2] &= \operatorname{Exp}\left[\left(\sum_{i=1}^{m} \langle \mathbf{u}, \mathbf{y}_i - \mathbf{c}_i\rangle z_i\right)^2\right] \\ &\le \left(\left(\frac{1}{2\pi} + \frac{\epsilon}{1-\epsilon}\right)s^2 + \left(\frac{\epsilon}{1-\epsilon}\right)^2 s^2 m\right)\|\mathbf{z}\|^2 \\ &\le \left(\frac{1}{2\pi} + \frac{\epsilon}{1-\epsilon} + \left(\frac{\epsilon}{1-\epsilon}\right)^2 m\right)s^2\beta^2 \\ &\le s^2\beta^2.\end{aligned}$$

Using this bound in the expression for $\operatorname{Exp}[\langle \mathbf{u}, \mathbf{w}\rangle^2]$ we get that

$$\operatorname{Exp}[\langle \mathbf{u}, \mathbf{w}\rangle^2] < 2(s\beta)^2 + 2(s\beta)^2 = 4s^2\beta^2. \qquad \square$$

Appendix. Proof of Lemma 5.20. The largest eigenvalue of $\mathbf{W} \cdot \mathbf{W}^T$ is at most $3N\ell^2$ if and only if

$$\frac{1}{N}\sum_{i=1}^{N} \langle \mathbf{u}, \mathbf{w}_i \rangle^2 \leq 3\ell^2$$

for all unit vectors $\mathbf{u} \in \mathbb{R}^n$. In the following, we show that this condition is satisfied with the desired probability. Let $\xi : \mathbb{R}^n \to \mathbb{R}^n$ be the function defined by $\xi(\mathbf{x}) = \mathbf{x}$ if $\|x\| \leq K\ell$ and $\xi(\mathbf{x}) = 0$ otherwise. Clearly, for any unit vector $\mathbf{u}$,

$$\operatorname*{Exp}_{\mathbf{w}\sim D}[\langle \mathbf{u}, \xi(\mathbf{w})\rangle^2] \leq \operatorname*{Exp}_{\mathbf{w}\sim D}[\langle \mathbf{u}, \mathbf{w}\rangle^2] \leq \ell^2.$$

Moreover, the random variable $\langle \mathbf{u}, \xi(\mathbf{w})\rangle^2$ takes values in the interval $[0, (K\ell)^2]$. Hence, Hoeffding's inequality (14) implies that for any unit vector $\mathbf{u}$, a sequence of samples $\mathbf{w}_1, \ldots, \mathbf{w}_N$ from D satisfies

$$\frac{1}{N}\sum_{i=1}^{N} \langle \mathbf{u}, \xi(\mathbf{w}_i)\rangle^2 \leq 2\ell^2 \tag{21}$$

with probability at least $1 - e^{-N/K^4}$.

Consider an ϵ-net A on the unit sphere with parameter $\epsilon = \frac{1}{2}K^{-2}$, i.e., a set of points A such that any point on the unit sphere is within distance ϵ from some point in A. It is possible to construct such nets of size at most $(2\sqrt{n}/\epsilon)^n$. For instance, let C be $[-1, 1]^n$, i.e., the n-dimensional cube of edge length 2. Notice that C contains the unit sphere. Partition C into $(2\sqrt{n}/\epsilon)^n$ small cubes of edge length $\epsilon/\sqrt{n}$. For each small cube that intersects the n-dimensional sphere, choose any point in the intersection and include it in A. It is easy to see that the collection of these points constitutes an ϵ-net on the sphere, because any point in the sphere belongs to one of the small cubes, and the diameter of each small cube is exactly ϵ.

We now apply the union bound on the set of all unit vectors $\mathbf{u}$ in A. It follows that (21) holds with probability at least $1 - e^{-N/K^4}(4\sqrt{n}K^2)^n$ for all $\mathbf{u}$ in the net A *simultaneously*.

Next, we show that if (21) holds for all $\mathbf{u} \in A$, then a slightly weaker version of it holds for *all* unit vectors. Consider an arbitrary unit vector $\mathbf{u}'$. Let $\mathbf{u} \in A$ be the closest point to $\mathbf{u}'$ in A. Notice that $\|\mathbf{u} - \mathbf{u}'\| \leq \epsilon$. Thus,

$$\begin{aligned} \left| \frac{1}{N}\sum_{i=1}^{N} \langle \mathbf{u}', \xi(\mathbf{w}_i)\rangle^2 - \frac{1}{N}\sum_{i=1}^{N} \langle \mathbf{u}, \xi(\mathbf{w}_i)\rangle^2 \right| &\leq \frac{1}{N}\sum_{i=1}^{N} |\langle \mathbf{u}' - \mathbf{u}, \xi(\mathbf{w}_i)\rangle \langle \mathbf{u}' + \mathbf{u}, \xi(\mathbf{w}_i)\rangle| \\ &\leq 2\epsilon \max_i \|\xi(\mathbf{w}_i)\|^2 \leq 2\epsilon(K\ell)^2 = \ell^2. \end{aligned}$$

This yields that with probability at least $1 - e^{-N/K^4}(4\sqrt{n}K^2)^n$ over the choice of the $\mathbf{w}_i$'s it holds that

$$\frac{1}{N}\sum_{i=1}^{N} \langle \mathbf{u}, \xi(\mathbf{w}_i)\rangle^2 \leq 2\ell^2 + \ell^2 = 3\ell^2$$

for all unit vectors $\mathbf{u}$. It remains to note that with probability at least $1 - N\sigma$, $\xi(\mathbf{w}_i) = \mathbf{w}_i$ for all i.

Acknowledgments. Part of this work was done while both authors were visiting the Institute for Advanced Study, Princeton, NJ. We thank the anonymous referees for their helpful comments.

REFERENCES

[1] D. Aharonov and O. Regev, *Lattice problems in NP intersect coNP*, J. ACM, 52 (2005), pp. 749–765.

[2] M. Ajtai, *Generating hard instances of lattice problems*, in Complexity of Computations and Proofs, Quad. Mat. 13, Dept. Math., Seconda Univ. Napoli, Caserta, Italy, 2004, pp. 1–32.

[3] M. Ajtai and C. Dwork, *A public-key cryptosystem with worst-case/average-case equivalence*, in Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC '97), 1997, pp. 284–293.

[4] L. Babai, *On Lovasz' lattice reduction and the nearest lattice point problem*, Combinatorica, 6 (1986), pp. 1–13.

[5] W. Banaszczyk, *New bounds in some transference theorems in the geometry of numbers*, Math. Ann., 296 (1993), pp. 625–635.

[6] J. Blömer and J.-P. Seifert, *On the complexity of computing short linearly independent vectors and short bases in a lattice*, in Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC '99), Atlanta, GA, 1999, pp. 711–720.

[7] A. Bogdanov and L. Trevisan, *On worst-case to average-case reductions for NP problems*, in Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2003, pp. 308–317.

[8] J.-Y. Cai, *A new transference theorem in the geometry of numbers and new bounds for Ajtai's connection factor*, Discrete Appl. Math., 126 (2003), pp. 9–31.

[9] J-Y. Cai and A. Nerurkar, *An improved worst-case to average-case connection for lattice problems*, in Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 1997, pp. 468–477.

[10] W. Ebeling, *Lattices and Codes. A Course Partially Based on Lectures by F. Hirzebruch*, Adv. Lectures Math., 2nd revised ed., Friedr. Vieweg & Sohn, Braunschweig, Germany, 2002.

[11] O. Goldreich and S. Goldwasser, *On the limits of nonapproximability of lattice problems*, J. Comput. System Sci., 60 (2000), pp. 540–563.

[12] O. Goldreich, S. Goldwasser, and S. Halevi, *Collision-free Hashing from Lattice Problems*, Technical report TR96-056, Electronic Colloquium on Computational Complexity (ECCC), 1996.

[13] O. Goldreich, D. Micciancio, S. Safra, and J.-P. Seifert, *Approximating shortest lattice vectors is not harder than approximating closest lattice vectors*, Inform. Process. Lett., 71 (1999), pp. 55–61.

[14] V. Guruswami, D. Micciancio, and O. Regev, *The complexity of the covering radius problem*, Comput. Complexity, 14 (2005), pp. 90–121.

[15] W. Hoeffding, *Probability inequalities for sums of bounded random variables*, J. Amer. Statist. Assoc., 58 (1963), pp. 13–30.

[16] S. Khot, *Hardness of approximating the shortest vector problem in lattices*, J. ACM, 52 (2005), pp. 789–808.

[17] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász, *Factoring polynomials with rational coefficients*, Math. Ann., 261 (1982), pp. 515–534.

[18] D. Micciancio, *Improving lattice based cryptosystems using the Hermite normal form*, in Cryptography and Lattices, Lecture Notes in Comput. Sci. 2146, J. Silverman, ed., Springer-Verlag, Berlin, 2001, pp. 126–145.

[19] D. Micciancio, *The shortest vector in a lattice is hard to approximate to within some constant*, SIAM J. Comput., 30 (2001), pp. 2008–2035.

[20] D. Micciancio, *A note on the minimal volume of almost cubic parallelepiped*, Discrete Comput. Geom., 29 (2002), pp. 133–138.

[21] D. Micciancio, *Almost perfect lattices, the covering radius problem, and applications to Ajtai's connection factor*, SIAM J. Comput., 34 (2004), pp. 118–169.

[22] D. Micciancio, *Generalized compact knapsacks, cyclic lattices, and efficient one-way functions*, Comput. Complexity, to appear.

[23] D. Micciancio and S. Goldwasser, *Complexity of Lattice Problems: A Cryptographic Perspective*, Kluwer Internat. Ser. Engrg. Comput. Sci., Kluwer Academic Publishers, Boston, 2002.

[24] O. Regev, *New lattice-based cryptographic constructions*, J. ACM, 51 (2004), pp. 899–942.

SIAM J. COMPUT.
Vol. 37, No. 1, pp. 303–318

A POLYNOMIAL TIME ALGORITHM FOR COMPUTING AN ARROW–DEBREU MARKET EQUILIBRIUM FOR LINEAR UTILITIES*

KAMAL JAIN†

Abstract. We provide the first polynomial time exact algorithm for computing an Arrow–Debreu market equilibrium for the case of linear utilities. Our algorithm is based on solving a convex program using the ellipsoid algorithm and simultaneous diophantine approximation. As a side result, we prove that the set of assignments at equilibrium is convex and the equilibrium prices themselves are log-convex. Our convex program is explicit and intuitive, which allows maximizing a concave function over the set of equilibria. On the practical side, Ye developed an interior point algorithm [*Lecture Notes in Comput. Sci.* 3521, Springer, New York, 2005, pp. 3–5] to find an equilibrium based on our convex program. We also derive separate combinatorial characterizations of equilibrium for Arrow–Debreu and Fisher cases. Our convex program can be extended for many nonlinear utilities and production models. Our paper also makes a powerful theorem (Theorem 6.4.1 in [M. Grotschel, L. Lovasz, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, 2nd ed., Springer-Verlag, Berlin, Heidelberg, 1993]) even more powerful (in Theorems 12 and 13) in the area of geometric algorithms and combinatorial optimization. The main idea in this generalization is to allow ellipsoids to contain not the whole convex region but a part of it. This theorem is of independent interest.

Key words. polynomial time algorithms, economics, market equilibrium, convex set

AMS subject classifications. 02E10, 05C85, 52B55

DOI. 10.1137/S0097539705447384

1. Introduction.

We present the first polynomial time algorithm to compute the exact general market equilibrium for the linear utility case, thereby settling an open problem posed by Papadimitriou [35] and by Deng, Papadimitriou, and Safra [10]. A version of the problem was first formulated by a French economist, Leon Walras, in 1874 [37]. In this model, every person in an entire population has an initial endowment of divisible goods. Furthermore, every person in the population has a utility function for consuming goods. Every person sells the initial endowment and then buys an optimal bundle of goods with the entire revenue; i.e., the market clears. Walras asked whether a price can be assigned to every good so that this is possible. Such a price vector, if it exists, is called general market equilibrium. An answer was given by two Nobel laureates, Arrow and Debreu, in 1954 [2]. They showed that when the utility functions are concave then under some mild conditions a general market equilibrium always exists [2]. Their proof is nonconstructive and does not suggest any polynomial time algorithm for finding a general market equilibrium.

Fisher was the first to consider the computability of equilibrium prices (see [3]). Independent of Walras' work, Fisher defined another model, which is a special case of the Walras model, in 1891. In his model there are two kinds of people, producers and consumers. Consumers have money and utility functions for goods. Producers have initial endowments of goods and want to earn money. The equilibrium prices are defined as the assignment of prices to goods so that, when every consumer buys

*Received by the editors March 9, 2005; accepted for publication (in revised form) January 20, 2007; published electronically May 14, 2007. A preliminary version of this work appeared with the same title in Proceedings of FOCS 2004, pp. 286–294.

http://www.siam.org/journals/sicomp/37-1/44738.html

†Microsoft Research, One Microsoft Way, Redmond, WA 98052 (kamalj@microsoft.com).

an optimal bundle, then the market clears; i.e., all the money is spent and all the goods are sold. If money is also considered a commodity, then it is easy to see that the Fisher model is a special case of the Walras model; hence, the Arrow–Debreu theorem [2] also implies the existence of a market equilibrium for the Fisher model for the case of concave utilities.

Although the Walras and Fisher models seem very close, there are some fundamental differences between them. The Walras model suggests that money has no intrinsic value except that it is a scale to measure the value of a good. The Fisher model, on the other hand, assumes the value of the money. The Fisher model can also be viewed as a generalization of the concept of demand and supply curves, where demand is a decreasing function of prices and supply is an increasing function of prices; a point where they meet is called an *equilibrium*. On the other hand, the Walras model allows the demand to be an increasing function of income as well as a decreasing function of prices. In practice, at a macro level, when the prices increase, the demand decreases. But then the incomes also increase, because consumers and producers (i.e., workers in the global economy system) are the same, which has an increasing effect on demand. This feedback feature of the Walras model not only makes it more realistic but also puts it at a much higher difficulty level than the Fisher model. This is also evident from the results obtained about the Fisher model.

Eisenberg and Gale [17] gave a constructive proof using the variational method for the Fisher model in the case of linear utilities. Their proof shows that the equilibrium assignment is the one which maximizes the product of utilities obtained by every dollar of the buyers. This gives a convex function which is minimized at the equilibrium. This implies a polynomial time (approximation) algorithm, in the sense of numerical computing, i.e., polynomial in the input size and $\log(1/\epsilon)$, where ϵ is the precision in computation. The first exact algorithm was recently developed by Devanur et al. [12]. None of these algorithms is strongly polynomial time.

There was no polynomial time algorithm known for the Walras model, although the problem has been studied extensively and many heuristics have been developed using the Lemke method, Newton method, primal-dual method, convex programming, and various other techniques in numerical optimization (see, e.g., [15, 36, 14, 18]). Deng, Papadimitriou, and Safra [10] gave polynomial time algorithms for a bounded number of agents or goods and asked the question for the general case. The importance of polynomial time computational results for computing equilibria was also mentioned earlier in [34].

Recently two[1] approximation schemes have been developed for the Walras model [25, 20, 13]. The first scheme [25] also works for some nonlinear utility functions for those instances which satisfy the condition of gross substitutability (i.e., if the price of some good is raised, then the demand of other goods can only increase). An approximation scheme was also developed by Newman and Primak [33] by running the algorithm Ellipsoid on an infinite linear program. The inequalities of the infinite linear program were given by Arrow, Block, and Hurwicz [1] and hold for a very general class of utility functions satisfying gross substitutability. We refer readers to [6] to get further details and pointers on this approach. Approximation schemes developed in [25, 20, 13] interpret the approximation factor in a physical sense. On the other hand, the sense of approximation in [33] was not clear. One reason is that the linear inequalities used by [33] have both positive and negative coefficients, with their signs not clear a priori. We again refer the readers to [6] for a physical interpretation of the

[1] [13] is a simple variation of [25].

approximation in [33]. In some sense these approximation schemes [33, 25, 20, 13] are pseudoapproximation, because they do not guarantee that an equilibrium is nearby.

In this paper we propose a mathematical program for the Walras model for the case of linear utilities. We show that the program is valid for nonlinear utilities too. The program is simple and does not have complicated constraints like "optimality of the bundles purchased" and/or nonlinear constraints like "money spent equals money earned." Instead these constraints follow from the feasibility of the program. In general, constraints written do not seem to have simple economic interpretations, which probably is the reason that they were not discovered so far. For the case of linear utilities, they do have some economic interpretation, and also the program can be turned into at least two different convex programs. For linear utilities, [7] reports that one of the convex programs in this paper also exists in the Russian literature [32]. Our programs turn out to be convex for some cases of nonlinear utilities too. We show that all the general market equilibria are feasible points of this convex program and vice versa; hence, a general market equilibrium can be obtained using the ellipsoid method and simultaneous diophantine approximation. This leads us to develop Theorems 12 and 13, which make a powerful geometric algorithm even more powerful. These theorems are about general convex programs and not restricted to game theory.

Our convex program is explicitly given and can easily be converted into an optimization program. Using this convex program, Ye [38] developed computationally efficient interior point algorithms for computing market equilibria. Besides computability, we get some insight into the structure of the equilibrium itself. Our convex program shows that the set of all the equilibrium assignments is convex. It also shows that the set of all the equilibrium prices on a logarithmic scale is convex too. Indeed, algorithms via convex programs have distinct theoretical advantages. Almost any known basic fact about the Fisher model with linear utilities can be easily observed from the Eisenberg–Gale convex program. Convexity, which gives dual interpretation to prices, is one of them. Uniqueness of equilibrium utilities and prices, rationality of prices, proportionate fairness of the equilibrium [31], and combinatorial characterization (Theorem 11) are some others. Eisenberg and Gale's program gives the fastest known algorithm [38] for the problem. Their convex program can be generalized for homogeneous concave utilities [16] and homogeneous quasi-concave utilities [29] too. As shown in many thought experiments in [38], their convex program is quite natural and implies a proof of the Arrow–Debreu case by using a fixed point theorem. Furthermore, this convex program allows Jain, Vazirani, and Ye [29] to define equilibrium for a production model with some economies of scales in production. Note that, in general, the Arrow–Debreu theorem breaks down as soon as we have any kind of economies of scale. We do not deal with production in this paper. The ideas in this paper are extended to production setting in [26].

Indeed most of these features are also shared by our convex program in this paper. The only main difference is that the Eisenberg–Gale convex program gives a constructive proof of the existence of an equilibrium, whereas our convex program does not. The fact that no constructive proof has been developed for the Arrow–Debreu theorem in the last half century hints toward an intrinsic difficulty of the model. Intuitively, one reason that the use of a fixed point theorem cannot be done away with easily is the two-sided feedback of prices—demand decreases with prices, whereas income increases, which in turn has positive effect on demand. We are not aware of any other problem where the only proof of existence is via fixed point theorems and yet a polynomial time algorithm is successfully developed. Existence of taxes in selfish routing in [8] was one of them. Using a straightforward linear program, [19] proposed a fairly simple extension of this result and its various generalizations.

In other related work, [26] generalizes the results in this paper to include production planning constraints, and [5] generalizes the results in this paper to include utility functions with constant elasticity of substitution (CES). An ongoing work [30] extends the convex programs to include quasi-concave functions. We conjecture that our convex program remains convex for the utility functions satisfying the gross substitution property. One reason for this conjecture is that the set of equilibria is known to be convex for the utility functions satisfying the gross substitution property [1, 6]. The convexity is proved via an infinite linear program, which may not allow efficient interior point methods. An advantage of our approach is that we get explicit polynomial sized convex programs.

Finally, we would like to close this section by emphasizing the need for efficient computational results in economics of equilibria, not only for the development of computer science but for the development of economics itself. If a Turing machine cannot compute something, then an exonomic system cannot compute that thing either. An economic system, by converging at an equilibrium, is also computing it. Note that under the "rationality of agents" assumption, an economic system is a collection of Turing machines, which is equivalent to a single Turing machine itself. If equilibria are not computable with a Turing machine, then it is unlikely that an economic system will be able to compute them either.[2] Hence, a lack of computational results decreases the applicability of the equilibrium itself. In this regard, computing a Nash equilibrium was a big open question, which was recently resolved in [9, 4] in the negative. Approximating it with a nontrivial factor is still a big open problem.

2. Model. There are n people in the system. They each have some initial endowment of divisible goods. Without loss of generality (in the linear utility case) we can assume that each person has only one good, which is different from the goods which other people have.[3] Further, by simple scaling, we can assume that each person has only one unit of good. Each person also has a linear utility function. For the ith person we denote this utility function by $\sum_j u_{ij}x_{ij}$, where x_{ij} is the amount of good j consumed by i. To be able to keep full precision in digital computers we assume that each u_{ij} is an integer. Each person maximizes her utility by buying an optimal bundle of goods, disregarding any supply constraint, with the revenue made by selling her own endowment. The classical Arrow–Debreu [2] theorem says that there is a price vector, $(p_1, p_2, \dots, p_n)$, not all equal to zero, such that the buying and selling can be done at the prices in $(p_1, p_2, \dots, p_n)$ in such a way that the market clears.[4] This price vector is called *general market equilibrium.*

Without loss of generality we assume that everybody likes something; that is, for every i there is a j such that $u_{ij} > 0$. If somebody does not like anything, then the price of her good can be anything and she can be given any bundle. Again without

[2] If one believes that an economic system could compute something which a Turing machine could not, then the need for studying economic systems from a computational point of view is even greater, because in this case an economic system is also a computation which violates the Church–Turing thesis! Quantum mechanics is the only violation known to the Church–Turing thesis, hence justifiably being extensively studied from a computational point of view.

[3] Because if a person has multiple goods, then one can create clones of the person, each with one good, having the same utility function. If two persons have the same good, then name them differently. That these goods are the same can be reflected in the utility functions by always giving them the same utility coefficient.

[4] "Market clears" means that no good has smaller supply than demand, and demand could be smaller than supply only for zero priced goods. Note that "market clears" is a redundant phrase here because of the optimality condition being placed on each buyer.

loss of generality we assume that every good is liked by somebody; that is, for every j there is an i such that $u_{ij} > 0$. If some good is not liked by anybody, then its price must be zero. So not much to discover about its price. These assumptions we make only for convenience.

In the next two sections we assume that for every proper subset, S, of persons (i.e., neither empty nor everybody) there is a person, i, outside S, who likes some good possessed by S; i.e., $u_{ij} > 0$ for some $i \notin S$ and some $j \in S$. (This assumption also implies our previous assumption that everybody likes something.) This assumption implies that all the equilibrium prices are nonzero. If not, then consider S as the set of persons having the zero priced good. Then somebody outside S will demand infinite quantity of something possessed by S. This assumption is not without loss of generality. Thus, we will justify this assumption after the next two sections.

3. Nonconvex program. In this section we give a nonconvex program which has all and only general market equilibria as feasible points:

$$\begin{aligned} &\forall j:\ \sum_i x_{ij} = 1,\\ &\forall i,j:\ x_{ij} \geq 0,\\ &\forall i,j:\ \frac{u_{ij}}{p_j} \leq \frac{\sum_k u_{ik}x_{ik}}{p_i},\\ &\forall i:\ p_i > 0. \end{aligned} \tag{1}$$

The first two lines of this program say that x_{ij} are feasible assignments. The third line says that the goods purchased by i by spending p_i (which is her revenue) have the highest utility. The fourth line says that the prices are nonzero. The Arrow–Debreu theorem guarantees only that at least one price is nonzero, but our assumption before the section makes all the prices nonzero. It is easy to see that any general market equilibrium will satisfy all the lines of this nonconvex program. We claim that, in fact, the converse is also true.

THEOREM 1. *The feasible region of nonconvex program* (1) *has all and only general market equilibria.*

Proof. It is clear that all the market equilibria satisfy the program. So we need to show only that any feasible point is a market equilibrium. Line 3 of the program, when multiplied with $x_{ij}p_j$, gives

$$\forall i,j:\ u_{ij}x_{ij} \leq \frac{\sum_k u_{ik}x_{ik}}{p_i} x_{ij}p_j.$$

When we add these inequalities for all j we get

$$\forall i:\ \sum_j u_{ij}x_{ij} \leq \frac{\sum_k u_{ik}x_{ik}}{p_i} \sum_j x_{ij}p_j.$$

Note that our assumption that everybody likes something implies that $\sum_k u_{ik}x_{ik}$ is not zero. Thus the inequality above, after a simplification, gives

$$\forall i:\ p_i \leq \sum_j x_{ij}p_j.$$

When we add these inequalities for all i we get

$$\sum_i p_i \leq \sum_i \sum_j x_{ij}p_j.$$

When we interchange the order of summation on the right-hand side, we get

$$\sum_i p_i \leq \sum_j p_j \sum_i x_{ij}.$$

Note that the second summation on the right-hand side is 1, so we get

$$\sum_i p_i \leq \sum_j p_j.$$

This should have been an equality, which means that all the inequalities added to obtain this must have been equalities. This implies two facts, which we are writing as lemmas to be used in the later sections. The theorem follows from the following two lemmas. □

Lemma 2. *Every feasible point of the nonconvex program* (1) *satisfies the constraint of money earned equals money spent for every user; i.e.,*

$$\forall i: \ p_i = \sum_j x_{ij} p_j.$$

Lemma 3. *Every feasible point of the nonconvex program* (1) *satisfies that the money of every person is spent optimally; i.e., whenever $x_{ij} > 0$, the corresponding constraint on the third line of the program is tight.*

In the section addressing the nonlinear utility case, section 7, we will show that Theorem 1 still holds.

4. Solving nonconvex program (1). We will solve the nonconvex program (1) by converting it into a convex program. Note that the third line constraints of the program are not convex. After cross-multiplying, a constraint has the product of variables on one side and a linear variable on the other. In general these constraints are nonconvex.

Also, note that the third line of nonconvex program (1) is useful only when $u_{ij} > 0$. So we can rewrite the third line as

$$\forall i, j \text{ such that } u_{ij} > 0: \ \frac{u_{ij}}{p_j} \leq \frac{\sum_k u_{ik} x_{ik}}{p_i},$$

which in turn can rewritten as

$$\forall i, j \text{ such that } u_{ij} > 0: \ \frac{p_i}{p_j} \leq \frac{\sum_k u_{ik} x_{ik}}{u_{ij}}.$$

Now we construct a directed graph, G, with the n persons as the set of vertices. We draw an edge from i to j when $u_{ij} > 0$. (i and j may be the same vertex; in that case the edge is a loop.) We assign two kinds of weight to each edge, ij. The first is denoted by w, and the second is denoted by $LOGw$. For an edge from i to j, $w(ij) = \frac{\sum_k u_{ik} x_{ik}}{u_{ij}}$ and $LOGw(ij) = \log(w(ij))$. By Farka's lemma, nonconvex program (1) is feasible if and only if the product of w_{ij} is at least 1 over any cycle of the graph, G. In other words we have the following theorem.

Theorem 4. *Nonconvex program* (1) *is feasible if and only if there is no negative cycle in G with respect to the weight function $LOGw$.*

Proof. The problem of finding an equilibrium assignment is finding those x_{ij}'s for which there is a feasible solution of nonconvex program (1). Suppose that we have a feasible assignment of goods to people, i.e., values for variables x_{ij}'s satisfying the first two sets of constraints of nonconvex program (1). We want to find out whether there exists a feasible assignment of values to price variables satisfying the last two sets of constraints in nonconvex program (1). Since x_{ij}'s are given, they form the coefficients of a linear program whose variables correspond to prices. We can apply Farka's lemma to write conditions on the coefficients (i.e., on assignment variables) which allow a feasible solution to price variables. The best way to do this is to take the logarithm of the third set of inequalities and assume that the variables are $\log(p)$ and not p. The theorem now easily follows from Farka's lemma. □

This theorem is similar in flavor to the "no-negative-cycle" theorem for the classical problem of minimum cost flow. In fact, nondeterministically this has the same functionality too. It tells us when an assignment is an equilibrium assignment. A flow is minimum cost if there is no negative cycle. Similarly, an assignment of goods is an equilibrium solution if there is no negative cycle. This analogy gives hope of a combinatorial algorithm for the general equilibrium problem. Later in the paper we will see that the theorem holds for the concave utility functions too.

Theorem 4 also gives us the following convex program for the general equilibrium problem, which again makes it a functional theorem for computational purposes:

$$\begin{aligned}
&\forall j: \ \sum_i x_{ij} = 1,\\
&\forall i,j: \ x_{ij} \geq 0,\\
&\text{For every cycle, } C\text{, of } G \ : \ \prod_{ij \in C} w(ij) \geq 1.
\end{aligned} \tag{2}$$

Remark. Prices do not appear in this convex program. In this sense this convex program is a generalization of the Eisenberg–Gale convex program [17].

The separation oracle for the last set of inequalities can be derived using an algorithm for finding a negative cycle in a graph. Using the usual inequality of the arithmetic mean as at least the geometric mean, the last set of inequalities can also be converted into an infinite number of linear equalities as follows.

In the following program we denote by $\boldsymbol{\alpha}$ a vector of nonnegative real numbers. The number of coordinates will be clear by the context. A subscripted $\boldsymbol{\alpha}$ will denote a coordinate of $\boldsymbol{\alpha}$.

$$\begin{aligned}
&\forall j: \ \sum_i x_{ij} = 1,\\
&\forall i,j: \ x_{ij} \geq 0,\\
&\text{For every cycle, } C\text{, of } G \text{ and for every } \boldsymbol{\alpha}\text{:}\\
&\frac{1}{|C|} \sum_{ij \in C} \frac{w(ij)}{\boldsymbol{\alpha}_{ij}} \geq \left(\frac{1}{\prod_{ij \in C} \boldsymbol{\alpha}_{ij}} \right)^{\frac{1}{|C|}}.
\end{aligned} \tag{3}$$

LEMMA 5. *Convex program* (2) *is equivalent to linear program* (3).

Proof. Note that the third set of inequalities of the linear program follows from the third set of inequalities of the convex program, using the inequality of the arithmetic mean and geometric mean.[5] So we need only show the converse. Suppose that one of

[5] For nonnegative numbers the arithmetic mean is never smaller than the geometric mean.

the inequalities on the third line of the convex program is violated by an assignment vector $\mathbf{x}^*$ of x^*_{ij}. We show that $\mathbf{x}^*$ violates one of the inequalities of the linear program too.

Suppose that the inequality corresponding to a cycle C is violated; i.e., we have

$$\prod_{ij \in C} w^*(ij) < 1,$$

where $\mathbf{w}^*$ denotes the value of the weight function at $\mathbf{x}^*$. We claim that the inequality of the linear program corresponding to the same cycle and $\boldsymbol{\alpha} = \mathbf{w}^*$ is violated too. Indeed, the left-hand side is 1 when evaluated at $\mathbf{x}^*$, whereas the right-hand side is bigger than 1. This also shows that the convex program (2) is indeed convex. □

The following corollary follows from the above theorem using Theorem 12 or 13.

Corollary 6. *The Ellipsoid algorithm finds a market equilibrium in polynomial time.*

Proof. Eaves [15] showed that the problem of finding a market equilibrium with linear utilities can be written as a linear complementarity program. This implies that there is a market equilibrium with rational numbers of polynomial sized denominator. The proof then follows from Theorem 12 or 13. □

Corollary 7. *The set of all possible assignments of goods to people (x_{ij} variables) at equilibria is convex.*

For the purpose of using an Ellipsoid algorithm, linear program (3) does not offer any advantage over the convex program (2). But a linear program can be useful for designing primal-dual algorithms. The infinite size of the linear program should not be a concern in designing a primal-dual algorithm. In the past, exponential sized linear programs were used for designing primal-dual algorithms. A cleverly designed primal-dual algorithm identifies a polynomial number of dual variables to be used.

Convex program (2) is of exponential size, and if it is converted into a linear program, then it is of infinite size. So convex program (2) is not suitable for developing more efficient interior point methods. In the next section we develop a new polynomial size convex program, which gives a promising hope of developing interior point methods.

5. Convex program.

Note that we need to write the third line in the nonconvex program (1) for only those i and j's for which $u_{ij} > 0$. Further note that this implies that $\sum_k u_{ik} x_{ik} > 0$. We already have that $p_i, p_j > 0$. So we can take the log of the whole set of inequalities to get

$$\forall i, j \text{ such that } u_{ij} > 0: \;\; \log(p_i) - \log(p_j) \leq \log\left(\frac{\sum_k u_{ik} x_{ik}}{u_{ij}}\right).$$

We substitute every $\log(p_i)$ with a new variable $LOGp_i$. We then get

$$\forall i, j \text{ such that } u_{ij} > 0: \; LOGp_i - LOGp_j \leq \log\left(\frac{\sum_k u_{ik} x_{ik}}{u_{ij}}\right).$$

Note that log is a concave function; i.e., $\log(\frac{x+y}{2}) \geq \frac{\log(x)+\log(y)}{2}$. This means that if two feasible points satisfy the above inequality, then their average will also satisfy the inequality. So the equivalent convex program that we get for the nonconvex program (1) is

$$\forall j: \sum_i x_{ij} = 1,$$

$$\forall i,j: \; x_{ij} \geq 0,$$

$$\forall i,j \text{ such that } u_{ij} > 0:$$

$$LOGp_i - LOGp_j \leq \log\left(\frac{\sum_k u_{ik}x_{ik}}{u_{ij}}\right). \tag{4}$$

THEOREM 8. *Nonconvex program* (1) *is equivalent to convex program* (4).

COROLLARY 9. *The set of all possible equilibrium prices, on a logarithmic scale (i.e., variables $LOGp_i$), is convex.*

6. General case. In the model section we made an assumption that for every proper subset, S, of persons there are an $i \notin S$ and $j \in S$ such that $u_{ij} > 0$. This assumption is not without loss of generality. We will justify the assumption in this section. In this section we show that even without this assumption there is an equilibrium consisting of only nonzero prices. Hence, convex program (4) remains valid and gives all such equilibria. We also give a way to find other equilibria where some of the prices are zero.

We draw the *nonzero liking graph* of the problem. This graph has a node for every person in the economy. There is a directed edge from i to j whenever $u_{ij} > 0$. If i and j are the same, then we put a loop on i. If this graph is disconnected, i.e., there is a proper subset S such that there is no edge between S and $\bar{S}$ (S complement), then S and $\bar{S}$ can be considered separate economies. One can write the convex programs of two separate economies together and call it a convex program for the joint economy. So we assume that the graph is connected.

If the graph is strongly connected, then it satisfies the assumption and we are done. Else we compute the strongly connected component decomposition of the graph. First, write the convex program for the equilibria of each component. For each component we know that equilibrium prices are all nonzero. Now consider the underlying acyclic structure on the strongly connected components. We say that a component, S_1, is *lower* than another component, S_2, if there is an edge from S_1 to S_2. Take the transitive closure of this *lower* relation. This will be a partial order. Again denote it by *lower.* Note that if S_1 is lower than S_2, then S_2 is not lower than S_1. Hence if the goods in S_1 are nonzero priced, then they cannot move from S_1 to S_2. On the other hand, if goods are heavily priced in S_2, then they cannot move from S_2 to S_1 either. So we find nonzero equilibria for each component. If S_1 is lower than S_2, then we scale up the prices in S_2 by a huge number, so that every person in S_1 likes something in S_1 in comparison with anything in S_2. Hence all the prices are nonzero. For all such equilibrium vectors we can write the convex program (4).

In case one wants to allow zero prices, then note that only a lower ideal can have zero prices. So take any lower ideal and set the zero prices for these. For rest of the economy write the convex program (4). Also note that the corollaries in the previous two sections remain valid whenever they are meaningful.

7. Nonlinear utilities. In this section we explore the case when the utility functions are nonlinear but concave. We assume that the utility functions are differentiable.[6] Let $u_i(x_i)$ be the utility function of i, where x_i is her consumption vector.

[6] If the utility functions are not differentiable, then it may be possible to use subdifferentials instead of differentials.

We assume that u_i is concave, i.e.,

$$\frac{u_i(x_i)+u_i(y_i)}{2} \le u_i\left(\frac{x_i+y_i}{2}\right),$$

for every consumption vector x_i and y_i. Let $u_{ij}(x_i)$ be the partial derivative of u_i at point x_i with respect to the consumption of the jth good (consumption of the jth good by the ith person is denoted by x_{ij}.). Now replace the u_{ij} in the nonconvex program (1) by $u_{ij}(x_i)$, where $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})$. For brevity, $u_{ij}(x_i)$ is written as u_{ij}, since the argument is understood by the context. Now we claim that the nonconvex program (1) is valid for the nonlinear utility's case too.

THEOREM 10. *The feasible region of nonconvex program* (1) *has all and only general market equilibria even if utilities are general differentiable concave functions.*

Proof. First, fix a price vector. At this price whatever a person can potentially buy is a convex set. However, the person will buy that bundle of goods which will maximize her utility. Since the utility function is concave, any local minimum will also be a global minimum, or more generally, the set of maximum utility bundles is a convex set. The conditions of local minima are straightforward; the marginal utility per unit of additional money for all the consumed goods is the same, and for other goods it is no bigger.

This time we will prove the harder part first, i.e., that every feasible solution of the nonconvex program (1) is an equilibrium point. Note that we did not use the fact that u_{ij} is constant when we proved Lemmas 2 and 3. So these lemmas are still valid. Hence the harder side of the theorem follows.

Now we want to prove the easier side. In the linear case, the third line constraints, which represented optimally, were obvious. This time we will also have to use an additional fact that the "money earned is equal to money spent" at the equilibrium. We know that for every i, whenever $x_{ij} > 0$, marginal utility per unit of money is the same; i.e., for every i, the quantity $\frac{u_{ij}}{p_j}$ is the same whenever $x_{ij} > 0$. Since $x_{ij} > 0$, we can say that $\frac{x_{ij}u_{ij}}{x_{ij}p_j}$ is the same. Add all the numerators and denominators together and note that the denominator is p_i by the "money earned equal to money spent" constraint. Hence the easier part of the theorem also follows. $\square$

The above theorem shows that the nonconvex program (1) remains valid. In fact, program (4) is still valid too, but it may not always be convex. For some utility functions, e.g., if the utility of the ith person is $\sum_j \sqrt{x_{ij}}$, then the program (4) is convex. Another interesting case is if the utility of the ith person is $\sum_j \log(1+x_{ij})$; then also the program (4) is convex. The program is convex for $\sum_j \frac{x_{ij}}{1+x_{ij}}$ too. One example where the program is not convex is $\sum_j (1-e^{-x_{ij}})$. So a natural open question is when is program (4) convex. We think that the answer includes the utility functions with the weak gross substitutability property. Unlike in the Fisher case, the answer does not include the homogeneous utility function [21].[7] Gjerstad in [21] gave an example with homogeneous utilities in which the set of market equilibria is not even connected.

It is worth mentioning that program (4) simplifies for the Fisher case. In the Fisher case we do not need to take logs. One can analogously prove that the following program (5) characterizes equilibria in the Fisher case even for nonlinear utilities. It is convex for linear utilities:[8]

[7] Homogeneous function of degree d means that if the bundle is scaled by a factor α, the utility is scaled by a factor α^d.

[8] Note the second line of constraints. The left-hand side is constant. In general, the product of nonnegative linear functions bigger than a constant is a convex constraint.

$$
\begin{aligned}
&\forall j: \sum_i x_{ij} = 1,\\
&\forall i,j : m_i u_{ij} \le p_j \sum_k u_{ik} x_{ik},\\
&\sum_j p_j \le \sum_i m_i,\\
&\forall i,j: x_{ij} \ge 0,\\
&\forall j: p_j \ge 0.
\end{aligned} \tag{5}
$$

x_{ij} has the same meaning as in the program (4). Here m_i is the amount of money the ith buyer has. p_j is a variable for the price of good j.

8. Toward combinatorial algorithms.

The algorithm in this paper is neither combinatorial nor strongly polynomial. As discussed in the introduction algorithms, using convex programming has many theoretical advantages. Convex programs help us understand the problem by revealing basic structures of the problem. The convex program in section 5 is no different. It also leads to an efficient practical algorithm using interior point methods [38]. And as shown in Theorem 4, the convex programs in this paper also lead to a combinatorial characterization of the equilibria. This combinatorial characterization is somewhat passive. When we discover a negative cycle the characterization does not tell us how to fix it. On the other hand, a negative cycle in a minimum cost flow problem tells us how to decrease the cost of flow.

We have such a characterization for the Fisher model using Eisenberg and Gale's linear program [17]. This characterization may help us develop a strongly polynomial combinatorial algorithm or a strongly polynomial time algorithm for the Fisher model. The following theorem, which is an active characterization of equilibria in the Fisher model, may play a role. This characterization tells us not only when an assignment is not in equilibrium, but also how to fix it. To our knowledge there is no other combinatorial characterization known for the equilibria in the Fisher model that can tell us when an assignment is in equilibrium. The algorithm in [12] neither uses nor implies any combinatorial characterization for the problem.

THEOREM 11. *In the Fisher model, consider an assignment x_{ij}'s from goods to buyers. x_{ij} is an equilibrium if and only if there does not exist a good j and two buyers i and i' such that i has a nonzero quantity of good j, and when i gives a sufficiently small but nonzero quantity of good j to i', then the product of $U_i^{m_i} U_{i'}^{m_{i'}}$ increases, where U_i and $U_{i'}$ are the utilities of i and i', and m_i and $m_{i'}$ are their initial endowments of money.*

Proof. The Eisenberg–Gale approach maximizes the $\sum_i m_i \log(\sum_j u_{ij} x_{ij})$ over all feasible allocations denoted by variables x_{ij}'s. Therefore, the forward direction of the theorem is immediately implied by the Eisenberg–Gale convex program [17].

For the reverse direction let x be the assignment which is not in equilibrium. Let us say that x' is the assignment in equilibrium. So Eisenberg and Gale's objective function is higher at x' than x. Consider the straight line segment between x and x'. Let $z = x' - x$. Let $\epsilon \in [0,1]$. The Eisenberg–Gale objective function on any point on this line segment is

$$\sum_i m_i \log\left(\sum_j u_{ij}\left(x_{ij} + \epsilon z_{ij}\right)\right).$$

Note that the Eisenberg–Gale objective function is strictly concave. So its value is strictly higher at positive ϵ than at $\epsilon = 0$. So its derivative at $\epsilon = 0$ is positive.

$$\sum_i \frac{m_i \sum_j u_{ij} z_{ij}}{\sum_j u_{ij} x_{ij}} > 0.$$

Let $w_{ij} = m_i u_{ij} / \sum_{j'} u_{ij'} x_{ij'}$. Note that w_{ij} does not depend upon z. The above inequality can then be written as $\sum_i \sum_j w_{ij} z_{ij}$. We construct a bipartite graph with i's on one side and j's on the other. For every $z_{ij} > 0$, we draw an edge from i to j with (fractional) multiplicity z_{ij}. For every $z_{ij} < 0$, we draw an edge from j to i with (fractional) multiplicity $-z_{ij}$. Note that the bipartite graph is Eulerian on the right-hand side; i.e., the sum of z_{ij}'s on edges incoming to a node is the same as the sum of z_{ij}'s on edges outgoing from the node. This property implies with a simple inductive proof that the above graph can be decomposed into graphs with two edges $i'j$ and ji with the same multiplicity. This means that the weight function, i.e., $\sum_i \sum_j w_{ij} z_{ij}$, can be written as a positive combination of the weight function on these two edge graphs. This implies that there is at least one two-edge graph with positive weight function. Let one such graph be $i'j$ and ji, i.e., $w_{i'j} - w_{ij} > 0$. Note that since z_{ij} is negative, i has some positive quantity of good j. Consider another assignment y which is obtained by x by giving all of i's good j to i'. Now consider the line segment between x and y. Let us consider the derivative of the Eisenberg–Gale objective function in the direction of x to y. This derivative is positive at x. There are two cases. The first is that the derivative remains positive on this line segment; in this case we move to assignment y. The other case is that the derivative is zero somewhere on the line segment, say at y'. In that case we move to y'. In both cases we prove the theorem. □

9. Generalized convex feasibility testing algorithm via ellipsoid and simultaneous diophantine approximation. This section generalizes Theorem 6.4.1 in Grotschel, Lovasz, and Schrijver's book [22]. The theorem says "The strong nonemptiness problem for well-described polyhedra, given by a strong separation oracle,[9] can be solved in oracle-polynomial time."

The theorem makes an assumption of well-described polyhedra which is not true for our convex program or for the Eisenberg–Gale convex program. Well-described polyhedra means that any facet of the polyhedra can be encoded with binary encoded length ϕ. This implies that the coefficients used to describe a facet are rational numbers with binary encoded length ϕ. This also implies that the corner points of the polyhedra also use rational numbers with binary encoded length polynomial in ϕ and n, the dimension of the space. Binary encoded length of a rational number is the sum of the binary encoded lengths of the numerator and the denominator.

The assumption of well-described polyhedra goes naturally with the theorem. Given a point outside the convex-set, the well-described polyhedra assumption guarantees that a hyperplane with small binary encoded length will exist. This may not be true in case the polyhedra is not well described. In the proof of the next theorem we will do away with this assumption by letting the convex set be partly outside the

[9] Strong separation oracle means that if the point lies outside the convex set, then a separating hyperplane could be found where the point lies on one side of the hyperplane and the convex set lies either on the hyperplane or on the other side. Strong separation oracles are different than weak separation in the sense that the point is allowed to be on the side of the convex set but not too far away from the hyperplane.

ellipsoids constructed. Usually ellipsoid algorithms keep the convex set completely inside of any ellipsoid constructed during the execution.

THEOREM 12. *Given a convex set via a strong separation oracle with a guarantee that the set contains a point with binary encoding length at most ϕ, a point in the convex set can be found in polynomial time.*

Proof. In the proof we assume that the convex set contains a point with binary encoded length ϕ, and then we find a point in the convex set. If our algorithm fails, that means the promise that the convex set contains a point with binary encoded length ϕ is false, so the algorithm responds with this answer.

We assume that we do not accidentally hit a point of the given convex set. If we do, we will stop the algorithm and output the answer. Otherwise, consider the box $[-2^\phi, 2^\phi]^n$. This box has all the rational points with encoded length ϕ. Let C be the intersection of this box with the given convex set. A strong separation oracle for the given convex set also implies a strong separation oracle for C. Additionally C is bounded. So we can start with an ellipsoid containing the box. This ellipsoid contains C too. We run the ellipsoid algorithm until the volume of the ellipsoid falls below $1/(2^{2n\phi}n!)$. At this point we know that the set of the rational points of encoded length ϕ in the ellipsoid is not full dimensional.[10] Thus there is a hyperplane of dimension $n-1$ which contains all the rational points of the ellipsoid with binary encoded length ϕ. If we can find this hyperplane (call it H), then we will consider the convex set C to intersect with H.[11] This is one lower dimensional problem and hence by induction we prove the theorem. So now we focus on finding H.

H can be represented by a linear equation. Lemma 6.2.4 in [22] shows that the binary encoded length of H is at most $3n^2\phi$. By multiplying the equation for H by a common denominator, we may assume that the binary encoded length of H is $3n^3\phi$.

Let us continue with the ellipsoid algorithm until the volume of the ellipsoid falls below V. We will determine the value of V later. This means that half of the shortest axis of the ellipsoid is at most $nV^{1/n}$; this is estimated by taking a simple lower bound of $(r/n)^n$ on the volume of an n-sphere of radius r. Let us denote the center of the ellipsoid by v and the unit vector parallel to the shortest axis by w. The hyperplane H' defined by equation $w \cdot x = w \cdot v$ is an exponentially good approximation of H, where $\cdot$ denotes the inner product or dot product. Let u be any point in the ellipsoid; then we have

$$|w \cdot u - w \cdot v| = |w \cdot (u - v)| \le nV^{\frac{1}{n}}.$$

To recover H from H', let us use simultaneous diophantine approximation (Theorem 5.3.19 in [22]) to approximate the vector $(w, w \cdot v)$ by an integer numerator vector (p, π) and a common denominator q, where p has n dimensions and π has one. Let us denote the relative error in the above equation by ϵ, $0 < \epsilon < 1$. Theorem 5.3.19 in [22] says that this can be achieved in polynomial time by choosing the parameters such that

$$1 \le q < 2^{n^2}\epsilon^{-n},$$

$$|w_i q - p_i| < \epsilon \quad \text{and} \quad |w \cdot v - \pi| < \epsilon.$$

[10] A rational point of binary encoded length ϕ has denominator at most 2^ϕ. The difference between two different rational numbers of denominator at most 2^ϕ is at least $2^{-2\phi}$. The volume of a right-angled simplex with unit length edges is $1/(n!)$. Hence $1/(2^{2n\phi}n!)$ is the volume of the right-angled simplex with edges $2^{-2\phi}$.

[11] Note that we may lose some part of the convex set outside the subsequent ellipsoids. This is the reason that we do not need the assumption of a well-described polyhedron.

Now let us take a vector z of binary encoding length ϕ in the ellipsoid. We have

$$\begin{aligned}|p \cdot z - \pi| &\leq |qw \cdot z - qw \cdot v| + \epsilon\,(\|z\|_1 + 1) \leq q|w \cdot z - w \cdot v| + \epsilon n 2^{\phi} \\ &< 2^{n^2} \epsilon^{-n} n V^{\frac{1}{n}} + \epsilon n 2^{\phi}.\end{aligned}$$

Separately let us argue that $|p \cdot z - \pi|$ either is zero or has a lower bound. Note that both p and π are integral and z has denominator at most 2^{ϕ}, so the common denominator of $|p \cdot z - \pi|$ is at most $2^{(n+1)\phi}$. Thus $|p \cdot z - \pi|$ is either zero or at least $2^{-(n+1)\phi}$. Let us choose ϵ so that

$$\epsilon n 2^{\phi} < \frac{1}{2\ 2^{(n+1)\phi}}, \quad \text{i.e.,} \quad \epsilon < \frac{1}{2n2^{(n+2)\phi}}.$$

Let us choose

$$\epsilon = \frac{1}{4n2^{(n+2)\phi}}.$$

With this choice of ϵ, let us choose V so that

$$2^{n^2} \epsilon^{-n} n V^{\frac{1}{n}} < \frac{1}{2\ 2^{(n+1)\phi}}, \quad \text{i.e.,} \quad V < \left(\frac{\epsilon^n}{2n2^{(n+1)\phi + n^2}} \right)^n.$$

We can choose

$$V = \frac{1}{2^{(n+1)^3(\phi+1)} n^{n^2+n}}.$$

This volume is only exponentially small, not superexponentially; i.e., the exponent of two in the denominator is a polynomial in n and linear in ϕ, so the ellipsoid algorithm will achieve this volume in time polynomial in n and linear in ϕ. Note that the formula for the center of the next ellipsoid could give a center with irrational coordinates. We do the usual trick of shifting the center to a suitable close rational point and simultaneously expanding the ellipsoid by a small factor. Even after these changes, the volume of the sequence of ellipsoids shrinks at an exponential rate. □

THEOREM 13. *Given a convex set by a strong separation oracle and a prescribed precision ϕ, there is an oracle-polynomial time and ϕ-linear time algorithm which does one of the following:*

- *concludes that there is no point in the convex set with binary encoded length at most ϕ,*
- *produces a point in the convex set with binary encoded length at most $P(n)\phi$, where $P(n)$ is a polynomial.*

Note that $P(n)$ is a sort of approximation factor, which is an artifact of the exponential approximation factor in the simultaneous diophantine approximation. Finding an algorithm with $P(n) = 1$ or a matter of fact $P(n) = O(1)$ should be a challenging problem.

10. Discussion about primal-dual for Fisher setting market equilibria. In computational economics primal-dual-type algorithms are frequently used. Some recent examples are [27, 12, 28]. The word "type" is used because these algorithms are not similar to traditional linear programming based primal-dual algorithms. There is no duality theory used in these papers. [27] does not use any convex program (although, in hindsight, there is a convex program similar to the Eisenberg–Gale convex program). The situation of [12] is the same as [27]; [12] does not use any convex program, though again in hindsight there is a convex program. [28] uses a convex program but does not really use any duality theorem.

The reason computational economics frequently uses primal-dual-type algorithms is because "auction" is a primal-dual-type algorithm, although found by economists for the purpose of economics. In this regard even the auction-based algorithms by Garg and Kapoor [20] are primal-dual-type algorithms. These recent uses of primal-dual-type algorithms for problems having a convex program in hindsight makes one wonder whether there is a primal-dual schema for convex programs. Jain formally asked the question of developing a primal-dual schema for convex programs in general and Fisher setting market equilibria in particular [24].

Acknowledgments. I would like to thank Laszlo Lovasz for verifying a complicated proof and then inspiring a simplification. I would also like to thank him for providing a reference to his book and making me aware of the need for Theorems 12 and 13. I also want to thank Rakesh Vohra for exchanging many pieces of email and providing important references. Finally, thanks to Bruno Codenotti, Mohammad Mahdian, Vahab S. Mirrokni, Amin Saberi, Kasturi Varadarajan, Vijay V. Vazirani, Santosh Vempalla, and Yinyu Ye for helpful discussions.

REFERENCES

[1] K. Arrow, H. D. Block, and L. Hurwicz, *On the stability of the competitive equilibrium,* II, Econometrica, 27 (1959), pp. 82–109.

[2] K. Arrow and G. Debreu, *Existence of an equilibrium for a competitive economy*, Econometrica, 22 (1954), pp. 265–290.

[3] W. C. Brainard and H. E. Scarf, *How to Compute Equilibrium Prices in* 1891, Cowles Foundation Discussion Paper 1270, New Haven, CT, 2000.

[4] X. Chen and X. Deng, *Settling the complexity of* 2*-player Nash-equilibrium*, in Proceedings of the 5th European Continuous Casting Conference (ECCC 2005), Nice, France, 2005.

[5] B. Codenotti, B. McCune, S. Penumatcha, and K. Varadarajan, *Market equilibrium for exchange CES economies: Existence, multiplicity, and computation*, in Proceedings of the 25th Conference on Foundations of Software Technology and Theoretical Computer Science, Hyderabad, India, 2005, Lecture Notes in Comput. Sci. 3821, Springer, New York, 2005, pp. 505–516.

[6] B. Codenotti, S. Pemmaraju, and K. Varadarajan, *On the polynomial time computation of equilibria for certain exchange economies*, in Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Vancouver, BC, 2005, SIAM, Philadelphia, 2005, pp. 72–81.

[7] B. Codenotti, S. Pemmaraju, and K. Varadarajan, *Algorithms Column: The computation of market equilibria*, ACM SIGACT News, 35 (2004).

[8] R. Cole, Y. Dodis, and T. Roughgarden, *Pricing network edges for heterogeneous selfish users*, in Proceedings of the 35th ACM Symposium on Theory of Computing (STOC 2003), San Diego, CA, 2003, ACM, New York, 2003, pp. 521–530.

[9] K. Daskalakis and C. H. Papadimitriou, *Three-player games are hard*, in Proceedings of the 5th European Continuous Casting Conference (ECCC 2005), Nice, France, 2005.

[10] X. Deng, C. Papadimitriou, and S. Safra, *On the complexity of equilibria*, in Proceedings of the 34th ACM Symposium on Theory of Computing (STOC 2002), Montreal, QC, 2002, ACM, New York, 2002, pp. 67–71.

[11] N. R. Devanur, C. H. Papadimitriou, A. Saberi, and V. V. Vazirani, *Market equilibrium via a primal-dual-type algorithm*, in Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS 2002), Vancouver, BC, 2002, IEEE Computer Society Press, Piscataway, NJ, 2002, pp. 389–395.

[12] N. Devanur, C. Papadimitriou, A. Saberi, and V. Vazirani, *Market Equilibrium via a Primal-Dual-Type Algorithm*, not published as of 2006.

[13] N. R. Devanur and V. V. Vazirani, *An improved approximation scheme for computing Arrow-Debreu prices for the linear case*, in Proceedings of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2003), Mumbai, India, 2003, Lecture Notes in Comput. Sci. 2914, Springer, New York, 2003, pp. 149–155.

[14] S. P. Dirkse and M. C. Ferris, *A pathsearch damped Newton method for computing general equilibria*, Ann. Oper. Res., 68 (1996), pp. 211–232.

[15] B. C. Eaves, *A finite algorithm for the linear exchange model*, J. Math. Econom., 3 (1976), pp. 197–203.

[16] E. Eisenberg, *Aggregation of utility functions*, Management Sci., 7 (1961), pp. 337–350.

[17] E. Eisenberg and D. Gale, *Consensus of subjective probabilities: The pari-mutuel method*, Ann. Math. Statist., 30 (1959), pp. 165–168.

[18] M. C. Ferris, S. P. Dirkse, and A. Meeraus, *Mathematical Programs with Equilibrium Constraints: Automatic Reformulation and Solution via Constrained Optimization*, Numerical Analysis Group Research Report NA-02/11, Oxford University Computing Laboratory, Oxford University, 2002.

[19] L. Fleisher, K. Jain, and M. Mahdian, *Taxes for heterogeneous selfish users in a multicommodity network*, submitted.

[20] R. Garg and S. Kapoor, *Auction algorithms for market equilibrium*, in Proceedings of the 36th ACM Symposium on Theory of Computing (STOC 2004), Chicago, IL, 2004, ACM, New York, 2004, pp. 511–518.

[21] S. Gjerstad, *Multiple Equilibria in Exchange Economies with Homothetic, Nearly Identical Preference*, Discussion Paper 288, Center for Economic Research, University of Minnesota, 1996.

[22] M. Grotschel, L. Lovasz, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, 2nd ed., Springer-Verlag, Berlin, Heidelberg, 1993.

[23] K. Jain, *A Polynomial Time Algorithm for Computing the Arrow-Debreu Market Equilibrium for Linear Utilities*, unpublished manuscript, 2003.

[24] K. Jain, *Polynomial time algorithms for market equilibria*, tutorial in Proceedings of the 6th ACM Conference on Electronic Commerce (EC'05), Vancouver, BC, 2005, ACM, New York, 2005.

[25] K. Jain, M. Mahdian, and A. Saberi, *Approximating market equilibrium*, in Proceedings of the 6th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2003), Princeton, NJ, 2003, Lecture Notes in Comput Sci. 2764, Springer, New York, 2003, pp. 98–108.

[26] K. Jain and K. Varadarajan, *Equilibria for economies with production: Constant-returns technologies and production planning constraints*, in Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, Miami, FL, 2006, SIAM, Philadelphia, 2006, pp. 688–697.

[27] K. Jain and V. V. Vazirani, *Equitable cost allocation via primal-dual-type algorithms*, in Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC 2002), Montreal, QC, 2002, ACM, New York, 2002, pp. 313–321.

[28] K. Jain and V. V. Vazirani, *Eisenberg-Gale Markets: Algorithms and Structural Properties*, manuscript.

[29] K. Jain, V. V. Vazirani, and Y. Ye, *Market equilibria for homothetic, quasi-concave utilities and economies of scale in production*, in Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Vancouver, BC, 2005, SIAM, Philadelphia, 2005, pp. 63–71.

[30] K. Jain, V. V. Vazirani, and Y. Ye, *Market equilibria for homothetic quasi-concave utilities and economies of scale in production*, in Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Vancouver, BC, 2005, SIAM, Philadelphia, PA, 2005, pp. 63–71.

[31] F. P. Kelly, *Charging and rate control for elastic traffic*, European Trans. Telecom., 8 (1997), pp. 33–37.

[32] E. Nenakov and M. E. Primak, *One algorithm for finding solutions of the Arrow-Debreu model*, Kibernetica, 3 (1983), pp. 127–128.

[33] D. J. Newman and M. E. Primak, *Complexity of circumscribed and inscribed ellipsoid methods for solving equilibrium models*, Appl. Math. Comput., 52 (1992), pp. 223–231.

[34] C. H. Papadimitriou, *On the complexity of the parity argument and other inefficient proofs of existence*, J. Comput. System Sci., 48 (1994), pp. 498–532.

[35] C. H. Papadimitriou, *Algorithms, games, and the internet*, in Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing, Heraklion, Crete, 2001, ACM, New York, 2001, pp. 749–753.

[36] M. E. Primak, *A converging algorithm for a linear exchange model*, J. Math. Econom., 22 (1993), pp. 181–187.

[37] L. Walras, *Elements of Pure Economics, or the Theory of Social Wealth*, Lausanne, Paris, 1874 (in French).

[38] Y. Ye, *Computing the Arrow-Debreu competitive market equilibrium and its extensions*, in Proceedings of the 1st International Conference on Algorithmic Applications in Management (AAIM 2005), Lecture Notes in Comput. Sci. 3521, Springer, New York, 2005, pp. 3–5.

SIAM J. Comput.
Vol. 37, No. 1, pp. 319–357

OPTIMAL INAPPROXIMABILITY RESULTS FOR MAX-CUT AND OTHER 2-VARIABLE CSPs?*

SUBHASH KHOT†, GUY KINDLER‡, ELCHANAN MOSSEL§, AND RYAN O'DONNELL¶

Abstract. In this paper we show a reduction from the Unique Games problem to the problem of approximating MAX-CUT to within a factor of $\alpha_{\mathrm{GW}} + \epsilon$ for all $\epsilon > 0$; here $\alpha_{\mathrm{GW}} \approx .878567$ denotes the approximation ratio achieved by the algorithm of Goemans and Williamson in [*J. Assoc. Comput. Mach.*, 42 (1995), pp. 1115–1145]. This implies that if the Unique Games Conjecture of Khot in [*Proceedings of the* 34*th Annual ACM Symposium on Theory of Computing*, 2002, pp. 767–775] holds, then the Goemans–Williamson approximation algorithm is optimal. Our result indicates that the geometric nature of the Goemans–Williamson algorithm might be intrinsic to the MAX-CUT problem. Our reduction relies on a theorem we call Majority Is Stablest. This was introduced as a conjecture in the original version of this paper, and was subsequently confirmed in [E. Mossel, R. O'Donnell, and K. Oleszkiewicz, *Proceedings of the* 46*th Annual IEEE Symposium on Foundations of Computer Science*, 2005, pp. 21–30]. A stronger version of this conjecture called Plurality Is Stablest is still open, although [E. Mossel, R. O'Donnell, and K. Oleszkiewicz, *Proceedings of the* 46*th Annual IEEE Symposium on Foundations of Computer Science*, 2005, pp. 21–30] contains a proof of an asymptotic version of it. Our techniques extend to several other two-variable constraint satisfaction problems. In particular, subject to the Unique Games Conjecture, we show tight or nearly tight hardness results for MAX-2SAT, MAX-q-CUT, and MAX-2LIN(q). For MAX-2SAT we show approximation hardness up to a factor of roughly .943. This nearly matches the .940 approximation algorithm of Lewin, Livnat, and Zwick in [*Proceedings of the* 9*th Annual Conference on Integer Programming and Combinatorial Optimization*, Springer-Verlag, Berlin, 2002, pp. 67–82]. Furthermore, we show that our .943... factor is actually tight for a slightly restricted version of MAX-2SAT. For MAX-q-CUT we show a hardness factor which asymptotically (for large q) matches the approximation factor achieved by Frieze and Jerrum [*Improved approximation algorithms for MAX k-CUT and MAX BISECTION*, in Integer Programming and Combinatorial Optimization, Springer-Verlag, Berlin, pp. 1–13], namely $1 - 1/q + 2(\ln q)/q^2$. For MAX-2LIN(q) we show hardness of distinguishing between instances which are $(1-\epsilon)$-satisfiable and those which are not even, roughly, $(q^{-\epsilon/2})$-satisfiable. These parameters almost match those achieved by the recent algorithm of Charikar, Makarychev, and Makarychev [*Proceedings of the* 38*th Annual ACM Symposium on Theory of Computing*, 2006, pp. 205–214]. The hardness result holds even for instances in which all equations are of the form $x_i - x_j = c$. At a more qualitative level, this result also implies that $1 - \epsilon$ vs. ϵ hardness for MAX-2LIN(q) is *equivalent* to the Unique Games Conjecture.

Key words. MAX-CUT, Unique Games, hardness of approximation, constraint satisfaction

AMS subject classification. 68Q17

DOI. 10.1137/S0097539705447372

*Received by the editors April 4, 2005; accepted for publication (in revised form) February 12, 2007; published electronically May 14, 2007. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

http://www.siam.org/journals/sicomp/37-1/44737.html

†College of Computing, Georgia Tech, Atlanta, GA 30332 (khot@cc.gatech.edu). This work was performed while this author was at the Institute for Advanced Study. This author's material is based upon work supported by the National Science Foundation under grants DMS-0111298 and CCR-0324906.

‡Faculty of Mathematics and Computer Science, Weizmann Institute of Science, 44453 Kfar Saba, Israel (gkindler@weizmann.ac.il). This work was performed while this author was at DIMACS and at the Institute for Advanced Study, Princeton. This author's research was partly supported by CCR grants $\mathcal{N}$CCR-0324906 and $\mathcal{N}$DMS-0111298.

§Department of Statistics, University of California at Berkeley, Berkeley, CA 94720 (mossel@stat.berkeley.edu). This author's research was supported by a Miller Fellowship in Computer Science and Statistics, U.C. Berkeley.

¶Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 (odonnell@cs.cmu.edu).

1. Introduction.

The main result in this paper is a bound on the approximability of the MAX-CUT problem which matches the approximation ratio achieved by the well-known Goemans–Williamson algorithm [27]. The proof of this hardness result relies on the *Unique Games Conjecture* of Khot [39]. We also rely critically on a theorem we call *Majority Is Stablest*, which was introduced as a conjecture in the original version of this paper. For the convenience of the reader, we will now briefly describe these two tools; formal statements appear in sections 3 and 4.

Unique Games Conjecture (roughly). Given a bipartite graph G, a large constant size set of labels $[M]$, and a permutation of $[M]$ written on each edge, consider the problem of trying to find a labeling of the vertices of G from $[M]$ so that each edge permutation is "satisfied," i.e., is consistent with the labeling. The conjecture is that if M is a large enough constant, then it is NP-hard to distinguish instances which are 99% satisfiable from instances which are 1% satisfiable.

Majority Is Stablest theorem (roughly). Let f be a boolean function which is equally often 0 or 1. Suppose the string x is picked uniformly at random and the string y is formed by flipping each bit of x independently with probability η; we call $\Pr[f(x) = f(y)]$ the *noise stability* of f. The theorem states that among all f in which each coordinate has $o(1)$ "influence," the Majority function has the highest noise stability, up to an additive $o(1)$.

We add in passing that the name Majority Is Stablest is a bit of a misnomer in that almost all balanced boolean (weighted) threshold functions are equally noise stable (see Theorem 4). We also note that the Majority Is Stablest theorem has interesting applications outside of this work—to the economic theory of social choice [35] for example—and has already proven useful for other PCP-based inapproximability results [15]. In section 6.3 we mention interesting generalizations of the Majority Is Stablest theorem for q-ary functions, $q > 2$, which are relevant for hardness of approximation and are not resolved in full.

Despite the fact that our hardness result for MAX-CUT relies on the unproven Unique Games Conjecture, we feel it is interesting for several reasons. First, in our opinion it is remarkable that the Unique Games Conjecture should yield a *tight* hardness of approximation ratio for MAX-CUT, and that indeed the best factor should be the peculiar number α_{GW}. It is intriguing that the precise quantity α_{GW} should arise from a noise stability property of the Majority function, and certainly there was previously little evidence to suggest that the Goemans–Williamson algorithm might be optimal.

Another reason we believe our result is interesting is related to this last point. Since the Goemans–Williamson algorithm was published a decade ago, there has been no algorithmic progress on approximating MAX-CUT. Since Håstad's classic inapproximability paper [33] from two years later, there has been no progress on the hardness of approximating MAX-CUT, except for the creation of a better reduction gadget [56]. As one of the most natural and simple problems to have resisted matching approximability bounds, we feel MAX-CUT deserves further investigation and analysis. In particular, we think that regardless of the truth of the Unique Games Conjecture, this paper gives interesting insight into the geometric nature of MAX-CUT. Indeed, insights we have gleaned from studying the MAX-CUT problem in this light have motivated us to give new positive approximation results for variants of other 2-variable CSPs such as MAX-2SAT; see section 9.

Finally, instead of viewing our result as relying on the unproven Unique Games Conjecture, we can view it as being an investigation into the truth of the Unique

Games Conjecture. Indeed, our hardness results for both MAX-CUT and for two-variable linear equations modulo q provide explicit parameters for which the Unique Games Conjecture, if true, must hold. (Note that both problems are Unique Games themselves.) Thus our work gives a target for algorithmic attacks on the Unique Games Conjecture, which if passed will refute it.

Indeed, works subsequent to the original version of this paper have provided approximation algorithms for the Unique Games problem [55, 30, 11] improving on Khot's original algorithm [39]. In particular, in [11] Charikar, Makarychev, and Makarychev gave a semidefinite programming-based approximation algorithm for Unique Games whose approximation factor nearly matches our hardness bound for MAX-2LIN(q). The current situation is therefore that any improvement in the approximation factors for either MAX-CUT or for the more general MAX-2LIN(q) will refute the Unique Games Conjecture.

1.1. Overview of the paper.

In section 2 we describe the MAX-CUT problem and discuss its history. We then state the Unique Games Conjecture in section 3 and discuss very recent algorithm results for the problem. The Majority Is Stablest problem is discussed in section 4, along with its generalization to q-ary domains, $q \geq 2$. We discuss the geometric aspects of MAX-CUT and their connection with the Majority Is Stablest result and the Goemans–Williamson approximation algorithm in section 5. Our main results are stated in section 6. Section 7 is devoted to some technical definitions, preliminaries, and Fourier analytic formulas. In section 8 we prove our main theorem on the hardness of approximating MAX-CUT, based on the Unique Games Conjecture. In section 9 we investigate the approximability of other binary 2-CSPs, such as MAX-2SAT. In section 10 we prove some special cases of the Majority Is Stablest theorem that are of independent interest, with proofs simpler than those in [47]. Finally, section 11 is devoted to extending our techniques to the q-ary domain; we prove some results about noise stability in this domain and then prove our Unique Games-hardness results for MAX-q-CUT and MAX-2LIN(q) and MAX-q-CUT.

2. About MAX-CUT.

The MAX-CUT problem is a classic and simple combinatorial optimization problem: Given a graph G, find the size of the largest cut in G. By a cut we mean a partition of the vertices of G into two sets; the size of the cut is the number of edges with one vertex on either side of the partition. One can also consider a weighted version of the problem in which each edge is assigned a nonnegative weight and the goal is to cut as much weight as possible.

MAX-CUT is NP-complete (indeed, it is one of Karp's original NP-complete problems [38]), and so it is of interest to try to find polynomial time approximation algorithms. For maximization problems such as MAX-CUT, we say an algorithm gives an α-approximation if it always returns an answer which is at least α times the optimal value; we also often relax this definition to allow randomized algorithms which in expectation give α-approximations. Crescenzi, Silvestri, and Trevisan [12] have shown that the weighted and unweighted versions of MAX-CUT have equal optimal approximation factors (up to an additive $o(1)$), and so we pass freely between the two problems in this paper.

The trivial randomized algorithm for MAX-CUT—put each vertex on either side of the partition independently with equal probability—is a 1/2-approximation, and this algorithm is easy to derandomize; Sahni and Gonzalez [50] gave the first 1/2-approximation algorithm in 1976. Following this some $(1/2 + o(1))$-approximation algorithms were given, but no real progress was made until the breakthrough 1994 paper of Goemans and Williamson [27]. This remarkable work used semidefinite programming to achieve an α_{GW}-approximation algorithm, where the constant $\alpha_{\text{GW}} \approx .878567$ is the trigonometric quantity

$$\alpha_{\text{GW}} = \min_{0<\theta<\pi} \frac{\theta/\pi}{(1-\cos\theta)/2}.$$

The minimizing choice of θ here is the solution of $\theta = \tan(\theta/2)$, namely $\theta^* \approx 2.33 \approx 134°$, and $\alpha_{\text{GW}} = \frac{2}{\pi \sin\theta^*}$. The geometric nature of Goemans and Williamson's algorithm might be considered surprising, but as we shall see, this geometry seems to be an inherent part of the MAX-CUT problem.

On the hardness of approximation side, MAX-CUT was proved MAX-SNP hard [49], and Bellare, Goldreich, and Sudan [3] explicitly showed that it was NP-hard to approximate MAX-CUT to any factor higher than 83/84. The hardness factor was improved to $16/17 \approx .941176$ by Håstad [33] via a reduction from MAX-3LIN using a gadget of Trevisan et al. [56]. This stands as the current best hardness result.

Despite much effort and many improvements in the approximation guarantees of other semidefinite programming based algorithms, no one has been able to improve on the algorithm of Goemans and Williamson. Although the true approximation ratio of Goemans and Williamson was proved to be not more than α_{GW} [37, 20] and the integrality gap of their semidefinite relaxation was also proved to be α_{GW} [20], there appears on the face of it to be plenty of possibilities for improvement. Adding triangle constraints and other valid constraints to the semidefinite program has been suggested, alternate rounding schemes have been proposed, and local modification heuristics that work for special graphs have been proven (see, e.g., [27, 19, 18, 37, 57, 17, 20]). And of course, perhaps a completely different algorithm altogether can perform better. Several papers have either explicitly [18] or implicitly [20] given the problem of improving on α_{GW} as an important research goal.

However, in this paper we show that approximating MAX-CUT to within any factor larger than α_{GW} will in fact overturn the Unique Games Conjecture.

3. About the unique games conjecture.

MAX-CUT belongs to the class of constraint satisfaction problems on 2 variables (2-CSPs). In a k-CSP we are given a set of variables and a set of constraints, where each constraint depends on exactly k variables. The goal is to find an assignment to the variables so as to maximize the number of constraints satisfied. In the case of MAX-CUT, the vertices serve as variables and the edges as constraints. Every constraint says that two certain variables should receive different boolean values.

Proving inapproximability results for a k-CSP is equivalent to constructing a k-query PCP with a specific acceptance predicate. Usually the so-called Label Cover problem is a starting point for any PCP construction. Label Cover is a 2-CSP where the variables range over a large (non-boolean) domain. Usually, inapproximability results for boolean CSPs are obtained by encoding assignments to Label Cover variables via a binary code and then running PCP tests on the (supposed) encodings. This approach has been immensely successful in proving inapproximability results for k-CSPs with $k \geq 3$ (see, for example, [33, 51, 31]). However, the approach gets stuck in the case of 2-CSPs. We seem to have no techniques for constructing boolean 2-query PCPs, and the bottleneck seems to be the lack of an appropriate PCP "outer verifier."

Khot suggested the Unique Games Conjecture in [39] as a possible direction for proving inapproximability results for some important 2-CSPs, such as Min-2SAT-

Deletion, Vertex Cover, Graph-Min-Bisection, and MAX-CUT. This conjecture asserts the hardness of the "Unique Label Cover" problem.

Definition 1. *The Unique Label Cover problem,* $\mathcal{L}(V, W, E, [M], \{\sigma_{v,w}\}_{(v,w)\in E})$, *is defined as follows: Given is a bipartite graph with left-side vertices* V, *right-side vertices* W, *and a set of edges* E. *The goal is to assign one "label" to every vertex of the graph, where* $[M]$ *is the set of allowed labels. The labeling is supposed to satisfy certain constraints given by bijective maps* $\sigma_{v,w} : [M] \to [M]$. *There is one such map for every edge* $(v, w) \in E$. *A labeling "satisfies" an edge* (v, w) *if*

$$\sigma_{v,w}(\text{label}(w)) = \text{label}(v).$$

The optimum OPT of the Unique Label Cover problem is defined to be the maximum fraction of edges satisfied by any labeling.

The Unique Label Cover problem is a special case of the Label Cover problem. It can also be stated in terms of 2-Prover-1-Round Games, but the Label Cover formulation is easier to work with. The Unique Games Conjecture asserts that this problem is hard.

Unique Games Conjecture: *For any* η, $\gamma > 0$, *there exists a constant* $M = M(\eta, \gamma)$ *such that it is NP-hard to distinguish whether the Unique Label Cover problem with label set of size* M *has optimum at least* $1 - \eta$ *or at most* γ.

The Unique Games Conjecture asserts the existence of a powerful outer verifier that makes only two queries (albeit over a large alphabet) and has a very specific acceptance predicate: for every answer to the first query, there is exactly one answer to the second query for which the verifier would accept, and vice versa. Once we have such a powerful outer verifier, we can possibly construct a suitable inner verifier and prove the desired inapproximability results. Typically, though, the inner verifier will need to rely on rather deep theorems about the Fourier spectrum of boolean functions, e.g., the theorem of Bourgain [8] or of Friedgut [23].

The Unique Games Conjecture was used in [39] to show that Min-2SAT-Deletion is NP-hard to approximate within any constant factor. The inner verifier is based on a test proposed by Håstad [32] and on Bourgain's theorem. It is also implicit in this paper that the Unique Games Conjecture with an additional "expansion-like" condition on the underlying bipartite graph of the Label Cover problem would imply that Graph-Min-Bisection is NP-hard to approximate within any constant factor. Khot and Regev [40] showed that the conjecture implies that Vertex Cover is NP-hard to approximate within any factor less than 2. The inner verifier in their paper is based on Friedgut's theorem and is inspired by the work of Dinur and Safra [16], which showed 1.36 hardness for Vertex Cover. In the present paper we continue this line of research, showing an inner verifier that together with the Unique Games Conjecture yields a tight hardness result for MAX-CUT. Our inner verifier relies critically on the Majority Is Stablest theorem.

Algorithmic results for Unique Label Cover. It is natural to ask how the function $M(\eta, \gamma)$ in the Unique Games Conjecture can behave. Lower bounds on M are obtained by giving algorithms for Unique Label Cover. Several very recent results have provided such algorithms. Most relevant for this paper is the algorithm of [11], which has the following behavior for Unique Label instances with label set of size q: For any constant $\eta > 0$, on instances with optimum $1 - \eta$ it satisfies roughly a $(1/q)^{\eta/(2-3\eta)}$ fraction of edges, up to lower order powers of q. Also, for $\eta = 1/\log q$, it seems to satisfy an $\Omega(1)$ fraction of edges.

4. About the Majority Is Stablest problem. To state the Majority Is Stablest problem, we need some definitions. For convenience we regard the boolean values as -1 and 1 rather than 0 and 1. Thus a boolean function is a map $f\colon \{-1,1\}^n \to \{-1,1\}$. We will often generalize to the case of functions $f\colon \{-1,1\}^n \to \mathbb{R}$. In all of what follows, we consider the set of strings $\{-1,1\}^n$ to be a probability space under the uniform distribution.

First, we recall the well-known notion of "influence," introduced to computer science in [4] and studied even earlier in economics.

Definition 2. *Let $f\colon \{-1,1\}^n \to \mathbb{R}$. Then the* influence of x_i on f *is defined by*

$$\mathrm{Inf}_i(f) = \mathop{\mathbf{E}}_{(x_1,\dots,x_{i-1},x_{i+1},\dots,x_n)} \left[\mathrm{Var}_{x_i}[f]\right].$$

(Note that for $f\colon \{-1,1\}^n \to \{-1,1\}$,

$$\mathrm{Inf}_i(f) = \mathop{\Pr}_{x\in\{-1,1\}^n}[f(x) \neq f(x_1,\dots,-x_i,\dots x_n)].)$$

Instead of picking x at random, flipping one bit, and seeing if this changes the value of f, we can instead flip a constant fraction (in expectation) of the bits. This leads to the study of "noise sensitivity," pioneered in computer science by [34, 33, 5].

Definition 3. *Let $f\colon \{-1,1\}^n \to \mathbb{R}$ and let $-1 \le \rho \le 1$. The* noise stability of f at ρ *is defined as follows: Let x be a uniformly random string in $\{-1,1\}^n$ and let y be a "ρ-correlated" copy; i.e., pick each bit y_i independently so that $\mathbf{E}[x_iy_i] = \rho$. Then the noise stability is defined to be*

$$\mathbb{S}_\rho(f) = \mathbf{E}_{x,y}[f(x)f(y)].$$

(Note that for $f\colon \{-1,1\}^n \to \{-1,1\}$ we have $\mathbb{S}_\rho(f) = 2\Pr_{x,y}[f(x) = f(y)] - 1$.)

We may now state the Majority Is Stablest theorem. This result was presented as a strongly believed conjecture in the original version of this paper. It has recently been proved in [47]. Informally, the theorem says that among all balanced boolean functions with small influences, the Majority function has the highest noise stability. Note that the assumption of small influences is necessary since the "dictator" function $f(x) = x_i$ provably has the highest noise stability among all balanced boolean functions for every ρ. Note that when n tends to infinity, the noise stability at ρ of the n-bit Majority function approaches $(1 - \frac{2}{\pi}\arccos\rho)$ (this fact was stated in a paper of Gulibaud from the 1960's [29] and is ultimately derived from the central limit theorem plus a result from an 1890's paper of Sheppard [52]). Thus we have the formal statement of the theorem.

Majority Is Stablest theorem: *Fix $\rho \in [0,1)$. Then for any $\epsilon > 0$ there is a small enough $\delta = \delta(\epsilon,\rho) > 0$ such that if $f : \{-1,1\}^n \to [-1,1]$ is any function satisfying $\mathbf{E}[f] = 0$ and $\mathrm{Inf}_i(f) \le \delta$ for all $i = 1\dots n$, then*

$$\mathbb{S}_\rho(f) \le 1 - \tfrac{2}{\pi}\arccos\rho + \epsilon.$$

In the remainder of this section, we shall describe why the Majority Is Stablest theorem is relevant for MAX-CUT inner verifiers.

As described in the previous section, inapproximability results for many problems are obtained by constructing a tailor-made PCP; usually, the PCP is obtained by composing an "outer verifier" (almost always a Label Cover problem) with an "inner verifier." As mentioned, the outer verifier for our reduction is the Unique Label Cover problem. As for the inner verifier, it is always application-specific and its acceptance predicate is tailor-made for the problem at hand, in our case MAX-CUT.

A *codeword test* is an essential submodule of an inner verifier. It is a probabilistic procedure for checking whether a given string is a codeword of an error-correcting code, most commonly the "Long Code" (see [3]).

DEFINITION 4. *The Long Code over domain* $[n]$ *is a binary code in which the message space is in fact the set of truth tables of boolean functions* $f : \{-1, 1\}^n \to \{-1, 1\}$. *The codeword encoding the "message"* $i \in [n]$ *is given by the* i*th dictator function; i.e., the function* $f(x_1, x_2, \ldots, x_n) = x_i$.

A codeword test for the Long Code can often be extended to a full-fledged inner verifier. So in the following, we will focus only on a Long Code test. The choice of the test is determined by the problem at hand, in our case MAX-CUT. The test must read two bits from a Long Code and accept iff the values read are distinct. Note that a legal Long Code word, i.e., a dictator, is the truth table of a boolean function in which one coordinate has influence 1. Let us say that a function f is *far from being a Long Code* if all the coordinates have $o(1)$ influences (note that this is not a standard notion of being far from a codeword but rather a notion tailored for our proof technique).

We expect the following from a codeword test: a correct Long Code word passes the test with probability c (called the "completeness" parameter of the test), whereas any function far from being a Long Code passes the test with probability at most s (called the "soundness" parameter). Once we construct a full-fledged inner verifier, the ratio s/c will be the inapproximability factor for MAX-CUT.

The Long Code test. As mentioned before, our Long Code test will need to take a boolean function $f : \{-1, 1\}^n \to \{-1, 1\}$, pick two inputs x and y, and check that $f(x) \neq f(y)$. In fact, our test will be precisely a "noise stability" test for some fixed noise rate ρ; i.e., x will be chosen uniformly at random and y will be formed by flipping each bit of x independently with probability $\frac{1}{2} - \frac{1}{2}\rho$. Here ρ will be a value between -1 and 0, and therefore y is a *highly* noisy version of x, or alternatively, a moderately noisy version of $-x$. Thus (at least for legal Long Code words) we expect $f(x)$ to be quite *anticorrelated* with $f(y)$; i.e., it should pass the test with relatively high probability. Recalling Definition 3, we see that the probability a given function f passes our test is precisely $\frac{1}{2} - \frac{1}{2}\mathbb{S}_\rho(f)$.

A legal Long Code word, i.e., a dictator function, has noise stability precisely ρ, and thus the completeness of the Long Code test is $c = \frac{1}{2} - \frac{1}{2}\rho$. The crucial aspect of our test is the analysis of the soundness parameter.

This is where the Majority Is Stablest theorem comes in. Suppose $f \colon \{-1, 1\}^n \to \{-1, 1\}$ is any function that is far from being a Long Code word. By a simple trick (see Proposition 5), we can show that the Majority Is Stablest theorem (which is stated only for $\rho \geq 0$) implies that for $\rho < 0$ the noise stability of f at ρ is at *least* $1 - \frac{2}{\pi} \arccos \rho$ (a negative number). Hence it follows that functions that are far from being a Long Code pass the test with probability *at most* $s = \frac{1}{2} - \frac{1}{2}(1 - \frac{2}{\pi} \arccos \rho) = (\arccos \rho)/\pi$.

Choosing $\rho < 0$ as we please, this leads to an inapproximability ratio of

$$\frac{s}{c} = \min_{-1<\rho<0} \frac{(\arccos \rho)/\pi}{\frac{1}{2} - \frac{1}{2}\rho} = \min_{0 \leq \theta \leq \pi} \frac{\theta/\pi}{(1 - \cos \theta)/2} = \alpha_{\mathrm{GW}},$$

precisely the Goemans–Williamson constant.

4.1. History of the Majority Is Stablest problem. There has been a long line of work in the analysis of boolean functions studying the noise sensitivity of functions and the associated Fourier-theoretic quantities (some examples, roughly in chronological order: [34, 9, 53, 24, 54, 10, 23, 5, 7, 8, 25, 35, 46, 48, 14]). Building on the intuition gathered from this past work, we were motivated to make the Majority Is Stablest conjecture in the original version of the paper. We discuss these relevant previous results below.

The Majority and weighted majority (or balanced threshold) functions have always played an important role in the study of noise sensitivity of boolean functions. This family of functions is, in a sense, the set of all "uniformly noise-stable" functions. In [5], it is shown that a family of monotone functions is asymptotically noise sensitive iff it is asymptotically orthogonal to the family of balanced threshold functions; by asymptotically noise sensitive functions it is meant those that have $\mathbb{S}_\rho(f) = o(1)$ for any constant ρ.

Stated in terms of Fourier coefficients (see section 7.2), the Majority Is Stablest theorem says that among all "nonjunta-like" functions, the one which has most Fourier mass on the lower levels is the Majority function. This is because $\mathbb{S}_\rho(f)$ is just a weighted sum of the squared Fourier coefficients of f, where coefficients at level k have weight ρ^k. Some strong evidence in favor of the Majority Is Stablest theorem was given by Bourgain [8], who showed that nonjunta functions f have their Fourier tails $\sum_{|S|>k} \hat{f}(S)^2$ lower bounded by $k^{-1/2-o(1)}$. As Bourgain noted, the Majority function has precisely this tail decay, and thus his theorem is "basically" optimal. In other words, Majority has the "least" Fourier weight on higher levels and therefore the "most" Fourier weight on lower levels.

The expression $\mathbb{S}_{-1/3}(f)$ played a central role in a Fourier-theoretic approach to the Condorcet paradox and Arrow's theorem given by Kalai [35]. This expression determines the probability of an "irrational outcome" in a certain voting scheme. Much of [35] is devoted to the study of $\mathbb{S}_{-1/3}(f)$, and in particular, it is conjectured there (Conjecture 5.1) that for "transitive" functions, which have the property that all influences are the same, the sum $\sum_{|S|\leq k} \hat{f}(S)^2$ is maximized by the Majority function for all k. Although this conjecture turns out to be false [47], the corollaries of the conjecture in [35] are implied by the fact that Majority is the stablest transitive function, and this is a consequence of the Majority Is Stablest theorem.

Finally, in [48] it was shown that Majority is essentially the maximizer for another noise stability problem, namely maximizing the kth norm of $T_\rho f$, where T_ρ is the Bonami–Beckner operator (see section 7) among balanced functions f for large k and $n = \infty$.

In the original version of this paper, when Majority Is Stablest was still a conjecture, some special cases of the problem were proven. Since these proofs are much simpler than those in [47], and since the proofs have already proven to be of independent interest (see [44] for use of all three), we have included these partial results in section 10.

4.2. Generalizations to the q-ary domain. Our methods can also be used to obtain hardness results for constraint satisfaction problems over variables ranging over larger domains $[q]$. In the q-ary regime we need a multivalued analogue of the Majority Is Stablest theorem. Before we can formulate the appropriate analogue, we need to specify what we mean by "q-ary functions" and also to define the notions of noise stability and influences for them.

The obvious generalization of a boolean function to the q-ary regime would be a function of the form $f : [q]^n \to [q]$. However, as we did for boolean functions, we will consider a continuous relaxation of the range. Specifically, define

$$\Delta_q = \left\{(x_1, \dots, x_q) \in [0,1]^q : \sum x_i = 1\right\},$$

which can be thought of as the space of probability distributions over $[q]$. We will consider functions $f : [q]^n \to \Delta_q$; this generalizes functions $f : [q]^n \to [q]$ if we identify the elements $a \in [q]$ in f's range with the points $(0, \dots, 0, 1, 0, \dots, 0) \in \Delta_q$.

Definition 5. *Let* $-\frac{1}{q-1} \leq \rho \leq 1$ *and let* x *and* y *be* $[q]^n$*-valued random variables. We say that* x *and* y *are a* ρ-correlated pair *if* x *is uniformly distributed on* $[q]^n$*, and* y *is formed from* x *by choosing each* y_i *so that* $\Pr[y_i = a] = \delta_{\{x_i = a\}}\rho + \frac{1-\rho}{q}$ *for each* a*, independently for each* i*. Note that for* $0 \leq \rho \leq 1$*, it is equivalent to say that each coordinate* y_i *is independently chosen to be* x_i *with probability* ρ *and is a uniformly random element of* $[q]$ *otherwise.*

Definition 6. *Let* $f : [q]^n \to \Delta_q$ *and let* $-\frac{1}{q-1} \leq \rho \leq 1$*. The* noise stability of f at ρ *is defined to be*

$$\mathbb{S}_\rho(f) = \mathop{\mathbf{E}}_{x,y}[\langle f(x), f(y)\rangle],$$

where x *and* y *are a* ρ*-correlated pair. Equivalently, we may define the noise stability of functions* $g : [q]^n \to \mathbb{R}$ *via*

$$\mathbb{S}_\rho(g) = \mathop{\mathbf{E}}_{x,y}[g(x)g(y)],$$

and then denoting by f^i *the* i*th coordinate projection of* f*, we have* $\mathbb{S}_\rho(f) = \sum_{i=1}^n \mathbb{S}_\rho(f^i)$.

We remark that when f's range is simply $[q]$ (as embedded in Δ_q), the quantity $\mathbb{S}_\rho(f)$ is simply the probability that $f(x) = f(y)$ when x and y are a ρ-correlated pair.

The definition of influences is very similar to that in the boolean case.

Definition 7. *Let* $f : [q]^n \to \Delta_q$*. For* $1 \leq i \leq n$*, the* influence of the ith coordinate on f *is defined to be*

$$\mathrm{Inf}_i(f) = \mathop{\mathbf{E}}_{x_1,\dots,x_{i-1},x_{i+1},\dots,x_n}[\mathrm{Var}_{x_i}[f(x_1, \dots, x_n)]],$$

where $\mathrm{Var}[f]$ *denotes* $\mathbf{E}[\langle f, f\rangle] - \langle \mathbf{E}[f], \mathbf{E}[f]\rangle$.

We say that $f : [q]^n \to \Delta_q$ is "balanced" if $\mathbf{E}[f^i] = 1/q$ for each i. The most obvious generalization of the Majority function to the q-ary domain is the Plurality function, which on input $x \in [q]^n$ outputs the most common value for x_i (tie-breaking is unimportant). It is natural to ask whether a "Plurality Is Stablest" theorem holds. This question is still open, and we present it as a conjecture. For this purpose, define

$$\mathrm{PlurStab}(q, \rho) = \lim_{n\to\infty} \mathbb{S}_\rho(\mathrm{Plurality}_{n,q}).$$

The limit in the formula above indeed exists, and there appears to be no closed formula for it; however, we provide an exact description of it in Theorem 7 in section 6.

Plurality Is Stablest Conjecture: *Fix* $q \geq 2$ *and* $-\frac{1}{q-1} \leq \rho \leq 1$*. Then for any* $\epsilon > 0$ *there is a small enough* $\delta = \delta(\epsilon, \rho, q)$ *such that if* $f : [q]^n \to [q]$ *is any balanced* q*-ary function with* $\mathrm{Inf}_i(f) \leq \delta$ *for all* $i = 1 \dots n$*, then*

$$\mathbb{S}_\rho(f) \leq \mathrm{PlurStab}(q, \rho) + \epsilon.$$

Note that in the case $q = 2$, Sheppard's formula gives $\mathrm{PlurStab}(2, \rho) = 1 - \frac{2}{\pi} \arccos \rho$, which is the noise stability of Majority; there is also a closed formula for $q = 3$ [28, 13]. For large values of q, we give asymptotics which hold up to a $1 + o_q(1)$ factor in section 6. For the reader's convenience, we remark here that

$$\mathrm{PlurStab}(q, \rho) = \tilde{\Theta}\left((1/q)^{(1-\rho)/(1+\rho)}\right).$$

Although we do not have Plurality Is Stablest, a result of [47] generalizing Majority Is Stablest serves us almost equally well. This result bounds the stability of a function in terms of the behavior of correlated Gaussians. To state it, we need one more definition.

DEFINITION 8. *Let $\mu \in [0,1]$ and $\rho \in [0,1]$. Let X and Y denote normal random variables with mean* 0 *and covariance matrix $\left(\begin{smallmatrix} 1 & \rho \\ \rho & 1 \end{smallmatrix}\right)$. We define*

$$\Lambda_\rho(\mu) = \Pr[X \geq t \text{ and } Y \geq t],$$

where t is chosen so that $\Pr[X \geq t] = \mu$.

MOO theorem: *Fix $q \geq 2$ and $\rho \in [0, 1)$. Then for any $\epsilon > 0$ there is a small enough $\delta = \delta(\epsilon, \rho, q) > 0$ such that if $f : [q]^n \to [0, 1]$ is any function satisfying* $\mathbf{E}[f] = \mu$ *and* $\mathrm{Inf}_i(f) \leq \delta$ *for all $i = 1 \ldots n$, then*

$$\mathbb{S}_\rho(f) \leq \Lambda_\rho(\mu) + \epsilon.$$

As a result we have that the noise stability of any balanced $f : [q]^n \to \Delta_q$ is essentially at most $q\Lambda_\rho(1/q)$. We give the asymptotics of this quantity in section 6, and they are extremely close to those of $\mathrm{PlurStab}_\rho(q)$; in particular, they are the same up to a constant multiplicative factor.

5. On the geometry of MAX-CUT. We shall now try to explain (nonrigorously) the connection between the Majority Is Stablest theorem and the geometric picture that arises from the Goemans–Williamson algorithm. But before going further, let us first note that the approximation ratio achieved by Goemans and Williamson arises as the solution of a trigonometric minimization problem, which in turn originates from a geometric setting. To obtain a matching inapproximability constant, it seems essential to introduce some similar geometric structure. Such a structure is present in the construction of our Long Code test, although it is implicit only in the actual proofs.

For the purposes of the following explanation, let us consider the n-dimensional discrete cube $\{-1, 1\}^n$ as a subset of the n-dimensional Euclidean unit sphere (we normalize the Euclidean norm accordingly). The Majority Is Stablest theorem essentially states that the discrete cube is a good approximation of the sphere in a certain sense.

The Goemans–Williamson algorithm. We start with a brief description of how the approximation ratio α_{GW} arises in the Goemans–Williamson algorithm. To find a large cut in a given graph $G = (V, E)$ with n vertices, the Goemans–Williamson algorithm embeds the graph in the unit sphere of $\mathbb{R}^n$, identifying each vertex $v \in V$ with a unit vector $\mathbf{x}_v$ on the sphere. The embedding is selected such that the sum

$$\sum_{(u,v) \in E} \frac{1}{2} - \frac{1}{2}\langle \mathbf{x}_u, \mathbf{x}_v \rangle, \tag{1}$$

involving the inner products of vectors associated with the endpoints of edges of G, is maximized. The maximum sum bounds from above the size of the maximum cut, since the size of every cut can be realized by associating all the vertices from one side of the cut with an arbitrary point $\mathbf{x}$ on the sphere, and associating all other vertices with $-\mathbf{x}$.

Once the embedding is set, a cut in G is obtained by choosing a random hyperplane through the origin and partitioning the vertices according to the side of the hyperplane on which their associated vectors fall. For an edge (u, v) in G, the probability that u and v lie on opposite sides of the random cut is proportional to the angle between $\mathbf{x}_u$ and $\mathbf{x}_v$. More precisely, letting $\rho = \langle \mathbf{x}_u, \mathbf{x}_v \rangle$ denote the inner product between the vectors associated with u and v, the probability that the edge (u, v) is cut is $(\arccos \rho)/\pi$.

The approximation ratio α_{GW} of the Goemans–Williamson algorithm is obtained by noting that

$$\alpha_{\text{GW}} = \min_{-1 \le \rho \le 1} \frac{(\arccos \rho)/\pi}{\frac{1}{2} - \frac{1}{2}\rho} \approx .878567 \tag{2}$$

is the smallest ratio possible between the probability of an edge being cut and its contribution to (1). Hence the expected size of the cut obtained by the Goemans–Williamson algorithm is at least an α_{GW}-fraction of (1), and therefore it is also at least an α_{GW}-fraction of the maximum cut in G.

Cutting the sphere. In [20], Feige and Schechtman considered the graph G_ρ whose vertices are all the vectors on the unit sphere and in which two vertices are connected by an edge in G_ρ iff their inner product is roughly ρ (we do not get into the precise details). It is shown in [20] that in this graph the largest cut is obtained by any hyperplane through the origin. (To state this rigorously, one should define appropriate measures, etc., but let us remain at a simplistic level for this discussion.) Such a hyperplane cuts an $(\arccos \rho)/\pi$-fraction of the edges in the graph.

Restricting to the cube. We would like to consider an edge-weighted graph H_ρ which is, in a nonrigorous sense, the graph induced by G_ρ on the discrete hypercube. For two vectors $\mathbf{x}, \mathbf{y}$ on the discrete cube, we define the weight of the edge $(\mathbf{x}, \mathbf{y})$ to be

$$\Pr[X = \mathbf{x} \text{ and } Y = \mathbf{y}],$$

where X and Y are ρ-correlated random elements of the discrete cube. The graph H_ρ resembles G_ρ in the sense that almost all the edge weight in H_ρ is concentrated on edges $(\mathbf{x}, \mathbf{y})$ for which $\langle \mathbf{x}, \mathbf{y} \rangle \approx \rho$; we call such edges *typical edges.* Let us examine how good H_ρ is as an "approximation" of the graph G_ρ.

Note that the structure of H_ρ is very reminiscent of our Long Code test, mentioned above. To make the similarity even clearer, note that a cut C in H_ρ immediately defines a boolean function f_C over the discrete cube. It is easy to observe that the size of C (namely the sum of weights of the edges that are cut) is exactly the noise stability of f_C—i.e., the acceptance probability of the Long Code test with parameter ρ when applied to f_C.

The size of the cut. So how large can the size of C be? If C is determined by a random hyperplane, then a typical edge is cut with probability about $(\arccos \rho)/\pi$. The expected size of such a cut is therefore roughly the same as the weight of the maximal cut in G_ρ (when the total weight of the edges in G_ρ is normalized to 1).

There are, however, cuts in H_ρ whose weight is larger than $(\arccos\rho)/\pi$. For example, one can partition the vertices in H_ρ according to their first coordinate, taking one side of the cut C to be the set of vectors in the discrete cube whose first coordinate is 1 and the other side of C to be the set of vectors whose first coordinate is -1; note that this is the cut defined by the hyperplane which is perpendicular to the first coordinate. When interpreted as a function, C corresponds to the function $f_C(x) = x_1$; i.e., it is a correct Long Code word. One can easily observe that the size of C is $\frac{1}{2} - \frac{1}{2}\rho$—i.e., it is exactly the completeness of the Long Code test with parameter ρ.

The Majority Is Stablest theorem comes in. The size of one-coordinate cuts in H_ρ is larger than the best cuts achievable in G_ρ. The Majority Is Stablest theorem implies, however, that essentially those are the only special cases, and that all other cuts in H_ρ are no larger than the maximum cut in G_ρ. That is, it implies that unless f_C depends significantly on one of the coordinates, then the size of C is at most $(\arccos\rho)/\pi + \epsilon$. Stated formally, Proposition 5 in section 7.3 says the following.

PROPOSITION. *For any $\rho \in (-1, 0]$ and any $\epsilon > 0$ there is a small enough $\delta = \delta(\epsilon, \rho) > 0$ such that if C is a cut in H_ρ such that $\mathrm{Inf}_i(f_C) \le \delta$ for every i, then the size of C is at most $(\arccos\rho)/\pi + \epsilon$*

6. Our results.

In this section we formally state our main results.

6.1. Hardness for MAX-CUT and 2-bit CSPs.

Our main result regarding MAX-CUT is the following.

THEOREM 1. *Assume the Unique Games Conjecture. Then for every constant $-1 < \rho < 0$ and $\epsilon > 0$, it is NP-hard to distinguish instances of MAX-CUT that are at least $(\frac{1}{2} - \frac{1}{2}\rho)$-satisfiable from instances that are at most $((\arccos\rho)/\pi + \epsilon)$-satisfiable. In particular, choosing $\rho = \rho^*$, where*

$$\rho^* = \operatorname*{argmin}_{-1<\rho<0} \frac{(\arccos\rho)/\pi}{\frac{1}{2} - \frac{1}{2}\rho} \approx -.689,$$

implies that it is NP-hard to approximate MAX-CUT to within any factor greater than the Goemans–Williamson constant $\alpha_{GW} \approx .878567$.

Recall that the main result of Goemans and Williamson [27] is an algorithm which, given instances of MAX-CUT with fractional optimum at least $\frac{1}{2} - \frac{1}{2}\rho$ (where $\rho \le \rho^*$), outputs a solution with value at least $(\arccos\rho)/\pi - \epsilon$ (where $\epsilon > 0$ can be an arbitrarily small constant). Thus our Unique Games-hardness theorem precisely matches the algorithmic guarantee of Goemans and Williamson for *all* $-1 < \rho \le \rho^*$. For ρ very close to -1, by considering the Taylor expansion $\arccos\rho = \pi/2 - \rho - \rho^3/6 - \cdots$, we have the following corollary.

COROLLARY 1. *Assume the Unique Games Conjecture. Then for all sufficiently small $\eta > 0$, it is NP-hard to distinguish instances of MAX-CUT that are at least $(1 - \eta)$-satisfiable from instances that are at most $(1 - (2/\pi)\sqrt{\eta})$-satisfiable.*

We prove Theorem 1 in section 8.

In section 9 we apply our techniques for other 2-bit CSPs besides MAX-CUT. In particular we prove the following.

THEOREM 2. *Assume the Unique Games Conjecture. Then it is NP-hard to approximate MAX-2SAT to within any factor greater than β, where*

$$\beta = \min_{\frac{\pi}{2} \le \theta \le \pi} \frac{2 + (2/\pi)\theta}{3 - \cos\theta} \approx .943.$$

The proof of Theorem 2 actually implies that MAX-2SAT is hard to approximate to within any factor greater than β, even if restricted to instances where each variable appears equally often positively and negatively (see section 9 for more details). We show that for this restricted problem, called Balanced-MAX-2SAT, the approximation bound β is tight; i.e., it *can* be approximated to within any factor smaller than β.

THEOREM 3. *Balanced-MAX-2SAT is polynomial-time approximable to within any factor smaller than* β.

6.2. Special cases of the Majority Is Stablest theorem. Some special cases of the Majority Is Stablest theorem are of independent interest.

First, it should be noted that the Majority function is not a "unique" optimizer, in the sense that every weighted threshold that does not depend largely on any one coordinate is equally noise-stable.

THEOREM 4. *Let* $f : \{-1,1\}^n \to \{-1,1\}$ *be any balanced threshold function, namely of the form* $f(x) = \operatorname{sgn}(a_1x_1 + \cdots + a_nx_n)$. *Let* $\delta = \max_i \{\operatorname{Inf}_i(f)\}$. *Then for all* $\rho \in [-1,1]$,

$$\mathbb{S}_\rho(f) = 1 - \tfrac{2}{\pi}\arccos\rho \pm O(\delta(1-|\rho|)^{-3/2}).$$

It is also of interest to consider the case where ρ tends to zero. It is easy to see that in this case the Majority Is Stablest theorem implies that the weight of a boolean function on the first level of its Fourier transform is essentially bounded by $2/\pi$. We give an easy and direct proof of this fact.

THEOREM 5. *Suppose* $f : \{-1,1\}^n \to [-1,1]$ *satisfies* $\operatorname{Inf}_i(f) \le \delta$ *for all* i. *Then*

$$\sum_{|S|=1} \hat{f}(S)^2 \le \tfrac{2}{\pi} + C\delta\ ,$$

where $C = 2(1 - \sqrt{2/\pi})$.

We also can give a direct proof of an improved version of Theorem 5 which depends on the mean of f; as the mean becomes small enough, this result approaches a result of Talagrand [54] (which states that for every function $f : \{-1,1\}^n \to \{-1,1\}$ with $\Pr[f=1] = p \le 1/2$ it holds that $\sum_{|S|=1} \hat{f}(S)^2 \le O(p^2\log(1/p))$).

THEOREM 6. *Let* ϕ *be the Gaussian density function and* Φ *be the Gaussian distribution function. Let* $U(x) = \phi(\Phi^{-1}(x)) : [0,1] \to [0, 1/\sqrt{2\pi}]$ *denote the so-called "Gaussian isoperimetric function."*

Suppose $f : \{-1,1\}^n \to [-1,1]$ *satisfies* $\operatorname{Inf}_i(f) \le \delta$ *for all* i. *Letting* $\mu = \frac{1}{2} + \frac{1}{2}\mathbf{E}[f]$, *we have*

$$\sum_{|S|=1} \hat{f}(S)^2 \le 4\left(U(\mu) + \epsilon\right)^2,$$

where the error term ϵ *is given by*

$$\epsilon = \max\{1, \sqrt{|\Phi^{-1}(\mu)|}\} \cdot O(\sqrt{\delta}).$$

This theorem is sharp up to the error term, as can be observed by considering restrictions symmetric threshold functions with various thresholds (see, e.g., [45] or [44] for explicit computations). Note that for x small, $U(x) \sim x\sqrt{2\ln(1/x)}$; this is why our result is comparable with Talagrand's.

6.3. Larger domains: q-ary functions. In this section we state our results for q-ary functions and for q-ary constraint satisfaction problems. We will be concerned with two such 2-CSPs. The first is MAX-q-CUT, the problem of partitioning a graph into q parts so as to maximize the number of edges between parts. The second is MAX-2LIN(q): Given an integer $q \geq 2$, the MAX-2LIN(q) problem is to maximize the number of satisfied equations in a given system of linear equations modulo q, where exactly two variables appear in each equation. See section 11.1 for formal definitions.

Stability estimates. Our hardness results are based in part on the following analysis of the noise stability of q-ary functions, as discussed in section 4. We first obtain an exact analytic expression for the noise stability of the plurality function.

THEOREM 7. *Fix q and $-\frac{1}{q-1} \leq \rho \leq 1$. Then*

$$\lim_{n\to\infty} \mathbb{S}_\rho(\text{Plurality}_n) = qI(q,\rho),$$

where $I(q,\rho)$ is defined as follows: Let $(U_1, V_1), \ldots, (U_q, V_q)$ be a set of q independent and identically distributed (i.i.d.) normal vectors with mean 0 and covariance matrix $\left(\begin{smallmatrix} 1 & \rho \\ \rho & 1 \end{smallmatrix}\right)$; then

$$I(q,\rho) = \Pr\left[U_1 = \max_{1\leq i\leq q} U_i, \quad V_1 = \max_{1\leq i\leq q} V_i\right].$$

Further, the quantity $I(q,\rho)$ is precisely equal the key quantity called $I(\rho)$ (with $q = k$) in Frieze and Jerrum's paper on MAX-q-CUT [26] *(see also* [13]*).*

As a corollary of Theorem 7, and a result of de Klerk, Pasechnik, and Warners [13] (see also [26]) which gives the asymptotics of $I(q,\rho)$, we obtain the following.

COROLLARY 2. *For every fixed $0 \leq \rho < 1$, we have*

$$\text{PlurStab}(q,\rho) \sim \left(\frac{1}{q-1}\right)^{(1-\rho)/(1+\rho)} (4\pi \ln(q-1))^{-\rho/(1+\rho)} \frac{\Gamma(1/(1+\rho))^2}{(1-\rho^2)^{1/2}}, \tag{3}$$

where the $\sim$ indicates that the ratio of the two sides is 1 as $q \to \infty$, and Γ is the gamma function.

Since we do not have the Plurality Is Stablest Conjecture, we cannot actually use Corollary 2 in our hardness results. Instead we use the MOO theorem, which is stated in terms of the function $\Lambda_\rho(\mu)$ (recall Definition 8); therefore we need bounds on its asymptotics. Slightly improving the estimate from Lemma 11.1 of [13], we have the following.

PROPOSITION 1. *Denote by ϕ the Gaussian density function $\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$, and let $N(x) = \int_x^\infty \phi$ denote the Gaussian tail probability function. For any $0 \leq \mu < 1/2$, let $t > 0$ be the number such that $N(t) = \mu$. Then for all $0 \leq \rho \leq 1$,*

$$\Lambda_\rho(\mu) \leq (1+\rho) \cdot \frac{\phi(t)}{t} \cdot N\left(t\sqrt{\tfrac{1-\rho}{1+\rho}}\right). \tag{4}$$

Note that in the case $\mu = 1/2$, $\Lambda_\rho(1/2) = \frac{1}{2}(1 - \frac{2}{\pi} \arccos \rho)$, and the case $\mu > 1/2$ can be easily reduced to the case $\mu < 1/2$. Also, it is relatively easy to see that the right-hand side of (4) becomes a lower bound on $\Lambda_\rho(\mu)$ if the $(1+\rho)$ factor is removed.

In fact, we are mainly interested in the case where $\mu \to 0$ ($t \to \infty$). In this case, it turns out that (4) holds as an equality up to a $1 + o_\mu(1)$ factor (even if ρ is a function of μ). This yields the following.

COROLLARY 3. *Let $\mu \to 0$ and let $t = t(\mu)$ be defined as in Proposition 1. Then the following hold:*

1. *For any* $\rho = \rho(\mu)$, $0 < \rho < 1$,

$$\Lambda_\rho(\mu) \quad \overset{\mu \to 0}{\sim} \quad (1+\rho) \cdot \mu \cdot N\Big(t\sqrt{\tfrac{1-\rho}{1+\rho}}\Big)$$

(where by "$\sim$" we mean that the ratio between the two sides tends to one).

2. *If* $0 < \rho < 1$ *is fixed, then*

$$\Lambda_\rho(\mu) \quad \overset{\mu \to 0}{\sim} \quad \mu^{2/(1+\rho)} \left(4\pi \ln\left(\frac{1}{\mu}\right)\right)^{-\rho/(1+\rho)} \frac{(1+\rho)^{3/2}}{(1-\rho)^{1/2}}.$$

3. *For any fixed* $0 < \eta < 1$,

$$q\Lambda_{1-\eta}(1/q) \le (1/q)^{\eta/(2-\eta)} .$$

4. *For any* $\lambda = \lambda(q) \in (0,1)$, *let* $\rho = 1 - \frac{\lambda}{\ln q}$. *Then*

$$q\Lambda_\rho(1/q) \le 1 - \sqrt{2/\pi} \cdot \sqrt{\lambda} + o_{\lambda \to 0}(1) + o_{q \to \infty}(1).$$

Part 2 of Corollary 3 is due de Klerk, Pasechnik, and Warners [13]. It implies that $q\Lambda_\rho(1/q)$ and $\mathrm{PlurStab}_\rho(q)$ have the same asymptotics as q tends to infinity, up to a small multiplicative constant. The other statements of Corollary 3 are proven in section 11.3.

Hardness results. We now move to stating our hardness results for q-ary domains. For MAX-q-CUT we show that assuming the Unique Games Conjecture, it is impossible to essentially improve on the approximation ratios for MAX-q-CUT achieved by Frieze and Jerrum [26] by more than an additive ϵ.

THEOREM 8. *Assume the Unique Games Conjecture. Then for every* $\epsilon > 0$ *it is NP-hard to distinguish* $(1-\epsilon)$*-satisfiable instances of MAX-q-CUT from instances that are at most* $(1 - 1/q + (2\ln q)/q^2 + O(\ln\ln q)/q^2)$*-satisfiable.*

Our hardness result for MAX-2LIN(q) is formulated in terms of $\Lambda_q(\mu)$, discussed above.

THEOREM 9. *Assume the Unique Games Conjecture. Then for every* $q \ge 2$, $\rho \in [0,1]$, *and* $\epsilon > 0$, *given an instance of MAX-2LIN(q), it is NP-hard to distinguish between the case where it is at least* $(\rho + \frac{1}{q}(1-\rho) - \epsilon)$*-satisfiable and the case where it is at most* $(q\Lambda_\rho(\frac{1}{q}) + \epsilon)$*-satisfiable. Furthermore, this holds even for instances in which all equations are of the form* $x_i - x_j = c$.

Using the asymptotics of $\Lambda_\rho(\mu)$ given above in Corollary 3, we have the following.

COROLLARY 4. *Assume the Unique Games Conjecture. Then for every fixed* $\eta > 0$ *there exists* $q_0 = q_0(\eta)$ *such that for every fixed* $q > q_0$ *the following holds. Given an instance of MAX-2LIN(q), it is NP-hard to distinguish between the case where the instance is at least* $(1-\eta)$*-satisfiable and the case where it is at most* $(1/q)^{\eta/(2-\eta)}$*-satisfiable.*

COROLLARY 5. *Assume the Unique Games Conjecture, and let* $\lambda = \lambda(q) \in (0,1)$. *Given an instance of MAX-2LIN(q), it is NP-hard to distinguish between the case where the instance is at least* $(1 - \frac{\lambda}{\ln q})$*-satisfiable and the case where it is at most s-satisfiable, where*

$$s = 1 - \sqrt{2/\pi} \cdot \sqrt{\lambda} + o_{\lambda \to 0}(1) + o_{q \to \infty}(1).$$

Note that MAX-2LIN(q) is itself essentially an instance of Unique Label Cover, except for the fact that the variable/equation structure need not be bipartite. But in fact, it is easy to observe that the "nonbipartite" version of the Unique Games Conjecture is equivalent to the usual Unique Games Conjecture [41] (up to a factor of 2 in the soundness). Hence Theorem 9 and its corollaries may be viewed as concerning the allowable parameter tradeoffs in the Unique Games Conjecture. In particular, Corollary 4 implies the following.

Corollary 6. *The Unique Games Conjecture holds iff it holds as follows: For every $\eta > 0$ and label set size q (sufficiently large as a function of η), it is NP-hard to distinguish whether the Unique Label Cover problem with label set size q has optimum at least $1-\eta$ or at most $(1/q)^{\eta/(2-\eta)}$.*

(The factor of 2 lost in soundness from passing to a bipartite version can be absorbed since the soundness obtained in the proof of Corollary 4 is actually stronger by a factor of $(\log q)^{\Omega(1)}$.)

Recently, a result of Charikar, Makarychev, and Makarychev [11] showed that the parameters in Corollary 6 are almost optimal. They give an algorithm for Unique Label Cover with label set size q that, given an instance with optimum $(1-\eta)$, outputs an assignment which satisfies at least a $(1/q)^{\eta/(2-3\eta)}$-fraction of the constraints.

7. Definitions and technical preliminaries. In this section we give some definitions and make some technical observations concerning the Majority Is Stablest theorem, reducing it to a form which is useful for our MAX-CUT reduction.

7.1. MAX-CUT and MAX-2SAT. For the majority of this paper, we will be concerned with the MAX-CUT problem; we will also later consider the MAX-2SAT problem. We give the formal definitions of these problems below.

Definition 9 (MAX-CUT). *Given an undirected graph $G = (V, E)$, the MAX-CUT problem is that of finding a partition $C = (V_1, V_2)$ which maximizes the size of the set $(V_1 \times V_2) \cap E$. Given a weight function $w : E \to \mathbb{R}^+$, the weighted MAX-CUT problem is that of maximizing*

$$\sum_{e \in (V_1 \times V_2) \cap E} w(e).$$

Definition 10 (MAX-2SAT). *An instance of the MAX-2SAT problem is a set of boolean variables and a set of disjunctions over (exactly) two literals each, where a literal is either a variable or its negation. The problem is to assign the variables so that the number of satisfied literals is maximized. Given a nonnegative weight function over the set of disjunctions, the weighted MAX-2SAT problem is that of maximizing the sum of weights of satisfied disjunctions.*

As we noted earlier, [12] implies that the achievable approximation ratios for the weighted versions of the above two problems are the same, up to an additive $o(1)$, as the approximation ratios of the respective nonweighted versions. Hence in this paper we freely work with the weighted version.

7.2. Analytic notions. In this paper we treat the bit TRUE as -1 and the bit FALSE as 1; we consider functions $f : \{-1,1\}^n \to \mathbb{R}$ and say a function is *boolean-valued* if its range is $\{-1, 1\}$. The domain $\{-1,1\}^n$ is viewed as a probability space under the uniform measure and the set of all functions $f : \{-1,1\}^n \to \mathbb{R}$ as an inner product space under $\langle f, g\rangle = \mathbf{E}[fg]$. The associated norm in this space is given by $\|f\|_2 = \sqrt{\mathbf{E}[f^2]}$.

Fourier expansion. For $S \subseteq [n]$, let χ_S denote the parity function on S, $\chi_S(x) = \prod_{i \in S} x_i$. It is well known that the set of all such functions forms an orthonormal basis for our inner product space, and thus every function $f : \{-1,1\}^n \to \mathbb{R}$ can be expressed as

$$f = \sum_{S\subseteq[n]} \hat{f}(S)\chi_S.$$

Here the real quantities $\hat{f}(S) = \langle f, \chi_S\rangle$ are called the *Fourier coefficients* of f and the above is called the *Fourier expansion* of f. *Plancherel's identity* states that $\langle f, g\rangle = \sum_S \hat{f}(S)\hat{g}(S)$ and in particular, $\|f\|_2^2 = \sum_S \hat{f}(S)^2$. Thus if f is boolean-valued, then $\sum_S \hat{f}(S)^2 = 1$, and if $f : \{-1,1\}^n \to [-1,1]$, then $\sum_S \hat{f}(S)^2 \le 1$. We speak of f's squared Fourier coefficients as *weights*, and we speak of the sets S being stratified into *levels* according to $|S|$. So for example, by the *weight of f at level* 1 we mean $\sum_{|S|=1} \hat{f}(S)^2$.

The Bonami–Beckner operator. For any $\rho \in [-1,1]$ we define the *Bonami–Beckner operator* T_ρ, a linear operator on the space of functions $\{-1,1\}^n \to \mathbb{R}$, by $T_\rho(f)(x) = \mathbf{E}[f(y)]$, where each coordinate y_i of y is independently chosen to be x_i with probability $\frac{1}{2} + \frac{1}{2}\rho$ and $-x_i$ with probability $\frac{1}{2} - \frac{1}{2}\rho$. It is easy to check that $T_\rho(f) = \sum_S \rho^{|S|}\hat{f}(S)\chi_S$. It is also easy to verify the following relation between T_ρ and the noise stability (see Definition 3).

Proposition 2. *Let $f\colon \{-1,1\}^n \to \mathbb{R}$ and $\rho \in [-1,1]$. Then*

$$\mathbb{S}_\rho(f) = \langle f, T_\rho f\rangle = \sum_{S\subseteq[n]} \rho^{|S|}\hat{f}(S)^2.$$

The following identity is a well-known one, giving a Fourier analytic formula for the influences of a coordinate on a function (see Definition 2).

Proposition 3. *Let $f\colon \{-1,1\}^n \to \mathbb{R}$. Then for every $i \in [n]$,*

$$\mathrm{Inf}_i(f) = \sum_{S\ni i} \hat{f}(S)^2. \tag{5}$$

Once we have the Fourier analytic formula for the influence, we can consider the contribution to the influence of characters of bounded size.

Definition 11. *Let $f\colon \{-1,1\}^n \to \mathbb{R}$ and let $i \in [n]$. The k*-degree influence of coordinate i on f *is defined by*

$$\mathrm{Inf}_i^{\le k}(f) = \sum_{\substack{S\ni i\\ |S|\le k}} \hat{f}(S)^2.$$

7.3. Different forms of the Majority Is Stablest theorem. Recall the Majority Is Stablest theorem (proved in [47]).

Majority Is Stablest theorem: *Fix $\rho \in [0,1)$. Then for any $\epsilon > 0$ there is a small enough $\delta = \delta(\epsilon,\rho) > 0$ such that if $f : \{-1,1\}^n \to [-1,1]$ is any function satisfying*

$$\mathbf{E}[f] = 0, \ \textit{and}$$

$$\mathrm{Inf}_i(f) \le \delta \quad \textit{for all } i = 1\ldots n,$$

then

$$\mathbb{S}_\rho(f) \le 1 - \tfrac{2}{\pi}\arccos\rho + \epsilon.$$

In the MAX-CUT reduction we need a slightly altered version of the Majority Is Stablest theorem. First, we can replace influences by low-degree influences.

PROPOSITION 4. *The Majority Is Stablest theorem remains true if the assumption that* $\mathrm{Inf}_i(f) \leq \delta$ *for all* i *is replaced by the assumption that* $\mathrm{Inf}_i^{\leq k'}(f) \leq \delta'$, *where* δ' *and* k' *are universal functions of* ϵ *and* ρ.

Proof. Fix $\rho < 1$ and $\epsilon > 0$. Choose γ such that $\rho^k(1-(1-\gamma)^{2k}) < \epsilon/4$ for all k. Let δ be chosen such that if $\mathrm{Inf}_i(g) \leq \delta$ for all i, then $\mathbb{S}_\rho(g) \leq 1 - \frac{2}{\pi}\arccos\rho + \epsilon/4$. Choose $\delta' = \delta/2$ and k' such that $(1-\gamma)^{2k'} < \delta'$.

Let f be a function satisfying $\mathrm{Inf}_i^{\leq k'}(f) \leq \delta'$ and let $g = T_{1-\gamma}f$. Note that

$$\mathrm{Inf}_i(g) \leq \sum_{S: i\in S, |S|\leq k'} \hat{f}(S)^2 + (1-\gamma)^{2k'} \sum_{S: i \in S, |S| \leq k'} \hat{f}(S)^2 < \delta' + \delta' = \delta$$

for all i.

It now follows that $\mathbb{S}_\rho(g) \leq 1 - \frac{2}{\pi}\arccos\rho + \epsilon/4$ and therefore

$$\mathbb{S}_\rho(f) = \mathbb{S}_\rho(g) + \sum_S (\rho^{|S|}(1-(1-\gamma)^{|S|}))\hat{f}(S)^2 < 1 - \tfrac{2}{\pi}\arccos\rho + 3\epsilon/4. \qquad \square$$

Second, we need to treat the case of negative ρ.

PROPOSITION 5. *The Majority Is Stablest theorem is true "in reverse" for* $\rho \in (-1, 0]$. *That is,* $\mathbb{S}_\rho(f) \geq 1 - \frac{2}{\pi}\arccos\rho - \epsilon$, *and furthermore, the assumption* $\mathbf{E}[f] = 0$ *becomes unnecessary.*

Proof. Let $f : \{-1,1\}^n \to [-1,1]$ satisfy $\mathrm{Inf}_i(f) \leq \delta$ for all i. Let g be the odd part of f, $g(x) = (f(x) - f(-x))/2 = \sum_{|S| \text{ odd}} \hat{f}(S)x_S$. Then $\mathbf{E}[g] = 0$, $\mathrm{Inf}_i(g) \leq \mathrm{Inf}_i(f)$ for all i, and $\mathbb{S}_\rho(f) \geq \mathbb{S}_\rho(g) = -\mathbb{S}_{-\rho}(g)$, which exceeds $-(1 - \frac{2}{\pi}\arccos\rho + \epsilon)$ by the Majority Is Stablest theorem applied to g. $\square$

Combining the above two propositions, we get the result that will be used in our reduction from Unique Label Cover to 2-bit CSPs.

PROPOSITION 6. *Fix* $\rho \in (-1, 0]$. *Then for any* $\epsilon > 0$ *there is a small enough* $\delta = \delta(\epsilon, \rho) > 0$ *and a large enough* $k = k(\epsilon, \rho)$ *such that if* $f : \{-1,1\}^n \to [-1,1]$ *is any function satisfying*

$$\mathrm{Inf}_i^{\leq k}(f) \leq \delta \quad \textit{for all } i = 1 \dots n,$$

then

$$\mathbb{S}_\rho(f) \geq 1 - \tfrac{2}{\pi}\arccos\rho - \epsilon.$$

8. Reduction from Unique Label Cover to MAX-CUT. In this section we prove Theorem 1.

8.1. The PCP. We construct a PCP that reads two bits from the proof and accepts iff the two bits are unequal. The completeness and soundness are c and s, respectively. This implies that MAX-CUT is NP-hard to approximate within any factor greater than s/c. The reduction from the PCP to MAX-CUT is straightforward and can be considered standard: Let the bits in the proof be vertices of a graph and the tests of the verifier be the edges of the graph. The $\{-1, 1\}$ assignment to bits in the proof corresponds to a partition of the graph into two parts, and the tests for which the verifier accepts correspond to the edges cut by this partition.

The completeness and soundness properties of the PCP rely on the Unique Games Conjecture and the Majority Is Stablest theorem. The Unique Label Cover instance given by the Unique Games Conjecture serves as the PCP outer verifier. The soundness of the Long Code-based inner verifier is implied by the Majority Is Stablest theorem.

Before we explain the PCP test, we need some notation. For $x \in \{-1,1\}^M$ and a bijection $\sigma : [M] \to [M]$, let $x \circ \sigma$ denote the string $(x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(M)})$. For x, $\mu \in \{-1,1\}^M$, let $x\mu$ denote the M-bit string that is the coordinatewise product of x and μ.

The PCP verifier is given the Unique Label Cover instance $\mathcal{L}(V, W, E, [M], \{\sigma_{v,w}\}_{(v,w)\in E})$ given by the Unique Games Conjecture. Using a result from [40], we may assume the bipartite graph is regular on the V side, so that choosing a uniformly random vertex $v \in V$ and a random neighbor w of v yields a uniformly random edge (u, w). We assume that the Unique Label Cover instance is either $(1-\eta)$-satisfiable or at most γ-satisfiable, where we will choose the values of η and γ to be sufficiently small later. The verifier expects as a proof the Long Code of the label of every vertex $w \in W$. The verifier is parameterized by $\rho \in (-1, 0)$.

The PCP verifier for MAX-CUT with parameter $-1 < \rho < 0$.

- Pick a vertex $v \in V$ at random and two of its neighbors $w, w' \in W$ at random. Let $\sigma = \sigma_{v,w}$ and $\sigma' = \sigma_{v,w'}$ be the respective bijections for edges (v, w) and (v, w').
- Let f_w and $f_{w'}$ be the supposed Long Codes of the labels of w and w', respectively.
- Pick $x \in \{-1,1\}^M$ at random.
- Pick $\mu \in \{-1,1\}^M$ by choosing each coordinate independently to be 1 with probability $\frac{1}{2} + \frac{1}{2}\rho < \frac{1}{2}$ and -1 with probability $\frac{1}{2} - \frac{1}{2}\rho > \frac{1}{2}$.
- Accept iff

$$f_w(x \circ \sigma) \neq f_{w'}((x \circ \sigma')\mu).$$

8.2. Completeness. It is easy to see that the completeness of the verifier is at least $(1-2\eta)(\frac{1}{2} - \frac{1}{2}\rho)$. Assume that the Label Cover instance has a labeling that satisfies a $1-\eta$ fraction of edges. Take this labeling and encode the labels via Long Codes. We will show that the verifier accepts with probability at least $(1-2\eta)(\frac{1}{2} - \frac{1}{2}\rho)$.

With probability at least $1-2\eta$, both the edges (v, w) and (v, w') are satisfied by the labeling. Let the labels of v, w, w' be $i, j, j' \in [M]$, respectively, so that by the acceptance condition $\sigma(j) = i = \sigma'(j')$. The functions $f_w, f_{w'}$ are the Long Codes of j, j', respectively. Hence

$$f_w(x \circ \sigma) = x_{\sigma(j)} = x_i, \quad f_{w'}((x \circ \sigma')\mu) = x_{\sigma'(j')}\mu_{j'} = x_i \mu_{j'}.$$

Thus the two bits are unequal (and the test accepts) iff $\mu_{j'} = -1$, which happens with probability $\frac{1}{2} - \frac{1}{2}\rho$.

8.3. Soundness. We prove soundness in the contrapositive direction, as is usual in PCP proofs: Assume that some supposed Long Codes f_w cause the PCP verifier to accept with probability at least $(\arccos\rho)/\pi + \epsilon$. We use Fourier methods to "list-decode" the Long Codes and extract a labeling for the Unique Label Cover instance that satisfies some $\gamma' = \gamma'(\epsilon, \rho)$ fraction of its edges. Since this constant does not depend on the Label Cover label set size M, we can take M large enough in the Unique Games Conjecture to get soundness $\gamma < \gamma'$, as required.

We first analyze the probability of acceptance for the PCP verifier by arithmetizing it as follows:

$$
\begin{aligned}
\Pr[\text{acc}] &= \mathop{\mathbf{E}}_{v,w,w',x,\mu}\left[\frac{1}{2}-\frac{1}{2}f_w(x\circ\sigma)f_{w'}((x\mu)\circ\sigma')\right] && ((x\mu)\circ\sigma' \text{ has the same distrib. as } (x\circ\sigma')\mu)\\
&= \frac{1}{2}-\frac{1}{2}\cdot\mathop{\mathbf{E}}_{v,x,\mu}\left[\mathop{\mathbf{E}}_{w,w'}[f_w(x\circ\sigma)f_{w'}((x\mu)\circ\sigma')]\right]\\
&= \frac{1}{2}-\frac{1}{2}\cdot\mathop{\mathbf{E}}_{v,x,\mu}\left[\mathop{\mathbf{E}}_{w}[f_w(x\circ\sigma)]\cdot\mathop{\mathbf{E}}_{w'}[f_{w'}((x\mu)\circ\sigma')]\right] && (\text{using independence of } w \text{ and } w')\\
&= \frac{1}{2}-\frac{1}{2}\cdot\mathop{\mathbf{E}}_{v,x,\mu}[g_v(x)g_v(x\mu)] && \left(\text{where we define } g_v(z)=\mathop{\mathbf{E}}_{w\sim v}[f_w(z\circ\sigma_{v,w})]\right)\\
&= \frac{1}{2}-\frac{1}{2}\cdot\mathop{\mathbf{E}}_{v}[\mathbb{S}_\rho(g_v)]. && (6)
\end{aligned}
$$

(The reader may think of g_v as "polling" v's neighbors w on its labeling.) Now if $\Pr[\text{acc}]\geq(\arccos\rho)/\pi+\epsilon$, then for at least an $\epsilon/2$ fraction of $v\in V$,

$$\mathbb{S}_\rho(g_v)\leq 1-\tfrac{2}{\pi}\arccos\rho-\epsilon.$$

We say that such a vertex v is "good." For every good v, we apply the Majority Is Stablest theorem in the guise of Proposition 6 to conclude that g_v has at least one coordinate, say j, with k-degree influence at least δ. We shall give the label j to v. In this way, all good $v\in V$ are labeled. For a good v, since $\mathrm{Inf}_j^{\leq k}(g_v)\geq\delta$, we have

$$\delta\leq\sum_{\substack{S\ni j\\|S|\leq k}}\widehat{g_v}(S)^2=\sum_{\substack{S\ni j\\|S|\leq k}}\mathop{\mathbf{E}}_{w}[\widehat{f_w}(\sigma^{-1}(S))]^2\leq\sum_{\substack{S\ni j\\|S|\leq k}}\mathop{\mathbf{E}}_{w}[\widehat{f_w}(\sigma^{-1}(S))^2]=\mathop{\mathbf{E}}_{w}\left[\mathrm{Inf}_{\sigma^{-1}(j)}^{\leq k}(f_w)\right]. \tag{7}$$

For every $w\in W$, define the set of candidate labels for w to be

$$\mathrm{Cand}[w]=\{i\in[M]\colon\mathrm{Inf}_i^{\leq k}(f_w)\geq\delta/2\}.$$

Since $\sum_i\mathrm{Inf}_i^{\leq k}(f_w)\leq k$, we conclude that $|\mathrm{Cand}[w]|\leq 2k/\delta$. Inequality (7) implies that for every good v, at least a $\delta/2$ fraction of neighbors w of v have $\mathrm{Inf}_{\sigma_{v,w}^{-1}(j)}^{\leq k}(f)\geq\delta/2$, and therefore $\sigma^{-1}(j)\in\mathrm{Cand}[w]$. Now we label each vertex $w\in W$ by choosing a random element of $\mathrm{Cand}[w]$ (or any label if this set is empty). It follows that among the set of edges adjacent to good vertices v, at least a $(\delta/2)(\delta/2k)$-fraction is satisfied in expectation. Thus it follows that there is labeling for all vertices which satisfies a $\gamma'=(\epsilon/2)(\delta/2)(\delta/2k)$ fraction of all edges. This completes the proof of soundness.

8.4. Completion of the proof of Theorem 1. We have just shown how to reduce Unique Label Cover instances to MAX-CUT instances with completeness $(\frac{1}{2}-\frac{1}{2}\rho)(1-2\eta)$ and soundness $(\arccos\rho)/\pi+\epsilon$, where η and ϵ can be made arbitrarily small. The main statement of Theorem 1 follows by slightly modifying ρ to move the completeness correction $(1-2\eta)$ into the soundness correction ϵ. This result implies a hardness of approximation factor of

$$\frac{\arccos(\rho)/\pi}{\frac{1}{2}-\frac{1}{2}\rho}+\epsilon$$

for any constant $-1 < \rho < 0$ and $\epsilon > 0$; choosing $\rho = \rho^*$ as stated in the theorem yields the desired hardness factor $\alpha_{\text{GW}} + \epsilon$.

9. Other 2-bit CSPs. The same method used to prove hardness of approximation for MAX-CUT can be used to give improved hardness of approximation for another important 2-bit CSP, namely MAX-2SAT. Recall that the input to a MAX-2SAT problem is a collection of clauses, i.e., disjunctions, of at most two variables; the goal is to find an assignment that satisfies as many clauses as possible.

The natural inner verifier test for MAX-2SAT is this: with probability 1/2 test $f_w(x \circ \sigma) \vee f_{w'}((x \circ \sigma')\mu)$; with probability 1/2 test $-f_w(x \circ \sigma) \vee -f_{w'}((x \circ \sigma')\mu)$. It is easy to check that this leads to an acceptance probability of $\frac{3}{4} - \frac{1}{4}\mathbb{S}_\rho(g_v)$ in place of (6). The dictator passes this test with probability $\frac{3}{4} - \frac{1}{4}\rho$; the Majority Is Stablest theorem implies that no function with small low-degree influences can pass this test with probability exceeding $\frac{3}{4} - \frac{1}{4}(1 - \frac{2}{\pi}\arccos\rho) + \epsilon$. This leads to a hardness of approximation ration of

$$\beta = \min_{-1<\rho<0} \frac{\frac{3}{4} - \frac{1}{4}(1 - \frac{2}{\pi}\arccos\rho)}{\frac{3}{4} - \frac{1}{4}\rho} \approx .943943. \tag{8}$$

This is our Theorem 2.

Note that β is smaller than the best unconditional hardness factor known for MAX-2SAT, $21/22 \approx .954545$, due to Håstad [33] (using the gadget of Bellare, Goldreich, and Sudan [3]); as well, the best algorithm known for MAX-2SAT, due to Lewin, Livnat, and Zwick [43], achieves an approximation ratio of .9401 which is close to and smaller than β.

Our methodology does not seem to improve the hardness factors for other 2-bit CSPs beyond α_{GW}. Consider the MAX-2ConjSAT problem, in which the input is a collection of *conjunctions* of (at most) two variables and the goal is to satisfy as many conjunctions as possible. The natural inner verifier test is this: with probability 1/2 test $f_w(x \circ \sigma) \wedge f_{w'}((x \circ \sigma')\mu)$; with probability 1/2 test $-f_w(x \circ \sigma) \wedge -f_{w'}((x \circ \sigma')\mu)$. This leads to an acceptance probability of $\frac{1}{4} - \frac{1}{4}\mathbb{S}_\rho(g_v)$. By the Majority Is Stablest theorem, we get the same hardness of approximation for MAX-2ConjSAT as we do for MAX-CUT, α_{GW}, since $(\frac{1}{4} - \frac{1}{4}(1 - \frac{2}{\pi}\arccos\rho))/(\frac{1}{4} - \frac{1}{4}\rho) = ((\arccos\rho)/\pi)/(\frac{1}{2} - \frac{1}{2}\rho)$. In some sense this may not be surprising since the best algorithm known for this problem ([43] again) already achieves an approximation ratio of .8740, which is nearly α_{GW}. In fact, the same paper achieves .8740 even for the most general problem, MAX-2CSP, in which arbitrary 2-bit constraints are allowed.

Motivated by these results, we are led to conjecture that MAX-2SAT *is* polynomial-time approximable to within any factor less than β and that MAX-2CSP, MAX-DICUT, MAX-2ConjSAT, etc. are all polynomial-time approximable to within any factor less than α_{GW}. We will now show that these bounds *are* achievable for a slight weakening of the problems.

DEFINITION 12. *Given a* 2*-bit CSP, by its* balanced *version we mean the problem with the restriction that every input instance* $\{C_1, \ldots, C_m\}$ *has the following property: for each* $i = 1 \ldots n$*, the expected number of constraints satisfied when* x_i *is set to* 1 *and the other variables are set uniformly at random is equal to the expected number of constraints satisfied when* x_i *is set to* -1 *and the other variables are set uniformly at random.*

As an example, Balanced-MAX-2SAT is the MAX-2SAT problem with the additional constraint that each variable appears positively and negatively in equally many clauses (in the weighted case, with equal total weight).

We contend that the balanced versions of 2-bit CSPs ought to be equally hard as their general versions; the intuition is that if more constraints are expected to be satisfied if x_i is set to, say, 1 rather than -1, it is a "free hint" that the x_i should be set to TRUE. Note that the reductions we suggest from Unique Label Cover to MAX-2SAT, MAX-2ConjSAT, etc. produce balanced instances, and thus we get the same hardness of approximation bounds, β and α_{GW}, for the balanced problems (conditional on the two conjectures).

We can prove unconditionally that Balanced-MAX-2SAT is polynomial-time approximable to within any factor less than β, and that MAX-2CSP, MAX-DICUT, MAX-2ConjSAT, MAX-2LIN, etc. are all polynomial-time approximable to within any factor less than α_{GW}. By way of illustration, we prove Theorem 3.

Proof. The algorithm is essentially the same as that used by Goemans and Williamson. The input is a collection of clauses C of the form $(y \vee z)$, where $y = r_i x_i$ and $z = r_j x_j$ for some variables x_i and x_j and signs r_i and r_j. Arithmetizing each clause with $-1 \vee -1 = 1$, $-1 \vee 1 = 1$, $1 \vee -1 = 1$, $1 \vee 1 = 0$, we get $\frac{3}{4} - \frac{1}{4}y - \frac{1}{4}z - \frac{1}{4}y \cdot z$. Thus we have the objective function

$$\text{OBJ} = \sum_{C=(y\vee z)} \tfrac{3}{4} - \tfrac{1}{4}y - \tfrac{1}{4}z - \tfrac{1}{4}y \cdot z.$$

The condition that the instance is balanced is precisely equivalent to the condition that the linear terms cancel out. (This holds true by definition for all balanced 2-bit CSP problems.) Thus in fact

$$\text{OBJ} = \sum_{C=(y\vee z)} \tfrac{3}{4} - \tfrac{1}{4}y \cdot z.$$

Hence the optimum value of the Balanced-MAX-2SAT instance is

$$\text{OPT} = \max\ \text{OBJ} \qquad \text{subject to } x_i \in \{-1, 1\} \text{ for all } i.$$

Following Goemans and Williamson we directly relax this to a semidefinite program by replacing x_i with a high-dimensional vector v_i, subject to $v_i \cdot v_i = 1$, and solving; in polynomial time we can find a solution $\{v_i\}$ which achieves SDP $- \epsilon$, where SDP denotes the optimal value of the semidefinite program. We now round by picking r to be a random Gaussian vector and setting $x_i = \text{sgn}(r \cdot v_i)$. Recalling from [27] that this gives $\mathbf{E}[x_i \cdot x_j] = 1 - \frac{2}{\pi} \arccos(v_i \cdot v_j)$, we have for any clause $(y \vee z) = (r_i x_i \vee r_j x_j)$,

$$\mathbf{E}[\tfrac{3}{4} - \tfrac{1}{4}(r_i x_i) \cdot (r_j x_j)] = \tfrac{3}{4} - \tfrac{1}{4}(1 - \tfrac{2}{\pi} \arccos(r_i v_i \cdot r_j v_j)) \geq \beta(\tfrac{3}{4} - \tfrac{1}{4}(r_i v_i \cdot r_j v_j)),$$

where we have used the definition of β and the fact that it is unchanged if we let ρ range over $[-1, 1]$. It follows that $\mathbf{E}[\text{OBJ}] \geq \beta\text{SDP} \geq \beta\text{OPT}$, and the proof is complete. □

10. Special cases of the Majority Is Stablest theorem. In this section we describe special cases of the Majority Is Stablest theorem which are of particular interest and have elementary proofs.

We will need the following versions of the central limit theorem with error bounds: the first is a multidimensional version from [6, Corollary 16.3]; the second is the (nonuniform) version of the Berry–Esseen theorem [21].

THEOREM 10. *Let $\mathbf{X}_1, \ldots, \mathbf{X}_n$ be independent random variables taking values in $\mathbb{R}^k$ satisfying*

- $\mathbf{E}[\mathbf{X}_j] = 0, j = 1 \dots n$;
- $n^{-1} \sum_{j=1}^{n} \mathrm{Cov}(\mathbf{X}_j) = V$, *where* Cov *denotes the variance-covariance matrix;*
- λ *is the smallest eigenvalue of* V, *and* Λ *is the largest eigenvalue of* V;
- $\rho_3 = n^{-1} \sum_{j=1}^{n} \mathbf{E}[|\mathbf{X}_j|^3] < \infty$.

Let Q_n *denote the distribution of* $n^{-1/2}(\mathbf{X}_1 + \cdots + \mathbf{X}_n)$, *let* $\Phi_{0,V}$ *denote the distribution of the* k*-dimensional Gaussian with mean* 0 *and variance-covariance matrix* V, *and let* $\eta = C\lambda^{-3/2}\rho_3 n^{-1/2}$, *where* C *is a certain universal constant.*

Then for any Borel set A,

$$|Q_n(A) - \Phi_{0,V}(A)| \le \eta + B(A),$$

where $B(A)$ *is the following measure of the boundary of*

$$A : B(A) = 2 \sup_{y \in \mathbb{R}^k} \Phi_{0,V}((\partial A)^{\eta'} + y),$$

where $\eta' = \Lambda^{1/2}\eta$ *and* $(\partial A)^{\eta'}$ *denotes the set of points within distance* η' *of the topological boundary of* A.

Theorem 11 (Berry–Esseen). *Let* $X_1, \dots, X_n$ *be a sequence of independent random variables satisfying* $\mathbf{E}[X_j] = 0$ *for all* j, $(\sum_{j=1}^{n} \mathbf{E}[X_j^2])^{1/2} = \sigma$, *and* $\sum_{j=1}^{n} \mathbf{E}[|X_j|^3] = \rho_3$. *Let* $Q = \sigma^{-1}(X_1 + \cdots + X_n)$, *let* F *denote the cumulative distribution function of* Q, $F(x) = \Pr[Q \le x]$, *and let* Φ *denote the cumulative distribution function of a standard normal random variable. Then*

$$\sup_x (1 + |x|^3)|F(x) - \Phi(x)| \le O(\rho_3/\sigma^3).$$

In particular, if A *is any interval in* $\mathbb{R}$, $|\Pr[Q \in A] - \Pr[N(0,1) \in A]| \le O(\rho_3/\sigma^3)$.

10.1. Weighted majorities. In this subsection we prove Theorem 4, which makes the point that the Majority function is not unique as a noise stability maximizer, in the sense that all weighted majority functions with small influences have the same noise stability, i.e., $1 - \frac{2}{\pi} \arccos \rho$.

Theorem 4 follows from the following two propositions.

Proposition 7. *Let* $f : \{-1,1\}^n \to \{-1,1\}$ *be any balanced threshold function,* $f(x) = \mathrm{sgn}(a_1 x_1 + \cdots + a_n x_n)$,[1] *where* $\sum a_i^2 = 1$. *Let* $\delta = \max\{|a_i|\}$. *Then for all* $\rho \in [-1,1]$,

$$\mathbb{S}_\rho(f) = 1 - \tfrac{2}{\pi} \arccos \rho \pm O(\delta(1 - |\rho|)^{-3/2}).$$

Proposition 8. *Let* $f : \{-1,1\}^n \to \{-1,1\}$ *be any balanced threshold function,* $f(x) = \mathrm{sgn}(a_1 x_1 + \cdots + a_n x_n)$, *where* $\sum a_i^2 = 1$. *Let* $\delta = \max\{|a_i|\}$. *Then* $\max_i \{\mathrm{Inf}_i(f)\} \ge \Omega(\delta)$.

We prove the two propositions below.

Proof of Proposition 7. Since f is antisymmetric, we need only to prove the result for $\rho \in [0,1]$. Let x and y be ρ-correlated uniformly random strings, and let $X_j = a_j x_j$, $Y_j = a_j y_j$, and $\mathbf{X}_j = (X_j, Y_j) \in \mathbb{R}^2$. Let Q_n denote the distribution of $\mathbf{X}_1 + \cdots + \mathbf{X}_n = n^{-1/2}(\sqrt{n}\,\mathbf{X}_1 + \cdots + \sqrt{n}\,\mathbf{X}_n)$. Since $\mathbb{S}_\rho(f) = 2\Pr[f(x) = f(y)] - 1$, we are interested in computing $2Q_n(A_{++} \cup A_{--}) - 1$, where A_{++} denotes the positive quadrant of $\mathbb{R}^2$ and A_{--} denotes the opposite quadrant.

[1] Without loss of generality we assume the linear form is never 0.

We shall apply Theorem 10. We have $\mathbf{E}[\mathbf{X}_j] = 0$ for all j. We have $\mathrm{Cov}(\sqrt{n}\,\mathbf{X}_j) = na_i^2 \left[\begin{smallmatrix}1 & \rho\\ \rho & 1\end{smallmatrix}\right]$, and thus $V = n^{-1}\sum \mathrm{Cov}(\sqrt{n}\,\mathbf{X}_j) = \left[\begin{smallmatrix}1 & \rho\\ \rho & 1\end{smallmatrix}\right]$. The eigenvalues of V are $\lambda = 1-\rho$ and $\Lambda = 1+\rho$. Since $|\sqrt{n}\,\mathbf{X}_j|$ is $\sqrt{2n}\,|a_i|$ with probability 1, $\rho_3 = n^{-1}\sum \mathbf{E}[|\sqrt{n}\,\mathbf{X}_j|^3] = 2^{3/2}n^{1/2}\sum |a_i|^3 \le 2^{3/2}n^{1/2}\delta$. Thus $\eta = O(1)\delta(1-\rho)^{-3/2}$ and $\eta' = (1+\rho)^{1/2}\eta = O(\eta)$.

It is well known (see, e.g., [1, equation 26.3.19]) that $\Phi_{0,V}(A_{++}) = \Phi_{0,V}(A_{--}) = 1/2 - (1/2\pi)\arccos(\rho)$, and it is easy to check that $B(A_{++} \cup A_{--}) = O(\eta')$. Thus by Theorem 10 we get $Q_n(A_{++} \cup A_{--}) = 1 - (\arccos\rho)/\pi \pm O(\eta)$, and the theorem follows. □

Proof of Proposition 8. Let C be the constant hidden in the $O(\cdot)$ in the final part of the Berry–Esseen theorem, Theorem 11. For simplicity, we assume that C is a positive integer. We prove Proposition 8 first in the case where

$$1 - 100C^2\delta^2 \ge 1/4, \tag{9}$$

namely, where δ is smaller than some constant.

We may assume without loss of generality that $\delta = a_1 \ge a_2 \ge \cdots \ge a_n \ge 0$. Letting X_i denote the random variable $a_i x_i$, we will prove that $\mathrm{Inf}_1(f) \ge \Omega(\delta)$ by proving that

$$\Pr[|X_2 + \cdots + X_n| \le \delta] \ge \Omega(\delta). \tag{10}$$

Let $m = 100C^2 + 2$. We will split into two cases, depending on the magnitude of a_m. In either case, we shall apply the Berry–Esseen theorem to the sequence $X_m, \dots, X_n$. We have

$$\sigma = \left(\sum_{j=m}^{n} \mathbf{E}[X_j]^2\right)^{1/2} = \left(\sum_{j=m}^{n} a_j^2\right)^{1/2} \ge (1-(m-2)\delta^2)^{1/2} \ge (1-100C^2\delta^2)^{1/2} \ge 1/2,$$

where we have used (9). We also have $\rho_3 = \sum_{j=m}^{n} \mathbf{E}[|X_j|^3] \le \sum_{j=m}^{n} a_m \mathbf{E}[X_j^2] = a_m\sigma^2$, and so the error term in the conclusion of the theorem, $O(\rho_3/\sigma^3)$, is at most $Ca_m/\sigma \le 2Ca_m$.

Case 1. $a_m \le \frac{1}{10C}\delta$. In this case, by the Berry–Esseen theorem we have that

$$\Pr[X_m + \cdots + X_n \in [0,\delta]] \ge \Phi([0,\delta]) - 2Ca_m \ge \delta\phi(\delta) - \delta/5 \ge .04\delta,$$

where we have used the fact that $\phi(\delta) \ge .24$ for $\delta \le 1$. On the other hand, since $a_2, \dots, a_{m-1}$ are all at most δ, it is easy to fix particular signs $y_i \in \{-1, 1\}$ such that $\sum_{i=2}^{m-1} a_i y_i \in [-\delta, 0]$. These signs occur with probability 2^{-m+2}, which is at least 2^{-100C^2}. Thus with probability at least $.04 \cdot 2^{-100C^2}\delta = \Omega(\delta)$ both events occur, and $|X_2 + \cdots + X_n| \le \delta$ as desired.

Case 2. $a_m \ge \frac{1}{10C}\delta$. In this case, we apply the Berry–Esseen theorem to the interval $[-10C\delta, 10C\delta]$ and merely use the fact that $a_m \le \delta$. We conclude that

$$\begin{aligned}\Pr[X_m + \cdots + X_n \in [-10C\delta, 10C\delta]] &\ge \Phi([-10C\delta, 10C\delta]) - 2C\delta\\ &\ge 20C\delta \cdot \phi(10C\delta) - 2C\delta \ge 20C\delta \cdot \tfrac{1}{\sqrt{2\pi}}\left(1 - \tfrac{(10C\delta)^2}{2}\right) - 2C\delta \ge 4C\delta,\end{aligned}$$

where we have used (9) in the last step to infer $1 - (10C\delta)^2/2 \ge 5/8$. Given $X_m + \cdots + X_n = t \in [-10C\delta, 10C\delta]$, it is easy to choose particular signs $y_2, \dots, y_{m-1}$ such that $t + \sum_{i=2}^{m-1} a_i y_i \in [-\delta, \delta]$. This uses the fact that each a_i is at least $\frac{1}{10C}\delta$, and hence $\sum_{i=2}^{m-1} a_i \geq 100C^2 \frac{1}{10C}\delta \geq 10C\delta$; it also uses the fact that each a_i is at most δ. Once again, these signs occur for $x_2, \dots, x_{m-1}$ with probability at least 2^{-100C^2}. Thus $|X_2 + \cdots + X_n| \leq \delta$ happens with probability at least $4C2^{-100C^2}\delta = \Omega(\delta)$, as desired.

Let us now deal with the case where $1 - 100C^2\delta^2 < \frac{1}{4}$, namely, where $\delta > \frac{\sqrt{3}}{20C}$. Let m be the first index for which $|a_m| \leq \frac{c}{C_5}$, where c is a small enough global constant to be chosen later. If such an m does not exist, we set $m = n + 1$.

We prove below that

$$\text{(11)} \qquad \Pr\left[\left|\sum_{i=m}^{n} X_i\right| \leq \delta\right] \geq \Pr\left[\left|\sum_{i=m}^{n} X_i\right| \leq \frac{\sqrt{3}}{20C}\right] > \Omega(1).$$

Since by the choice of m it must be bounded from above by a global constant, (11) implies (10) by arguments similar to those used in Case 1 above and thus completes the proof.

If $\sqrt{|\sum_m^n a_i^2|} \leq \frac{\sqrt{3}}{40C}$, (11) follows immediately from Chernoff's inequality. Otherwise, we use the Berry–Esseen theorem to obtain that

$$\begin{aligned}
\Pr&\left[\left|\sum_{i=m}^{n} X_i\right| \leq \frac{\sqrt{3}}{20C}\right] \\
&\geq \Pr\left[\frac{|\sum_{i=m}^{n} X_i|}{\sqrt{\sum_{i=m}^{n} a_i^2}} \leq \frac{\sqrt{3}}{20C}\right] \\
&\geq \Pr\left[|N(0,1)| \leq \frac{\sqrt{3}}{20C}\right] - C \cdot \left(\sum_{i=m}^{n} |a_i^3|\right) \cdot \left(\frac{40C}{\sqrt{3}}\right)^3 \quad \text{(using Berry–Esseen)} \\
&\geq \Omega\left(\frac{1}{C}\right) - \Omega(C^4) \cdot \frac{c}{C^5} \quad \left(\text{since } \sum_{m}^{n} |a_i|^3 \leq a_m \cdot \sum_{m}^{n} |a_i|^2 \leq |a_m|\right) \\
&\geq \Omega\left(\frac{1}{C^5}\right) \quad \text{(for } c \text{ small enough).}
\end{aligned}$$

This completes the proof. □

10.2. Bounds for the weight on the first level. Applying the Majority Is Stablest theorem for extremely small ρ, it follows that functions with small influences have no more weight at level 1 than Majority has, namely, $\frac{2}{\pi}$ (up to $o(1)$). This fact, stated in Theorem 5, has a very elementary proof which also provides a better bound on the additive term corresponding to the maximal influence.

Proof of Theorem 5. Let ℓ denote the linear part of f, $\ell(x) = \sum_{i=1}^{n} \hat{f}(\{i\})x_i$. We have that $|\hat{f}(\{i\})| \leq \mathrm{Inf}_i(f) \leq \delta$ for all i. Now $\sum_{|S|=1} \hat{f}(S)^2 = \|\ell\|_2^2$ and

$$\begin{aligned}
\|\ell\|_2^2 &= \langle f, \ell \rangle \\
&\leq \|f\|_\infty \|\ell\|_1 \\
&\leq \|\ell\|_1.
\end{aligned}$$

Since all of ℓ's coefficients are small, smaller than δ, we expect ℓ to behave like a Gaussian with mean zero and standard deviation $\|\ell\|_2$; such a Gaussian has L^1-norm equal to $\sqrt{2/\pi}\|\ell\|_2$. Several error bounds on the central limit theorem exist to this effect; the sharpest is a result of König, Schütt, and Tomczak-Jaegermann [42] which implies that $\|\ell\|_1 \leq \sqrt{2/\pi}\|\ell\|_2 + (C/2)\delta$. Thus

$$\|\ell\|_2^2 \leq \sqrt{2/\pi}\|\ell\|_2 + (C/2)\delta;$$

hence $\|\ell\|_2 \leq \sqrt{1/2\pi} + \sqrt{1/2\pi + C\delta/2}$ and therefore $\|\ell\|_2^2 \leq 2/\pi + C\delta$. □

In the following we improve the bound on the weight of the first level for not necessarily balanced functions with low influences. This result should be compared to the following theorem of Talagrand [54].

THEOREM 12 (Talagrand). *Suppose* $f : \{-1,1\}^n \to \{-1,1\}$ *satisfies* $\Pr[f = 1] = p \leq 1/2$. *Then*

$$\sum_{|S|=1} \hat{f}(S)^2 \leq O(p^2 \log(1/p)).$$

Proof. It will be more convenient to work with the $[0,1]$ valued function $g = \frac{1}{2} + \frac{1}{2}f$ and prove that $\sum_{|S|=1} \hat{g}(S)^2 \leq (U(\mu) + \max\{1, \sqrt{|\Phi^{-1}(\mu)|}\}O(\sqrt{\delta}))^2$. Note that $\mu = \mathbf{E}[g]$. We will assume without loss of generality that $\mu \geq 1/2$ (otherwise look at $\frac{1}{2} - \frac{1}{2}f$).

Let τ denote $(\sum_{|S|=1} \hat{g}(S)^2)^{1/2}$. As in the proof of Theorem 5, we let ℓ be the linear part of g, and we know that all of ℓ's coefficients are at most $\delta/2$. The function $L = \ell/\tau = \sum_S \hat{g}(S)\chi_S/\tau$ is a sum of independent random variables $X_S = \hat{g}(S)\chi_S/\tau$. Clearly, $\mathbf{E}[X_S] = 0$ for all S. Moreover, $\sum_S \mathbf{E}[X_S^2] = 1$ and $\sum_S \mathbf{E}[X_S^3] \leq \max_S |X_S| \leq \delta/(2\tau)$.

Now $\tau^2 = \langle g, \ell\rangle$ and therefore $\tau = \langle g, L\rangle$. We will show below that

$$\tau = \langle g, L\rangle \leq U(\mu) + \max\{1, |\Phi^{-1}(\mu)|\}O(\delta/\tau). \tag{12}$$

Multiplying by τ implies that

$$\left(\tau - \frac{U(\mu)}{2}\right)^2 \leq \frac{U^2(\mu)}{4} + \max\{1, |\Phi^{-1}(\mu)|\}O(\delta),$$

which in turn implies that

$$\tau \leq U(\mu) + \max\{1, \sqrt{|\Phi^{-1}(\mu)|}\}O(\sqrt{\delta}).$$

Finally, we will conclude that

$$\tau^2 \leq \left(U(\mu) + \max\{1, \sqrt{|\Phi^{-1}(\mu)|}\}O(\sqrt{\delta})\right)^2.$$

We now prove (12). Let t be a number such that $\Pr[L > t] = \mu$. Since g is a $[0,1]$ valued function, it follows that $\langle g, L\rangle \leq \mathbf{E}[\mathbf{1}_{L>t}L]$.

Letting F denote the cumulative distribution function of L, the Berry–Esseen theorem implies that $\sup_x(1 + |x|^3)|F(x) - \Phi(x)| \leq O(\delta/\tau)$. In particular, $|\Pr[L > t] - \Pr[N(0,1) > t]| \leq O(\delta/(\tau(1+t^3)))$, and hence

$$|\mu - \Phi(-t)| \leq O\left(\frac{\delta}{\tau(1+t^3)}\right). \tag{13}$$

Note that the function U satisfies

$$U'(x) = \phi'(\Phi^{-1}(x)) \cdot ((\Phi^{-1}(x))') = -\Phi^{-1}(x)\phi(\Phi^{-1}(x))\frac{1}{\phi(\Phi^{-1}(x))} = -\Phi^{-1}(x).$$

Therefore $U''(x) = -1/\phi(\Phi^{-1}(x)) = -1/U(x)$. It follows that U is concave.

We now estimate $U(\mu) - \phi(t)$. Since U' is a monotone function, it follows that

$$
\begin{aligned}
(14)\quad |U(\mu) - \phi(t)| = |U(\Phi(-t)) - U(\mu)| &\leq |\Phi(-t) - \mu| \max\{|U'(\Phi(-t))|, |U'(\mu)|\} \\
&\leq \max\{|t|, \Phi^{-1}(\mu)\} O(\delta/(\tau(1+t^3))) \leq \max\{1, |\Phi^{-1}(\mu)|\} O(\delta/\tau).
\end{aligned}
$$

Further,

$$
\begin{aligned}
\langle g, L\rangle \leq \mathbf{E}[\mathbf{1}_{L>t}L] &= t\Pr[L > t] + \int_t^\infty \Pr[L > x]\,dx \\
&= t\Pr[L > t] + \int_t^\infty \Pr[N(0,1) > x]\,dx + \int_t^\infty (F(x) - \Phi(x))\,dx \\
&= t\mu - t\Phi(-t) + \phi(t) + \int_t^\infty (F(x) - \Phi(x))\,dx \\
&\leq \phi(t) + |t| \cdot |\mu - \Phi(-t)| + \int_t^\infty |F(x) - \Phi(x)|\,dx \\
&\leq \phi(t) + \frac{|t|}{1+|t|^3} O\left(\frac{\delta}{\tau}\right) + O\left(\frac{\delta}{\tau}\right) \int_t^\infty \frac{1}{(1+|x|^3)}\,dx \quad \text{(by (13) and Berry–Esseen)} \\
&= \phi(t) + O\left(\frac{\delta}{\tau(1+t^2)}\right) \quad \text{(by (14))} \\
&\leq U(\mu) + \max\{1, |\Phi^{-1}(\mu)|\} O\left(\frac{\delta}{\tau}\right),
\end{aligned}
$$

which proves (12) as needed. □

11. Constraint satisfaction problems over $[q]$.

So far in this paper, we have mostly focused on 2-CSPs in which the variables are binary—i.e., take values in the alphabet $\{-1, 1\}$. The exception is the Unique Label Cover problem, which can be thought of as a 2-CSP where the set of values a variable can take is very large. In this section we develop our techniques for 2-CSPs over large alphabets, specifically the alphabet $[q] = \{1, 2, \ldots, q\}$ for $q \geq 2$. We will be concerned in particular with the MAX-2LIN(q) and MAX-q-CUT (i.e., approximate graph q-coloring) problems, and we will mostly be interested in the asymptotics when $q \to \infty$.

11.1. Γ-MAX-2LIN(q) and MAX-q-CUT.

The MAX-q-CUT problem is a natural generalization of MAX-CUT; its formal definition is as follows.

DEFINITION 13 (MAX-q-CUT). *Given a weighted graph $G = (V, W)$, where $W : V \times V \to \mathbb{R}_+$, the* MAX-$q$-CUT *problem is that of finding a partition of V into q sets $V_1, \ldots, V_q$ in such a way as to maximize the weight of edges between the different parts, $\sum_{i \neq j} \sum_{v \in V_i, w \in V_j} W(v, w)$.*

The MAX-2LIN(q) problem is defined as follows.

DEFINITION 14 (MAX-2LIN(q)). *Given a system of m linear equations mod q each having at most two variables, along with nonnegative weights $w_1, \ldots, w_m$ for these equations, the* MAX-2LIN(q) *problem is to find an assignment to the variables maximizing the total weight of satisfied equations.*

Observe that the MAX-2-CUT problem can be viewed as a special case of the MAX-2LIN(2) problem, by associating the vertices with variables and the edges with equations $x_u - x_v = 1$. However, for larger q, MAX-q-CUT is more naturally viewed as a special case of the problem of finding assignments for 2-variable linear *inequations* mod q.

The best approximation algorithm for MAX-q-CUT was obtained by Frieze and Jerrum in [26]; this paper gives a $(1 - 1/q + 2(\ln q)/q^2)$-approximation. The best NP-hardness result known is due to Kann et al. [36], who proved that $(1 - 1/(34q))$-approximating is hard. As for the approximability of MAX-2LIN(q), the best algorithm known was given very recently by Charikar, Makarychev, and Makarychev in [11], as discussed in subsection 6.3 (their algorithm is actually for the more general problem of not-necessarily-bipartite Unique Label Cover with label size q). When the optimal fraction of satisfiable constraints is $1 - \eta$, their semidefinite programming algorithm produces a solution satisfying about a fraction $(1/q)^{\eta/(2-3\eta)}$. The best known NP-hardness results come from a recent work of Feige and Reichman [22]. They show that it is NP-hard to approximate MAX-2LIN(q) to within a factor of $1/q^{\beta}$ for some universal constant $\beta > 0$; however, this hardness is located at a gap of ϵ vs. ϵ/q^{β}. In particular, given an instance with optimum fraction $1 - \eta$, Feige and Reichman can show only that it is NP-hard to find a solution with value $1 - C\eta$ for some relatively small constant $C > 1$. Thus with current knowledge, given a $(1 - \eta)$-satisfiable instance, we do not know whether one can satisfy almost all the constraints in polynomial time, or whether it is impossible to go beyond a very small fraction of them.

A special case of the MAX-2LIN(q) problem which seems somewhat easier algorithmically [11, 2] occurs when all the equations in the instance are of the form $x_i - x_j = c_{ij}$.

Definition 15 (Γ-MAX-2LIN(q)). Γ-MAX-2LIN(q) *is the special case of MAX-2LIN(q) in which each equation is of the form* $x_i - x_j = c_{ij}$.

Our hardness results hold even for Γ-MAX-2LIN(q). The Γ notation is essentially from Håstad [33]; we use it because our results actually hold equally well for the problem of satisfying equations of the form $x_i x_j^{-1} = c_{ij}$ over any fixed abelian group Γ of order q, not just $\mathbb{Z}_q$.

11.2. Analytic notions. We would like to generalize our notions of noise stability, influences, and Fourier expansions to q-ary functions, $f : [q]^n \to [q]$. Some of the definitions below were already given in subsection 4.2, but we repeat them here for clarity and convenience.

The way we treat the finite set $[q]$ in the domain and in the range of q-ary functions will be different. In the domain, $[q]$ and $[q]^n$ will be treated simply as finite probability spaces under the uniform measure, with no extra structure. In the range, we would like to embed $[q]$ into a larger space. Recall that for boolean function we identified the range with the two points $-1, 1 \in \mathbb{R}$ and then considered relaxed functions taking values in their convex hull. In the q-ary case we identify the elements of $[q]$ with the standard basis vectors in $\mathbb{R}^q$. A relaxed q-ary function will thus map $[q]^n$ into the simplex which is the convex hull of these vectors.

Definition 16. *Let* Δ_q *denote the* $(q-1)$*-dimensional simplex naturally embedded in* $\mathbb{R}^q$*, i.e., the convex hull of the* q *standard basis vectors. We call functions* $f : [q]^n \to \Delta_q$ relaxed q-ary functions.

We will also define the notion of a balanced function.

Definition 17. *A function* $f : [q]^n \to [q]$ *is called* balanced *if it obtains each value* $i \in [q]$ *in its range equally often. A relaxed function* $f : [q]^n \to \Delta_q$ *is called* balanced *if* $\mathbf{E}[f(x)] = (1/q, \ldots, 1/q)$.

Since a relaxed q-ary function f maps $[q]^n$ into $\mathbb{R}^q$, it can be viewed as a vector $f = (f_1, \ldots, f_q)$ of real-valued functions over $[q]^n$. We define the noise stability both for real-valued and Δ_q-valued functions.

Definition 18. *Let* $-\frac{1}{q-1} \le \rho \le 1$ *and let* x *and* y *be* $[q]^n$*-valued random variables. We say that* x *and* y *are a* ρ-correlated pair *if* x *is uniformly distributed on* $[q]^n$*, and* y *is formed from* x *by choosing each* y_i *so that* $\Pr[y_i = a] = \delta_{\{x_i=a\}}\rho + \frac{1-\rho}{q}$ *for each* a*, independently for each* i*. Note that for* $0 \le \rho \le 1$*, it is equivalent to say that each coordinate* y_i *is independently chosen to be* x_i *with probability* ρ *and is a uniformly random element of* $[q]$ *otherwise.*

Definition 19. *Let* $f : [q]^n \to \Delta_q$ *and let* $-\frac{1}{q-1} \le \rho \le 1$*. The* noise stability of f at ρ *is defined to be*

$$\mathbb{S}_\rho(f) = \mathop{\mathbf{E}}_{x,y}[\langle f(x), f(y)\rangle],$$

where x *and* y *are a* ρ*-correlated pair. Equivalently, we may define the noise stability of functions* $g : [q]^n \to \mathbb{R}$ *via*

$$\mathbb{S}_\rho(g) = \mathop{\mathbf{E}}_{x,y}[g(x)g(y)]$$

and then denoting by f^i *the* i*th coordinate projection of* f*, we have* $\mathbb{S}_\rho(f) = \sum_{i=1}^n \mathbb{S}_\rho(f^i)$*.*

We remark that when f's range is simply $[q]$ (as embedded in Δ_q), the quantity $\mathbb{S}_\rho(f)$ is simply the probability that $f(x) = f(y)$ when x and y are a ρ-correlated pair. For example, the noise stability at ρ of a dictator function $f : [q]^n \to [q]$ is equal to $\rho + \frac{1}{q}(1-\rho)$.

The definition of influences is very similar to that in the boolean case.

Definition 20. *Let* $f : [q]^n \to \Delta_q$*. For* $1 \le i \le n$*, the* influence of the ith coordinate on f *is defined to be*

$$\mathrm{Inf}_i(f) = \mathop{\mathbf{E}}_{x_1,\dots,x_{i-1},x_{i+1},\dots,x_n}[\mathrm{Var}_{x_i}[f^i(x_1,\dots,x_n)]],$$

where $\mathrm{Var}[f]$ *denotes* $\mathbf{E}[\langle f, f\rangle] - \langle \mathbf{E}[f], \mathbf{E}[f]\rangle$*.*

The space X of all functions $f : [q]^n \to \mathbb{R}^d$ (we use either $d = q$ or $d = 1$) is an inner product space with inner product

$$\langle f, g\rangle = \mathbf{E}_x[\langle f(x), g(x)\rangle]$$

and associated norm denoted $\|\cdot\|$. Given $x \in [q]^n$, write x_S for $\{x_i : i \in S\}$. It is well known that X can be written as an orthogonal sum of spaces $X = \oplus_{S\subset[n]} X_S$, where X_S denotes the space of all functions $f : [q]^n \to \mathbb{R}^d$ such that

- $f(x)$ depends only on x_S for all x, and
- f is orthogonal to all functions in the spaces $X_{S'}$ for $S' \subsetneq S$.

Thus we can write any $f : [q]^n \to \mathbb{R}^q$ as

$$f(x) = \sum_{S\subset[n]} f_S(x), \tag{15}$$

where $f_S(x)$ is the projection of f onto the space X_S. Parseval's identity holds for this expansion:

$$\|f\|_2^2 = \sum_{S\subseteq[n]} \|f_S\|_2^2.$$

For $-\frac{1}{q-1} \le \rho \le 1$, we can define the Bonami–Beckner operator on X in the obvious way, $T_\rho(f)(x) = \mathbf{E}_y[f(y)]$, where y is ρ-correlated to x. We have that if $f : [q]^n \to \Delta_q$, then $T_\rho f$ also has range Δ_q, and that if f is balanced, then so too is $T_\rho f$. We also have that $\mathbb{S}_\rho(f) = \langle f, T_\rho f\rangle$. The formula for noise stability from Proposition 2 holds in this setting:

$$\mathbb{S}_\rho(f) = \sum_{S\subseteq[n]} \rho^{|S|}\|f_S\|_2^2; \tag{16}$$

this follows from the following easy proposition, familiar from the boolean case.

PROPOSITION 9.

$$T_\rho(f) = \sum_{S\subseteq[n]} \rho^{|S|} f_S.$$

Proof. It is easy to see that for each x, $(T_\rho f)(x)$ is a polynomial in ρ. Therefore it suffices to prove the claim for $0 \le \rho \le 1$. Clearly, T_ρ is linear, and therefore it suffices to show that if $f \in X_S$, then $T_\rho f = \rho^{|S|} f$. From the definition of the space X_S, it follows that for every subset $S' \subsetneq S$ and for every vector of values z of size $|S'|$ it holds that $\mathbf{E}[f(y) \mid y_{S'} = z] = 0$. Now for $0 \le \rho \le 1$,

$$T_\rho f(x) = \sum_{S'\subset S} \rho^{|S'|}(1-\rho)^{|S|-|S'|} \cdot \mathbf{E}[f(y) \mid y_{S'} = x_{S'}] = \rho^{|S|} f(x),$$

as needed. □

In a similar fashion, it is easy to verify that a formula similar to (5) holds in the q-ary case:

$$\mathrm{Inf}_i(f) = \sum_{S\ni i} \|f_S\|_2^2.$$

Finally, we define low degree influences as in Definition 11.

PROPOSITION 10. *Let $f : [q]^n \to \mathbb{R}^d$ and $k \ge 1$. Then we define*

$$\mathrm{Inf}_i^{\le k}(f) = \sum_{\substack{S\ni i\\ |S|\le k}} \|f_S\|_2^2.$$

11.3. Stability estimates. In this subsection we analyze the plurality function and give estimates on $\Lambda_\rho(\mu)$, proving Theorem 7, Proposition 1, and Corollary 3. We also prove some other estimates on $\Lambda_\rho(\mu)$ for very small ρ parameters that are required for our MAX-q-CUT reduction. We begin with the proof of Theorem 7.

Proof of Theorem 7. Suppose we choose $x \in [q]^n$ at random and let y be a ρ-correlated copy of x. For each $i \in [q]$, let u_i denote the number of coordinates in x taking the value i, and let v_i denote the number of coordinates of y taking the value i. We wish to apply the multidimensional central limit theorem (specifically, Theorem 10) to calculate the stability of plurality, which is given by

$$\begin{aligned} &q\Pr\Big[u_1 \ge \max_{1\le i\le q} u_i \text{ and } v_1 \ge \max_{1\le i\le q} v_i\Big] \\ &\quad = q\Pr\Big[n - \sum_{i\ge 2} u_i \ge \max_{2\le i\le q} u_i \text{ and } n - \sum_{i\ge 2} v_i \ge \max_{2\le i\le q} v_i\Big]. \end{aligned}$$

Let us define the following vectors in $\mathbb{R}^{q-1}$: let $e_2, \dots, e_q$ denote the $q-1$ unit vectors, let r denote the vector $(1/q, \dots, 1/q)$, let r_1 denote the 0 vector, and let $r_i = e_i - r$ for $i = 2 \dots q$. Consider random variables $(\mathbf{X}_i, \mathbf{Y}_i)$ taking values in $\mathbb{R}^{2q-2}$ where

$$\Pr[(\mathbf{X}_i, \mathbf{Y}_i) = (r_a, r_b)] = \frac{\rho}{q}\delta_{\{a=b\}} + \frac{1-\rho}{q^2}. \tag{17}$$

Note that $(u_2, \dots, u_q, v_2, \dots, v_q) = nr + \sum_{i=1}^{n}(\mathbf{X}_i, \mathbf{Y}_i)$, where $(\mathbf{X}_i, \mathbf{Y}_i)$ are the i.i.d. random variables given in (17) and that the vectors (X_i, Y_i) have mean 0. Writing A for the $(q-1) \times (q-1)$ matrix given by $A_{i,j} = \delta_{\{i=j\}}/q - 1/q^2$, the covariance matrix V of $(\mathbf{X}_i, \mathbf{Y}_i)$ is given by

$$V = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \otimes A = \begin{pmatrix} A & \rho A \\ \rho A & A \end{pmatrix}.$$

Thus the eigenvalues of V are $1/q, 1/q^2, \rho/q$, and ρ/q^2. Finally, the third norm of $(\mathbf{X}_i, \mathbf{Y}_i)$ is at most 4. We may thus apply Theorem 10 to obtain

$$\begin{aligned}\lim_{n\to\infty} \Pr\Big[u_1 \ge \max_{1\le i\le q} u_i \ \text{ and } \quad v_1 \ge \max_{1\le i \le q} v_i\Big]\\ = \Pr\left[-\sum_{i=2}^{q} N_i \ge \max_{2\le i\le q} N_i \ \text{ and } \ -\sum_{i=2}^{q} M_i \ge \max_{2\le i\le q} M_i\right],\end{aligned}$$

where $(N_i, M_i)_{i=2}^{q}$ is a normal vector with covariance matrix V. Letting $N_1 = -\sum_{i=2}^{q} N_i$ and $M_1 = -\sum_{i=2}^{q} M_i$, we see that $(N_1, \dots, N_q, M_1, \dots, M_q)$ is a zero mean normal vector with covariance matrix $\left(\begin{smallmatrix} 1 & \rho \\ \rho & 1 \end{smallmatrix}\right) \otimes B$, where B is the $q \times q$ matrix given by $B_{i,j} = \delta_{\{i=j\}}/q - 1/q^2$. Finally, let $(U_1, V_1), \dots, (U_q, V_q)$ be a collection of i.i.d. mean zero normal vectors in $\mathbb{R}^2$, where (U_i, V_i) has the covariance matrix $\left(\begin{smallmatrix} 1 & \rho \\ \rho & 1 \end{smallmatrix}\right)$. It is then easy to see that $(N_q, \dots, N_q, M_q, \dots, M_q)$ has the same covariance matrix as the normal vector

$$\sqrt{\frac{q}{q-1}}\left(U_1 - \frac{1}{q}\sum_{j=1}^{q} U_j, \dots, U_q - \frac{1}{q}\sum_{j=1}^{q} U_j, V_1 - \frac{1}{q}\sum_{j=1}^{q} V_j, \dots, U_q - \frac{1}{q}\sum_{j=1}^{q} V_j\right).$$

Since both vectors are normal, they have the same distribution. Hence the stability of Plurality is given by

$$q\Pr\Big[M_1 = \max_{1\le j\le q} M_j \ \text{ and } \ N_1 = \max_j N_j\Big] = q\Pr\Big[U_1 = \max_{1\le j\le q} U_j \ \text{ and } \ V_1 = \max_j V_j\Big],$$

and this completes the proof of Theorem 7. □

Let us now move to discussing estimates of $\Lambda_\rho(\mu)$, proving Proposition 1.

Proof of Proposition 1. The proof of Lemma 11.1 in [13] gives

$$\Lambda_\rho(\mu) = \frac{1}{2\pi\sqrt{1-\rho^2}\cdot t^2}\exp\left(-\frac{t^2}{1+\rho}\right)\int_0^\infty\int_0^\infty \exp(-g(u,v))\,du\,dv, \tag{18}$$

where

$$g(u,v) = \frac{u+v}{1+\rho} + \frac{(u-v)^2 + 2(1-\rho)uv}{2(1-\rho^2)t^2}.$$

Since the range of integration is $u, v \geq 0$, we have $g(u,v) \geq h(u,v)$, where

$$h(u,v) = \frac{u+v}{1+\rho} + \frac{(u-v)^2}{2(1-\rho^2)t^2}.$$

We can therefore replace g by h in the integral and get

$$\begin{aligned}\int_0^\infty \int_0^\infty \exp(-g(u,v))\, du\, dv &\leq \int_0^\infty \int_0^\infty \exp(-h(u,v))\, du\, dv \\ &= \sqrt{2\pi}(1+\rho)\sqrt{1-\rho^2}\cdot t \cdot \exp\left(\frac{1-\rho}{1+\rho}\cdot\frac{t^2}{2}\right)\cdot N\Big(t\sqrt{\tfrac{1-\rho}{1+\rho}}\Big),\end{aligned} \tag{19}$$

where the integral computation follows straightforwardly after the change of variables $r = u+v$, $s = u-v$. Combining (18) and (19) completes the proof. □

Proposition 1 leads to the asymptotic estimates stated in Corollary 3.

Proof of Corollary 3. Part 1 follows simply from the well-known fact that $N(t) \sim \phi(t)/t$ as $t \to \infty$. As a side note, it is simple to see from its definition that $\mu \cdot N(t\sqrt{(1-\rho)/(1+\rho)})$ is a lower bound on $\Lambda_\rho(\mu)$. Part 2 of the Corollary is Lemma 11.1 of de Klerk, Pasechnik, and Warners [13]. Parts 3 and 4 follow straightforwardly from part 1; the bound written in part 3 actually neglects an additional negative power of $\ln q$ for simplicity. □

Finally, our hardness result for MAX-q-CUT requires the following more careful analysis of $\Lambda_\rho(\mu)$ in the case that ρ is very small.

PROPOSITION 11. *Let $\mu > 0$ be small and let $0 < \rho \leq \frac{1}{\ln^3(1/\mu)}$. Then*

$$\Lambda_\rho(\mu) \leq \mu\left(\mu + \rho\cdot 2\mu\ln\left(\frac{1}{\mu}\right)\cdot\left(1 + O\left(\tfrac{\ln\ln(1/\mu)}{\ln(1/\mu)} + \tfrac{\ln\ln(1/\rho)}{\ln(1/\mu)}\right)\right)\right).$$

Proof. Recall that $\Lambda_\rho(\mu) = \Pr[X \geq t, X' \geq t]$, where X is a standard Gaussian, $X' = \rho X + \sqrt{1-\rho^2}\, Y$ with Y an independent standard Gaussian, and $t = N^{-1}(\mu)$. (We use the functions ϕ and N from Proposition 1.) The probability that $X \geq t$ is μ, and so we need to show that

$$\Pr[X' \geq t \mid X \geq t] \leq \mu + \rho\cdot 2\mu\ln\left(\frac{1}{\mu}\right)\cdot\left(1 + O\left(\tfrac{\ln\ln(1/\mu)}{\ln(1/\mu)} + \tfrac{\ln\ln(1/\rho)}{\ln(1/\mu)}\right)\right). \tag{20}$$

Let us first estimate

$$\Pr[X' \geq t \mid X = t(1+\alpha)]$$

for $1/\ln(1/\mu) \leq \alpha \leq \ln(1/\rho)$. We have

$$\begin{aligned}\Pr[X' \geq t \mid X = t(1+\alpha)] &= \Pr[\rho X + \sqrt{1-\rho^2}\, Y \geq t \mid X = t(1+\alpha)] \\ &\leq \Pr[Y \geq (t - \rho t(1+\alpha))/\sqrt{1-\rho^2}] \\ &\leq \Pr[Y \geq t - \rho t(1+2\alpha)],\end{aligned}$$

where we have used that $\rho \leq 1/\ln(1/\mu) \leq \alpha$. Now $\Pr[Y \geq t - \beta] \leq \Pr[Y \geq t] + \beta\phi(t-\beta) = \mu + \beta\phi(t-\beta)$. We can upper bound $\phi(t-\beta)$ by expanding its definition and using the well-known fact $\phi(t) \leq tN(t) + O(1/t^2)$ along with $N(t) = \mu$. With our particular $\beta = \rho t(1+2\alpha)$ we get that

$$\phi(t-\beta) \leq t\mu(1 + O(1/\ln(1/\mu))),$$

where this also uses $\alpha \leq \log(1/\rho)$ and $\rho \leq 1/\ln^3(1/\mu)$. We thus conclude that

$$\Pr[X' \geq t \mid X = t(1+\alpha)] \leq \mu + \rho \cdot 2\mu \ln(1/\mu) \cdot (1 + O(\alpha)), \tag{21}$$

where we have also used $t \leq \sqrt{2\ln(1/\mu)}$ and $\alpha \geq 1/\ln(1/\mu)$.

Our next task is to estimate $\Pr[X \geq t(1+\alpha) \mid X \geq t]$. This is quite straightforward using $N(x) \sim \phi(x)/x$ and $N(t) = \mu$; the result is that

$$\Pr[X \geq t(1+\alpha) \mid X \geq t] \leq (C\mu^2 \ln(1/\mu))^{\alpha} \tag{22}$$

for some universal constant $C < \infty$ (where we used $\alpha \geq 1/\ln(1/\mu)$).

Set $\delta = A(\frac{\ln\ln(1/\mu)}{\ln(1/\mu)} + \frac{\ln\ln(1/\rho)}{\ln(1/\mu)})$ and $K = B\frac{\ln(1/\rho)}{\ln(1/\mu)} > 1$, where A and B are large universal constants to be chosen later. They will be chosen so that $\delta < K$ (this is possible because $\rho \leq 1/\ln^3(1/\mu)$). Write $\gamma = \Pr[(1+\delta)t \leq X \leq (1+K)t \mid X \geq t]$. We estimate

$$\begin{aligned}\Pr[X' \geq t \mid X \geq t] \leq (1-\gamma)\Pr[X' \geq t \mid t \leq X \leq t(1+\delta)]\\ + \gamma \Pr[X' \geq t \mid (1+\delta)t \leq X \leq (1+K)t]\\ + \Pr[X > (1+K)t \mid X \geq t].\end{aligned}$$

Since it is clear that $\Pr[X' \geq t \mid X \in [a,b]] \leq \Pr[X' \geq t \mid X = b]$, we can use (21) to bound the sum of the first two terms by

$$\mu + \rho \cdot 2\mu \ln(1/\mu)(1 + O(\delta)) + \gamma \cdot O(K\rho \cdot \mu \ln(1/\mu)).$$

The third term, and also γ, are bounded using (22). This gives an overall bound of

$$\begin{aligned}\Pr[X' \geq t \mid X \geq t] \leq \mu + \rho \cdot 2\mu \ln(1/\mu)(1 + O(\delta)) + (C\mu^2 \ln(1/\mu))^{\delta}\\ \cdot O(K\rho \cdot \mu \ln(1/\mu)) + (C\mu^2 \ln(1/\mu))^{K}.\end{aligned}$$

It is now relatively easy to check that we can take A and B large enough so that the above quantity is bounded as in (20), completing the proof. (One can take A so that the third term above is smaller than $B \cdot \rho\mu \ln\ln\frac{1}{\rho}$ and then take B large enough so that both $K > \delta$ and the last term is smaller than $\rho \cdot \mu \ln(1/\mu) \cdot \delta$.) □

Proposition 11 leads to a *lower bound* on the stability of a q-ary function with noise $\rho = -\frac{1}{q-1}$.

PROPOSITION 12. *For any $q \geq 2$, there is a small enough $\delta = \delta(q) > 0$, such that any function $f : [q]^n \to \Delta_q$ with $\mathrm{Inf}_i(f) \leq \delta$ for all $i = 1, \ldots, n$ satisfies*

$$\mathbb{S}_{-\frac{1}{q-1}}(f) \geq 1/q - (2\ln q)/q^2 - C \cdot (\ln\ln q)/q^2,$$

where $C < \infty$ is a universal constant.

Proof. Let $f^i : [q]^n \to [0,1]$ denote the ith coordinate function of f and let $\mu_i = \mathbf{E}[f^i]$. Then

$$\begin{aligned}\mathbb{S}_{-\frac{1}{q-1}}(f^i) &= \|f^i_\emptyset\|_2^2 - \tfrac{1}{q-1}\sum_{|S|=1}\|f^i_S\|_2^2 + (\tfrac{1}{q-1})^2\sum_{|S|=2}\|f^i_S\|_2^2 - \cdots\\ &\geq \|f^i_\emptyset\|_2^2 - \tfrac{1}{q-1}\sum_{|S|=1}\|f^i_S\|_2^2 - (\tfrac{1}{q-1})^2\sum_{|S|=2}\|f^i_S\|_2^2 - \cdots\\ &= 2\mu_i^2 - \mathbb{S}_{\frac{1}{q-1}}(f^i).\end{aligned}$$

Choosing δ to be small enough as function of q, we obtain from the MOO theorem that $\mathbb{S}_{\frac{1}{q-1}}(f^i) \leq \Lambda_{\frac{1}{q-1}}(\mu_i) + \epsilon$, where ϵ is, say, $1/q^3$. It thus suffices to prove that

$$\sum_{i=1}^{q}\Big[2\mu_i^2 - \Lambda_{\frac{1}{q-1}}(\mu_i)\Big]^+ \geq 1/q - (2\ln q)/q^2 - O(\ln\ln q)/q^2. \tag{23}$$

(Here the notation x^+ means x if $x \geq 0$, 0 otherwise.) We prove this using Proposition 11 and the fact that $\sum \mu_i = 1$. We will first carry out the estimates assuming that all μ_i's satisfy $\frac{1}{q-1} \leq \frac{1}{\ln^3(1/\mu_i)}$.

Suppose that $\mu_i \geq (1/q)^{1/10}$ for some i. Proposition 11 implies in this case that $\Lambda_{\frac{1}{q-1}}(\mu_i) \leq \mu_i^2 + O(\mu^{1/10})$, and so the ith summand already contributes at least $.5\mu_i^2 \geq .5(1/q)^{1/5}$ to the sum in (23), and so the inequality there holds. We may therefore assume that all μ_i's are at most $(1/q)^{1/10}$. With this in hand, Proposition 11 tells us that $\Lambda_{\frac{1}{q-1}}(\mu_i) \leq F(\mu_i)$, where

$$F(\mu_i) = \mu_i^2 + \frac{1}{q-1}\cdot 2\mu_i^2 \ln\left(\frac{1}{\mu_i}\right)\cdot\left(1 + C\frac{\ln\ln q}{\ln q}\right)$$

and $C < \infty$ is some universal constant. Thus

$$\sum_{i=1}^{q}\Big[2\mu_i^2 - \Lambda_{\frac{1}{q-1}}(\mu_i)\Big]^+ \geq \sum_{i=1}^{q}(2\mu_i^2 - F(\mu_i)). \tag{24}$$

It is not hard to check that $2\mu_i^2 - F(\mu_i)$ is a convex function of μ_i so long as μ_i is at most a certain universal constant smaller than 1 (which it is when q is sufficiently large, since all μ_i's are at most $(1/q)^{1/10}$). Using $\sum_{i=1}^q \mu_i = 1$, we conclude that the right-hand side of (24) is minimized when all μ_i's are equal to $1/q$, in which case it equals $1/q - (2\ln q)/q^2 - O(\ln\ln q)/q^2$; thus (23) is verified.

Finally, we consider the possibility that not all μ_i's satisfy $\frac{1}{q-1} \leq \frac{1}{\ln^3(1/\mu_i)}$. In this case, some of the term-by-term inequalities going into (24) may no longer hold. For such inequalities, though, the left-hand side term is always nonnegative and the right-hand side term is exponentially small in a power of q. Hence (24) still holds up to an additive term exponentially small in q, which is negligible; thus the argument above is unaffected. □

11.4. Hardness results for MAX-q-CUT and Γ-MAX-2LIN(q). This section is devoted to the proofs of Theorems 8 and 9. The proofs are similar to that of Theorem 1 in section 8, and so we omit some details.

As a preliminary technical step, we need the analogue of Proposition 4 for the MOO theorem and for Proposition 12.

PROPOSITION 13. *Both the MOO theorem and Proposition* 12 *remain true if the assumption that* $\mathrm{Inf}_i(f) \leq \delta$ *for all* i *is replaced by the assumption that* $\mathrm{Inf}_i^{\leq k'}(f) \leq \delta'$, *where* δ' *and* k' *are universal functions of* ϵ *and* ρ.

Proof (sketch). The proof is essentially the same as that of Proposition 4; one requires the following facts:

$$\sum_i \mathrm{Inf}_i^{\leq k}(f) \leq k,$$

$$\sum_{|S|>k} \|(T_{1-\gamma}f)_S\|_2^2 \leq (1-\gamma)^{2k},$$

which indeed hold for q-ary functions $f : [q]^n \to [0,1]$, as can easily be seen from the facts in subsection 7.2. □

We will also need to define the q-ary analogue of the Long Code.

DEFINITION 21 (q-ary Long Code). *The* q*-ary Long Code of an element* $i \in [M]$ *is the* q*-ary function* $f : [q]^M \to [q]$ *defined by* $f(x) = x_i$.

Hardness of MAX-q-CUT. We now begin the reduction from Unique Label Cover to MAX-q-CUT. As mentioned, the reduction is similar to the one in section 8; the main difference is that we use q-ary Long Codes and we fix ρ to be $-\frac{1}{q-1}$. As in section 8, we start with a given instance $\mathcal{L}(V, W, E, [M], \{\sigma_{v,w}\})$ of Unique Label Cover, and construct an instance of MAX-q-CUT, presented here as a PCP verifier.

The PCP verifier for MAX-q-CUT.

- Pick a vertex $v \in V$ at random and two of its neighbors $w, w' \in W$ at random. Let $\sigma = \sigma_{v,w}$ and $\sigma' = \sigma_{v,w'}$ be the respective bijections for edges (v, w) and (v, w').
- Let f_w and $f_{w'}$ be the supposed q-ary Long Codes of the labels of w and w', respectively.
- Pick $(x, y) \in [q]^M$ to be a $(-\frac{1}{q-1})$-correlated pair; in other words, pick $x \in [q]^M$ uniformly at random and form y by choosing each y_i independently to be a uniformly random element of $[q] \setminus \{x_i\}$.
- Accept iff

$$f_w(x \circ \sigma) \neq f_{w'}(y \circ \sigma').$$

Completeness. If the Unique Label Cover instance has a $(1-\eta)$-satisfying assignment, then the PCP verifier accepts the Long Code encoding of this assignment with probability at least $1 - 2\eta$. This is because whenever the PCP verifier chooses edges (v, w) and (v, w') that are properly labeled, the verifier accepts with probability 1. The Unique Games Conjecture allows us to take η to be an arbitrarily small positive constant.

Soundness. Our goal is to show that if the PCP verifier accepts with probability exceeding $1 - 1/q + (2 \ln q)/q^2 + O(\ln \ln q)/q^2$, then we can derive an assignment for the Unique Label Cover instance that satisfies at least some $\gamma' = \gamma'(q)$ fraction of its edges, independent of the label set size M.

As in section 8, we analyze the soundness by writing the success probability of the PCP in terms of the noise stability of certain averages of the f_w's. (We view these supposed Long Codes $f_w : [q]^n \to [q]$ as having the relaxed range Δ_q.) If the noise stability is large, the MOO theorem implies the existence of influential coordinates, which in turn are used to derive an assignment for the Unique Label Cover instance.

The probability that the PCP verifier accepts is given by

$$\begin{aligned}
\Pr[\mathrm{acc}] &= \mathop{\mathbf{E}}_{v,w,w',x,y}\left[1-\langle f_w(x\circ\sigma), f_{w'}(y\circ\sigma')\rangle\right]\\
&= 1-\mathop{\mathbf{E}}_{v,x,y}\left[\mathop{\mathbf{E}}_{w,w'}\left[\langle f_w(x\circ\sigma), f_{w'}(y\circ\sigma')\rangle\right]\right]\\
&= 1-\mathop{\mathbf{E}}_{v,x,y}\left[\langle \mathop{\mathbf{E}}_{w}[f_w(x\circ\sigma)], \mathop{\mathbf{E}}_{w'}[f_{w'}(y\circ\sigma')]\rangle\right] \quad \text{(using independence of } w \text{ and } w'\text{)}\\
&= 1-\mathop{\mathbf{E}}_{v,x,y}\left[\langle g_v(x), g_v(y)\rangle\right] \quad \left(\text{where we define } g_v(z)=\mathop{\mathbf{E}}_{w\sim v}\left[f_w(z\circ\sigma_{v,w})\right]\right)\\
&= 1-\mathop{\mathbf{E}}_{v}\left[\mathbb{S}_{-\frac{1}{q-1}}(g_v)\right].
\end{aligned}$$

We now proceed as in the proof of Theorem 1, using Proposition 12 in place of the Majority Is Stablest theorem. In particular, writing $\epsilon = (\ln\ln q)/q^2$, we have that if $\Pr[\mathrm{acc}] \geq 1 - 1/q + (2\ln q)/q^2 + (C+1)\cdot(\ln\ln q)/q^2$, then there is some $\epsilon/2$ fraction of "good" v's with $\mathbb{S}_{-\frac{1}{q-1}}(g_v) \leq 1/q - (2\ln q)/q^2 - (C+1/2)\cdot(\ln\ln q)/q^2$. By Proposition 12 such g_v's must have large low-degree influential coordinates, which we can use as Label Cover labels for their v's.

The remainder of the soundness proof is just as it is in the proof of Theorem 1 in section 8. The only difference arises in the analogue of (7) and is essentially notational; we replace this line with

$$\begin{aligned}
\delta &\leq \sum_{\substack{S\ni j\\|S|\leq k}} \|(g_v)_S\|_2^2 = \sum_{\substack{S\ni j\\|S|\leq k}} \|\mathop{\mathbf{E}}_{w}[(f_w)_{\sigma^{-1}(S)}]\|_2^2\\
&\leq \sum_{\substack{S\ni j\\|S|\leq k}} \mathop{\mathbf{E}}_{w}[\|(f_w)_{\sigma^{-1}(S)}\|_2^2] = \mathop{\mathbf{E}}_{w}\left[\mathrm{Inf}^{\leq k}_{\sigma^{-1}(j)}(f_w)\right].
\end{aligned}$$

With this proof of soundness in hand, the proof of Theorem 8 is now complete.

Hardness of MAX-2LIN(q). We move on to our hardness result for Γ-MAX-2LIN(q) and the proof of Theorem 9. The proof is very similar to the one we gave for MAX-q-CUT; the only new technique is the use of the old PCP trick of *folding*.

DEFINITION 22 (additive folding). *Let $f : [q]^M \to [q]$, where the $[q]$ in the domain is viewed as $\mathbb{Z}_q$, the integers mod q. We say that f is* folded *if for every $c \in \mathbb{Z}_q$ and $x \in (\mathbb{Z}_q)^M$ it holds that $f(x + (c, c, \ldots, c)) = f(x) + c$.*

Our PCP verifier for Γ-MAX-2LIN(q) will be able to assume that all the supposed q-ary Long Codes f_w with which it works are folded. This can be done by only making queries $f_w(x)$ when $x_1 = 0$, and simulating other queries using the assumption that the function is folded. In other words, to query $f_w(x_1, \ldots, x_n)$ the verifier instead queries $f(0, x_2 - x_1, \ldots, x_n - x_1)$ and computes the value $f(0, x_2 - x_1, \ldots, x_n - x_1) - x_1$. Note that Long Code functions (dictators) are folded, and that a folded q-ary function must be balanced.

We now give our verifier for Γ-MAX-2LIN(q), parameterized by $0 < \rho < 1$. Given an instance of Unique Label Cover, it proceeds as follows.

The PCP verifier for Γ-MAX-2LIN(q) *with parameter* $0 < \rho < 1$.

- Pick a vertex $v \in V$ at random and two of its neighbors $w, w' \in W$ at random. Let $\sigma = \sigma_{v,w}$ and $\sigma' = \sigma_{v,w'}$ be the respective bijections for edges (v, w) and (v, w').
- Let f_w and $f_{w'}$ be the *folded* supposed q-ary Long Codes of the labels of w and w', respectively.
- Pick $(x, y) \in [q]^M$ to be a ρ-correlated pair.
- Accept iff $f_w(x \circ \sigma) = f_{w'}(y \circ \sigma')$, i.e., iff $f_w(x \circ \sigma) - f_{w'}(y \circ \sigma') = 0$.

This verifier indeed yields a distribution over 2-variable linear equations mod q of the form "$x_i - x_j = c$"; note that since the verifier ensures the functions f_w are folded, the acceptance predicates $f_w(x \circ \sigma) - f_{w'}(y \circ \sigma') = 0$ will really be of the form $(f_w(x') - x'_1) - (f_{w'}(y') - y'_1) = 0$.

Analysis of this PCP verifier's completeness and soundness proceeds very much as it did in the previous proofs. The completeness is at least $(1 - 2\eta)$ times the noise stability at ρ of a q-ary Long Code function, i.e., $(1 - 2\eta)(\rho + \frac{1}{q}(1 - \rho))$. Soundness is again analyzed by arithmetizing the PCP verifier's acceptance probability, which in this case yields

$$\Pr[\text{acc}] = \mathop{\mathbf{E}}_{v}[\mathbb{S}_\rho(g_v)].$$

The functions $g_v : [q]^M \to \Delta_q$ are balanced, being the averages of folded and thus balanced f_w's. Hence their q projection functions $(g_v)^i : [q]^M \to [0, 1]$ all have mean equal to $\frac{1}{q}$. We may thus use the MOO theorem directly (instead of Proposition 12) and bound soundness by $q\Lambda_\rho(\frac{1}{q}) + \epsilon$. This completes the proof of Theorem 9.

Acknowledgments. We thank Jean Bourgain, Steve Evans, Elad Hazan, Gil Kalai, Oded Regev, and Gideon Schechtman for stimulating discussion. We thank Rocco Servedio for pointing out an error in our original proof of Proposition 8. We also thank an anonymous referee for helpful suggestions.

REFERENCES

[1] M. Abramowitz and I. Stegun, *Handbook of Mathematical Functions*, Dover, New York, 1972.

[2] G. Andersson, L. Engebretsen, and J. Håstad, *A new way of using semidefinite programming with applications to linear equations mod p*, J. Algorithms, 39 (2001), pp. 162–204.

[3] M. Bellare, O. Goldreich, and M. Sudan, *Free bits, PCPs and non-approximability—towards tight results*, SIAM J. Comput., 27 (1998), pp. 804–915.

[4] M. Ben-Or and N. Linial, *Collective coin flipping*, in Randomness and Computation, S. Micali, ed., Academic Press, New York, 1990.

[5] I. Benjamini, G. Kalai, and O. Schramm, *Noise sensitivity of boolean functions and applications to percolation*, Inst. Hautes Études Sci. Publ. Math., 90 (1999), pp. 5–43.

[6] R. Bhattacharya and R. Rao, *Normal Approximation and Asymptotic Expansions*, Robert E. Krieger, Melbourne, FL, 1986.

[7] J. Bourgain, *An appendix to* Sharp thresholds of graph properties, and the k-SAT problem*, by E. Friedgut*, J. Amer. Math. Soc., 12 (1999), pp. 1017–1054.

[8] J. Bourgain, *On the distribution of the Fourier spectrum of boolean functions*, Israel J. Math., 131 (2002), pp. 269–276.

[9] J. Bourgain, J. Kahn, G. Kalai, Y. Katznelson, and N. Linial, *The influence of variables in product spaces*, Israel J. Math., 77 (1992), pp. 55–64.

[10] J. Bourgain and G. Kalai, *Influences of variables and threshold intervals under group symmetries*, Geom. Funct. Anal., 7 (1997), pp. 438–461.

[11] M. Charikar, K. Makarychev, and Y. Makarychev, *Near-optimal algorithms for unique games*, in Proceedings of the 38th Annual ACM Symposium on Theory of Computing, 2006, pp. 205–214.

[12] P. Crescenzi, R. Silvestri, and L. Trevisan, *On weighted vs unweighted versions of combinatorial optimization problems*, Inform. Comput., 167 (2001), pp. 10–26.

[13] E. de Klerk, D. Pasechnik, and J. Warners, *On approximate graph colouring and MAX-k-CUT algorithms based on the ϑ-function*, J. Comb. Optim., 8 (2004), pp. 267–294.

[14] I. Dinur, E. Friedgut, G. Kindler, and R. O'Donnell, *On the Fourier tails of bounded functions over the discrete cube*, Israel J. Math., to appear.

[15] I. Dinur, E. Mossel, and O. Regev, *Conditional hardness for approximate coloring*, in Proceedings of the 38th Annual ACM Symposium on Theory of Computing, 2005, pp. 344–353.

[16] I. Dinur and S. Safra, *The importance of being biased*, in Proceedings of the 34th Annual ACM Symposium on Theory of Computing, 2002, pp. 33–42.

[17] U. Feige, M. Karpinski, and M. Langberg, *Improved approximation of MAX-CUT on graphs of bounded degree*, J. Algorithms, 43 (2002), pp. 201–219.

[18] U. Feige, *Randomized rounding of semidefinite programs—variations on the MAX-CUT example*, Lecture Notes in Comput. Sci. 1761, Springer-Verlag, Berlin, 1999, pp. 189–196.

[19] U. Feige and M. Goemans, *Approximating the value of two prover proof systems, with applications to MAX-2SAT and MAX-DICUT*, in Proceedings of the 3rd Annual Israel Symposium on Theory of Computing and Systems, 1995, pp. 182–189.

[20] U. Feige and G. Schechtman, *On the optimality of the random hyperplane rounding technique for MAX-CUT*, Random Structures Algorithms, 20 (2002), pp. 403–440.

[21] W. Feller, *An Introduction to Probability Theory and Its Applications*, John Wiley and Sons, New York, 1968.

[22] U. Fiege and D. Reichman, *On systems of linear equations with two variables per equation*, in APPROX-RANDOM, 2004, pp. 117–127.

[23] E. Friedgut, *Boolean functions with low average sensitivity depend on few coordinates*, Combinatorica, 18 (1998), pp. 474–483.

[24] E. Friedgut and G. Kalai, *Every monotone graph property has a sharp threshold*, Proc. Amer. Math. Soc., 124 (1996), pp. 2993–3002.

[25] E. Friedgut, G. Kalai, and A. Naor, *Boolean functions whose Fourier transform is concentrated at the first two levels*, Adv. in Appl. Math., 29 (2002), pp. 427–437.

[26] A. Frieze and M. Jerrum, *Improved approximation algorithms for MAX k-CUT and MAX BISECTION*, in Integer Programming and Combinatorial Optimization, Lecture Notes in Comput. Sci. 920, E. Balas and J. Clausen, eds., Springer-Verlag, Berlin, 1995, pp. 1–13.

[27] M. Goemans and D. Williamson, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, J. Assoc. Comput. Mach., 42 (1995), pp. 1115–1145.

[28] M. Goemans and D. Williamson, *Approximation algorithms for MAX-3-CUT and other problems via complex semidefinite programming*, J. Comput. System Sci., 68 (2004), pp. 442–470.

[29] G. Guilbaud, *Theories of general interest, and the logical problem of aggregation*, in Readings in Mathematical Social Science, MIT Press, Cambridge, MA, 1966, pp. 262–307.

[30] A. Gupta and K. Talwar, *Approximating Unique Games*, in Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2006.

[31] V. Guruswami, J. Håstad, and M. Sudan, *Hardness of approximate hypergraph coloring*, SIAM J. Comput., 31 (2002), pp. 1663–1686.

[32] J. Håstad, *On a Protocol Possibly Useful for MIN-2SAT*, manuscript, 2001.

[33] J. Håstad, *Some optimal inapproximability results*, J. ACM, 48 (2001), pp. 798–869.

[34] J. Kahn, G. Kalai, and N. Linial, *The influence of variables on boolean functions*, in Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science, 1988, pp. 68–80.

[35] G. Kalai, *A Fourier-theoretic perspective on the Concordet paradox and Arrow's theorem*, Adv. in Appl. Math., 29 (2002), pp. 412–426.

[36] V. Kann, S. Khanna, J. Lagergren, and A. Panconesi, *On the hardness of approximating Max k-Cut and its dual*, Chicago J. Theoret. Comput. Sci., 1997, article 2.

[37] H. Karloff, *How good is the Goemans–Williamson MAX-CUT algorithm?*, SIAM J. Comput., 29 (1999), pp. 336–350.

[38] R. Karp, *Reducibility among combinatorial problems*, in Complexity of Computer Computations, Plenum Press, New York, 1972, pp. 85–103.

[39] S. Khot, *On the power of unique 2-prover 1-round games*, in Proceedings of the 34th Annual ACM Symposium on Theory of Computing, 2002, pp. 767–775.

[40] S. Khot and O. Regev, *Vertex Cover might be hard to approximate to within $2 - \epsilon$*, in Proceedings of the 18th Annual IEEE Conference on Computational Complexity, 2003.

[41] S. Khot and N. Vishnoi, *The Unique Games Conjecture, integrality gap for cut problems and*

embeddability of negative type metrics into l1., in Proceedings of the 46rd Annual IEEE Symposium on Foundations of Computer Science, 2005.

[42] H. König, C. Schütt, and N. Tomczak-Jaegermann, *Projection constants of symmetric spaces and variants of Khintchine's inequality*, J. Reine Angew. Math., 511 (1999), pp. 1–42.

[43] M. Lewin, D. Livnat, and U. Zwick, *Improved rounding techniques for the MAX-2SAT and MAX-DICUT problems*, in Proceedings of the 9th Annual Conference on Integer Programming and Combinatorial Optimization, Springer-Verlag, Berlin, 2002, pp. 67–82.

[44] K. Matulef, R. O'Donnell, R. Rubinfeld, and R. Servedio, *Testing Linear Threshold Functions*, manuscript.

[45] E. Mossel, *Lecture Notes for Stat*206*A,* Sept. 8, 2005, http://www.stat.berkeley.edu/∼mossel/teach/206af05/scribes/sep8.ps.

[46] E. Mossel and R. O'Donnell, *On the noise sensitivity of monotone functions*, in Trends in Mathematics: Mathematics and Computer Science II, B. Chauvin, P. Flajolet, D. Gardy, and A. Mokkadem, eds., Birkhäuser, Basel, 2002.

[47] E. Mossel, R. O'Donnell, and K. Oleszkiewicz, *Noise stability of functions with low influences: Invariance and optimality*, in Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, 2005, pp. 21–30.

[48] E. Mossel, R. O'Donnell, O. Regev, B. Sudakov, and J. Steif, *Non-interactive correlation distillation, inhomogeneous Markov chains, and the reverse Bonami-Beckner inequality*, Israel J. Math., 154 (2006), pp. 299–336.

[49] C. Papadimitriou and M. Yannakakis, *Optimization, approximation, and complexity classes*, J. Comput. System Sci., 43 (1991), pp. 425–440.

[50] S. Sahni and T. Gonzales, *P-complete approximation problems*, J. Assoc. Comput. Mach., 23 (1976), pp. 555–565.

[51] A. Samorodnitsky and L. Trevisan, *A PCP characterization of NP with optimal amortized query complexity*, in Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, 2000, pp. 191–199.

[52] W. Sheppard, *On the application of the theory of error to cases of normal distribution and normal correlation*, Phil. Trans. Royal Soc. London, 192 (1899), pp. 101–168.

[53] M. Talagrand, *On Russo's approximate* 0-1 *law*, Ann. Probab., 22 (1994), pp. 1476–1387.

[54] M. Talagrand, *How much are increasing sets positively correlated?*, Combinatorica, 16 (1996), pp. 243–258.

[55] L. Trevisan, *Approximation algorithms for Unique Games*, in Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, 2005.

[56] L. Trevisan, G. B. Sorkin, M. Sudan, and D. P. Williamson, *Gadgets, approximation, and linear programming*, SIAM J. Comput., 29 (2000), pp. 2074–2097.

[57] U. Zwick, *Outward rotations: a tool for rounding solutions of semidefinite programming relaxations, with applications to MAX-CUT and other problems*, in Proceedings of the 31st Annual ACM Symposium on Theory of Computing, 1999, pp. 679–687.

SIAM J. COMPUT. © 2007 Society for Industrial and Applied Mathematics
Vol. 37, No. 2, pp. 359–379

RANGE-EFFICIENT COUNTING OF DISTINCT ELEMENTS IN A MASSIVE DATA STREAM*

A. PAVAN† AND SRIKANTA TIRTHAPURA‡

Abstract. Efficient one-pass estimation of F_0, the number of distinct elements in a data stream, is a fundamental problem arising in various contexts in databases and networking. We consider *range-efficient estimation of* F_0: estimation of the number of distinct elements in a data stream where each element of the stream is not just a single integer but an interval of integers. We present a randomized algorithm which yields an (ϵ, δ)-approximation of F_0, with the following time and space complexities (n is the size of the universe of the items): (1) The amortized processing time per interval is $O(\log \frac{1}{\delta} \log \frac{n}{\epsilon})$. (2) The workspace used is $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \log n)$ bits. Our algorithm improves upon a previous algorithm by Bar-Yossef, Kumar and Sivakumar [*Proceedings of the* 13*th ACM–SIAM Symposium on Discrete Algorithms (SODA)*, 2002, pp. 623–632], which requires $O(\frac{1}{\epsilon^5} \log \frac{1}{\delta} \log^5 n)$ processing time per item. This algorithm can also be used to compute the max-dominance norm of a stream of multiple signals and significantly improves upon the previous best time and space bounds by Cormode and Muthukrishnan [*Proceedings of the* 11*th European Symposium on Algorithms (ESA)*, Lecture Notes in Comput. Sci. 2938, Springer, Berlin, 2003, pp. 148–160]. This algorithm also provides an efficient solution to the *distinct summation problem*, which arises during data aggregation in sensor networks [*Proceedings of the* 2*nd International Conference on Embedded Networked Sensor Systems*, ACM Press, New York, 2004, pp. 250–262, *Proceedings of the* 20*th International Conference on Data Engineering (ICDE)*, 2004, pp. 449–460].

Key words. data streams, range efficiency, distinct elements, reductions, sensor networks

AMS subject classifications. 68W20, 68W25, 68W40, 68P15

DOI. 10.1137/050643672

1. Introduction.

One of the most significant successes of research on data stream processing has been the efficient estimation of the frequency moments of a stream in one-pass using limited space and time per item. An especially important aggregate is the number of distinct elements in a stream, which is often referred to as the zeroth frequency moment and denoted by F_0. The counting of distinct elements arises in numerous applications involving a stream. For example, most database query optimization algorithms need an estimate of F_0 of the data set [HNSS95]. In network traffic monitoring, this can be used to compute the number of distinct web pages requested from a web site or the number of distinct source addresses among all Internet protocol (IP) packets passing through a router. Further, the computation of many other aggregates of a data stream can be reduced to the computation of F_0.

In most data stream algorithms in the literature the following model is studied: Each element in the stream is a single item (which can be represented by an integer), and the algorithm needs to process this item "efficiently," both with respect to time and space; i.e., the time taken to process each element should be small, and the total

*Received by the editors October 27, 2005; accepted for publication (in revised form) November 13, 2006; published electronically May 16, 2007. A preliminary version of this article appeared in *Proceedings of the* 21*st International Conference on Data Engineering (ICDE)*, IEEE Computer Society, 2005.

http://www.siam.org/journals/sicomp/37-2/64367.html

†Department of Computer Science, Iowa State University, Ames, IA 50010 (pavan@cs.iastate.edu). This author's research was supported in part by NSF grant CCF-0430807.

‡Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50010 (snt@iastate.edu). This author's research was supported in part by NSF grant CNS-0520102.

workspace used should be small. Many algorithms have been designed in this model to estimate frequency moments [AMS99, GT01, BYJK^{+}02, CK04] and other aggregates [FKSV02, DGIM02, CDIM02, AJKS02, Mut05, BBD^{+}02] of massive data sets.

However, in many cases it is advantageous to design algorithms which work on a more general data stream, where each element of the stream is not a single item but a *list of items*. In a stream of integers, this list often takes the form of an interval of integers. To motivate this requirement for processing intervals of integers, we give some examples below.

Bar-Yossef, Kumar, and Sivakumar [BYKS02] formalize the concept of *reductions* between data stream problems and demonstrate that, in many cases, reductions naturally lead to the need for processing a list of items quickly, much faster than processing them one by one. They consider the problem of estimating the number of triangles in a graph G, where the edges of G arrive as a stream in an arbitrary order. They present an algorithm that uses a reduction from the problem of computing the number of triangles in a graph to the problem of computing the zeroth and second frequency moments (denoted F_0 and F_2, respectively) of a stream of integers. However, for each edge e in the stream, the reduction produces a list of integers, and the size of each such list could be as large as n, the number of vertices in the graph. If one used an algorithm for F_0 or F_2 which processed these integers one by one, the processing time per edge would be $\Omega(n)$, which is prohibitive. Thus, this reduction needs an algorithm for computing F_0 and F_2 which can handle a list of integers efficiently, and such an algorithm is called *list-efficient*. In many cases, including the above, these lists are simply intervals of integers, and an algorithm which can handle such intervals efficiently is called *range-efficient*.

Another application of such list-efficient and range-efficient algorithms is in aggregate computation over sensor networks [NGSA04, CLKB04]. The goal here is to compute the sum (or average) of a stream of sensor observations. However, due to multipath routing of data, the same observation could be repeated multiple times at a node, and the sum should be computed over only *distinct elements* of the stream. This leads to the *distinct summation* problem defined below, which was studied by Considine et al. [CLKB04] and Nath et al. [NGSA04]. Given a multiset of items $M = \{x_1, x_2, \dots\}$, where $x_i = (k_i, c_i)$, compute $\sum_{distinct(k_i,c_i)\in M} c_i$. The distinct summation problem can be reduced to F_0 computation as follows: For each $x_i = (k_i, c_i)$, generate a list l_i of c_i distinct but consecutive integers such that for distinct x_i's the lists l_i do not overlap, but if element x_i reappeared, the same list l_i would be generated. An F_0 algorithm on this stream of l_i's will give the distinct summation required, but the algorithm should be able to efficiently process a list of elements. Each list l_i is a range of integers, so this needs a range-efficient algorithm for F_0.

Yet another application of range-efficient algorithms for F_0 is in computing the *max-dominance norm* of multiple data streams. The concept of the max-dominance norm is useful in financial applications [CM03] and IP network monitoring [CM03]. The max-dominance norm problem is as follows: Given k streams of m integers each, let
$a_{i,j}, i = 1, 2, \dots, k, j = 1, 2, \dots, m$, represent the jth element of the ith stream. The max-dominance norm is defined as $\sum_{j=1}^{m} \max_{1\le i\le k} a_{i,j}$. Assume for all $a_{i,j}$, $1 \le a_{i,j} \le n$. In section 5 we show that the computation of the max-dominance norm can be reduced to range-efficient F_0 and derive efficient algorithms using this reduction.

The above examples illustrate that range-efficient computation of F_0 is a fundamental problem, useful in diverse scenarios involving data streams. In this paper, we present a novel algorithm for range-efficient computation of F_0 of a data stream that provides the current best time and space bounds. It is well known [AMS99] that exact computation of the F_0 of a data stream requires space linear in the size of the input in the worst case. In fact, even deterministically approximating F_0 using sublinear space is impossible. For processing massive data streams, it is clearly infeasible to use space linear in the input size. Thus, we focus on designing randomized approximation schemes for range-efficient computation of F_0.

DEFINITION 1. *For parameters* $0 < \epsilon < 1$ *and* $0 < \delta < 1$*, an* (ϵ, δ)*-estimator for a number* Y *is a random variable* X *such that* $\Pr[|X - Y| > \epsilon Y] < \delta$.

1.1. Our results.

We consider the problem of range-efficient computation of F_0, defined as follows. The input stream is $R = r_1, r_2, \ldots, r_m$, where each stream element $r_i = [x_i, y_i] \subset [1, n]$ is an interval of integers $x_i, x_i + 1, \ldots, y_i$. The length of an interval r_i could be any number between 1 and n. Two parameters, $0 < \epsilon < 1$ and $0 < \delta < 1$, are supplied by the user.

The algorithm is allowed to view each element of the stream only once and has a limited workspace. It is required to process each item quickly. Whenever the user desires, it is required to output the F_0 of the input stream, i.e., the total number of distinct integers contained in all the intervals in the input stream R. For example, if the input stream was $[1, 10], [2, 5], [5, 12], [41, 50]$, then $F_0 = |[1, 12] \cup [41, 50]| = 22$.

We present an algorithm with the following time and space complexities:

- the amortized processing time per interval is $O(\log \frac{1}{\delta} \log \frac{n}{\epsilon})$;
- the time to answer a query for F_0 at anytime is $O(1)$;
- the workspace used is $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \log n)$ bits.

Prior to this work, the most efficient algorithm for range-efficient F_0 computation was by Bar-Yossef, Kumar, and Sivakumar [BYKS02] and took $O(\frac{1}{\epsilon^5} \log \frac{1}{\delta} \log^5 n)$ processing time per interval and $O(\frac{1}{\epsilon^3} \log \frac{1}{\delta} \log n)$ space. Our algorithm performs significantly better with respect to time and better with respect to space.

Extensions to the basic algorithm. The basic algorithm for range-efficient F_0 can be extended to the following more general scenarios:

- It can process a list of integers that is in an arithmetic progression as efficiently as it can handle an interval of integers.
- The algorithm can also be used in the *distributed streams* model, where the input stream is split across multiple parties and F_0 has to be computed on the union of the streams observed by all the parties.
- If the input consists of multidimensional ranges, then the algorithm can be made range-efficient in every coordinate [BYKS02].

Our algorithm is based on random sampling. The set of distinct elements in the stream is sampled at an appropriate probability, and this sample is used to determine the number of distinct elements. The random sampling algorithm is based on the algorithm of Gibbons and Tirthapura [GT01] for counting the number of distinct elements in a stream of integers; however, their algorithm [GT01] was not range-efficient.

A key technical ingredient in our algorithm is a novel *range sampling* algorithm, which can quickly determine the size of a sample resulting from an interval of integers. Using this range sampling algorithm, an interval of integers can be processed by the algorithm much faster than processing them one by one, and this ultimately leads to a faster range-efficient F_0 algorithm.

1.2. Applications.

As a result of the improved algorithm for range-efficient F_0, we obtain improved space and time bounds for dominance norms and distinct summation.%newpage

Dominance norms. Using a reduction to range-efficient F_0, which is elaborated on in section 5, we derive an (ϵ, δ)-approximation algorithm for the max-dominance norm with the following performance guarantees:

- the workspace in bits is $O((\log m + \log n)\frac{1}{\epsilon^2}\log\frac{1}{\delta})$;
- the amortized processing time per item is $O(\log\frac{a}{\epsilon}\log\frac{1}{\delta})$, where a is the value of the item being processed;
- the worst-case processing time per item is $O((\log\log n \log a)\frac{1}{\epsilon^2}\log\frac{1}{\delta})$.

In prior work, Cormode and Muthukrishnan [CM03] gave an (ϵ, δ)-approximation algorithm for the max-dominance norm. Their algorithm uses space $O((\log n + l\frac{1}{\epsilon}\log m \log\log m)\frac{1}{\epsilon^2}\log\frac{1}{\delta})$ and spends $O((\log a \log m)\frac{1}{\epsilon^4}\log\frac{1}{\delta})$ time to process each item, where a is the value of the current element. Our algorithm performs better spacewise and significantly better timewise.

Distinct summation and data aggregation in sensor networks. Our algorithm also provides improved space and time bounds for the distinct summation problem, through a reduction to the range-efficient F_0 problem. Given a multiset of items $M = \{x_1, x_2, \dots\}$, where each x_i is a tuple (k_i, c_i), where $k_i \in [1, m]$ is the "type" and $c_i \in [1, n]$ is the "value." The goal is to compute $S = \sum_{distinct(k_i,c_i)\in M} c_i$. In distinct summation, it is assumed that a particular type is always associated with the same value. However, the same (type, value) pair can appear multiple times in the stream. This reflects the setup used in sensor data aggregation, where each sensor observation has an identifier and a value, and the same observation may be transmitted to the sensor data "sink" through multiple paths. The sink should compute aggregates such as the sum and average over only distinct observations.

We provide an (ϵ, δ)-approximation for the distinct summation problem with the following guarantees:

- the amortized processing time per item is $O(\log\frac{1}{\delta}\log\frac{n}{\epsilon})$;
- the time to answer a query is $O(1)$;
- the workspace used is $O((\log m + \log n)\frac{1}{\epsilon^2}\log\frac{1}{\delta})$ bits.

Considine et al. [CLKB04] and Nath et al. [NGSA04] present alternative algorithms for the distinct summation problem. Their algorithms are based on the algorithm of Flajolet and Martin [FM85] and assume the presence of an ideal hash function, which produces completely independent random outputs on different inputs. However, their analysis does not consider the space required to store such a hash function. Moreover, it is known that hash functions that generate completely independent random values on different inputs cannot be stored in limited space.

Our algorithm does not make an assumption about the availability of ideal hash functions. Instead, we use a hash function which can be stored using limited space but generates only pairwise-independent random numbers. We note that Alon, Matias, and Szegedy [AMS99] provide a way to replace the hash functions used in the Flajolet–Martin algorithm [FM85] by pairwise-independent hash functions. However, the F_0 algorithm by Alon, Matias, and Szegedy [AMS99] is not range-efficient. If the hash functions introduced by Alon, Matias, and Szegedy [AMS99] were used in the algorithms in [CLKB04, NGSA04], there still is a need for a range sampling function similar to the one we present in this paper.

Counting triangles in a data stream. As described above, the problem of counting triangles in a graph that arrives as a stream of edges can be reduced to range-efficient computation of F_0 and F_2 on a stream of integers. Our range-efficient

F_0 algorithm provides a faster solution to one of the components of the problem. However, due to the high complexity of the range-efficient algorithm for F_2, this is not sufficient to obtain an overall reduction in the runtime of the algorithm for counting triangles in graphs. There is recent progress on the problem of counting triangles by Buriol et al. [BFL$^+$06] (see also Jowhari and Ghodsi [JG05]), who give an algorithm through direct random sampling, without using the reduction to F_0 and F_2. The algorithm by Buriol et al. is currently the most efficient solution to this problem and has a space complexity proportional to the ratio between the number of length 2 paths in the graph and the number of triangles in the graph and an expected update time $O(\log |V| \cdot (1 + s \cdot |V|/|E|))$, where s is the space requirement and V and E are the vertices and the edges of the graph, respectively.

1.3. Related work. Estimating F_0 of a data stream is a very well studied problem, because of its importance in various database and networking applications. It is well known that computing F_0 exactly requires space linear in the number of distinct values, in the worst case. Flajolet and Martin [FM85] gave a randomized algorithm for estimating F_0 using $O(\log n)$ bits. This assumed the existence of certain ideal hash functions, which we do not know how to store in small space. Alon, Matias, and Szegedy [AMS99] describe a simple algorithm for estimating F_0 to within a constant relative error which worked with pairwise-independent hash functions and also gave many important algorithms for estimating other frequency moments $F_k, k > 1$ of a data set.

Gibbons and Tirthapura [GT01] and Bar-Yossef et al. [BYJK$^+$02] present algorithms for estimating F_0 to within arbitrary relative error. Neither of these algorithms is range-efficient. The algorithm by Gibbons and Tirthapura used random sampling; its space complexity was $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \log n)$, and processing time per element was $O(\log \frac{1}{\delta})$. The space complexity of this algorithm was improved by Bar-Yossef et al. [BYJK$^+$02], who gave an algorithm with better space bounds, which are the order of the sum of $O(\log n)$ and $poly(\frac{1}{\epsilon})$ terms rather than their product, but at the cost of increased processing time per element.

Indyk and Woodruff [IW03] present a lower bound of $\Omega(\log n + \frac{1}{\epsilon^2})$ for the space complexity of approximating F_0, thus narrowing the gap between known upper and lower bounds for the space complexity of F_0. Since the F_0 problem is a special case of the range-efficient F_0 problem, these lower bounds apply to range-efficient F_0, too. Thus our space bounds of $O((\log m + \log n)\frac{1}{\epsilon^2} \log \frac{1}{\delta})$ compare favorably with this lower bound, showing that our algorithms are near optimal spacewise. However an important metric for range-efficient F_0 is the processing time per item, for which no useful lower bounds are known.

The work of Feigenbaum et al. [FKSV02] on estimating the L^1-difference between streams also has the idea of reductions between data stream algorithms and uses a range-efficient algorithm in their reduction. Bar-Yossef, Kumar, and Sivakumar [BYKS02] mention that their notion of reductions and list-efficiency were inspired by the above-mentioned work of Feigenbaum et al. The algorithm for the L^1-difference [FKSV02] is not based on random sampling but relies on quickly summing many random variables. They develop limited independence random variables which are *range summable*, while our sampling-based algorithm makes use of a *range sampling* technique. Further work on range summable random variables includes Gilbert et al. [GKMS03], Reingold and Naor (for a description see [GGI$^+$02]), and Calderbank et al. [CGL$^+$05].

There is much other interesting work on data stream algorithms. For an overview, the reader is referred to the excellent surveys in [Mut05, BBD$^+$02].

Organization of this paper. The rest of the paper is organized as follows: Section 2 gives a high level overview of the algorithm for range-efficient F_0. Section 3 gives the range sampling algorithm, its proof, and analysis of complexity. Section 4 presents the algorithm for computing F_0 using the range sampling algorithm as the subroutine. Section 5 gives extensions of the basic algorithm to distributed streams and the computation of dominance norms.

2. A high level overview.

Our algorithm is based on random sampling. A random sample of the distinct elements of the data stream is maintained at an appropriate probability, and this sample is used in finally estimating the number of distinct elements in the stream. A key technical ingredient is a novel *range sampling* algorithm, which quickly computes the number of integers in a range $[x, y]$ which belong to the current random sample. The other technique needed to make random sampling work here is *adaptive sampling* (see Gibbons and Tirthapura [GT01]), where the sampling probability is decreased every time the sample overflows. The range sampling algorithm, when combined with adaptive random sampling, yields the algorithm for range-efficient estimation of F_0.

2.1. A random sampling algorithm for F_0.

At a high level, our random sampling algorithm for computing F_0 follows a similar structure to the algorithm by Gibbons and Tirthapura [GT01]. We first recall the main idea of their random sampling algorithm. The algorithm keeps a random sample of all the distinct elements seen so far. It samples each element of the stream with a probability P, while making sure that the sample has no duplicates. Finally, when an estimate is asked for F_0, it returns the size of the sample, multiplied by $1/P$. However, the value of P cannot be decided in advance. If P is large, then the sample might become too big if F_0 is large. On the other hand, if P is too small, then the approximation error might be very high if the value of F_0 was small. Thus, the algorithm starts off with a high value of $P = 1$ and decreases the value of P every time the sample size exceeds a predetermined maximum sample size, α.

The algorithm maintains a current sampling level ℓ which determines a sampling probability P_ℓ. The sampling probability P_ℓ decreases as level ℓ increases. Initially, $\ell = 0$, and ℓ never decreases. The sample at level ℓ, denoted $S(\ell)$, is determined by a hash function $S(\cdot, \ell)$ for $\ell = 0, 1, \ldots, \ell_{max}$, where ℓ_{max} is some maximum level to be specified later.

When item x is presented to the algorithm, if $S(x, \ell) = 1$, then x is stored in the sample $S(\ell)$, and it is not stored otherwise. Each time the size of $S(\ell)$ exceeds α (i.e., an overflow occurs), the current sampling level ℓ is incremented, and an element $x \in S(\ell)$ is placed in $S(\ell + 1)$ only if $S(x, \ell + 1) = 1$. We will always ensure that if $S(x, \ell + 1) = 1$, then $S(x, \ell) = 1$. Thus $S(\ell + 1)$ is a subsample of $S(\ell)$. Finally, anytime an estimate of the number of distinct elements is asked for, the algorithm returns $\frac{|S(\ell)|}{P_\ell}$, where ℓ is the current sampling level. We call the algorithm described so far as the *single-item* algorithm.

2.2. Range efficiency.

When each element of the stream is an interval of items rather than a single item, the main technical problem is to quickly figure out how many points in this interval belong in the sample. More precisely, if ℓ is the current sampling level and an interval $r_i = [x_i, y_i]$ arrives, it is necessary to know the size of the set $\{x \in r_i | S(x, \ell) = 1\}$. We call this the *range sampling* problem.

A naive solution to range sampling is to consider each element $x \in r_i$ individually and check if $S(x, \ell) = 1$, but the processing time per item would be $\Theta(y_i - x_i)$, which could be as much as $\Theta(n)$. We show how to reduce the time per interval significantly, to $O(\log(y_i - x_i))$ operations.

The range-efficient algorithm simulates the single-item algorithm as follows: When an interval $r_i = [x_i, y_i]$ arrives, the size of the set $\{x \in r_i | S(x, \ell) = 1\}$ is computed. If the size is greater than zero, then r_i is added to the sample; otherwise, r_i is discarded. If the number of intervals in the sample becomes too large, then the sampling probability is decreased by increasing the sampling level from ℓ to $\ell + 1$. When such an increase in sampling level occurs, every interval r in the sample such that $\{x \in r | S(x, \ell + 1) = 1\}$ is empty is discarded.

Finally, when an estimate for F_0 is asked for, suppose the current sample is the set of intervals $S = \{r_1, \ldots, r_k\}$ and ℓ is the current sampling level. The algorithm returns $|T|/P_\ell$, where $T = \{x \in r_i \mid r_i \in S, S(x, \ell) = 1\}$.

2.2.1. Hash functions. Since the choice of the hash function is crucial for the range sampling algorithm, we first precisely define the hash functions used for sampling. We use a standard 2-universal family of hash functions [CW79]. First choose a prime number p between $10n$ and $20n$, and then choose two numbers a, b at random from $\{0, \ldots, p - 1\}$. Define hash function $h : \{1, \ldots, n\} \to \{0, \ldots, p - 1\}$ as $h(x) = (a \cdot x + b) \mod p$.

The two properties of h that are important to the F_0 algorithm are as follows:

1. For any $x \in \{1, \ldots, n\}$, $h(x)$ is uniformly distributed in $\{0, \ldots, p - 1\}$.
2. The mapping is pairwise independent.
   For $x_1 \neq x_2$ and $y_1, y_2 \in \{0, \ldots, p - 1\}$,
   $\Pr[(h(x_1) = y_1) \wedge (h(x_2) = y_2)] = \Pr[h(x_1) = y_1] \cdot \Pr[h(x_2) = y_2]$.

For each level $\ell = 0, \ldots, \lfloor \log n \rfloor$, we define the following:

- Region $R_\ell \subset \{0, \ldots, p - 1\}$:

$$R_\ell = \left\{0, \ldots, \left\lfloor \frac{p}{2^\ell} \right\rfloor - 1\right\}.$$

- For every $x \in [1, n]$, the sampling function $S(x, \ell)$ is defined as
  $S(x, \ell) = 1$ if $h(x) \in R_\ell$ and $S(x, \ell) = 0$ otherwise.
- The sampling probability at level ℓ is denoted by P_ℓ. For all $x_1, x_2 \in [1, n]$, $\Pr[S(x_1, \ell) = 1] = \Pr[S(x_2, \ell) = 1]$, and we denote this probability by P_ℓ. Since $h(x)$ is uniformly distributed in $\{0, \ldots, p - 1\}$, we have $P_\ell = |R_\ell|/p$.

2.2.2. Range sampling. During range sampling, the computation of the number $\{x \in r_i | S(x, \ell) = 1\}$ reduces to the following problem: Given numbers a, b, and p such that $0 \leq a, b < p$, function $f : [1, n] \to [0, p - 1]$ is defined as $f(x) = (a \cdot x + b) \mod p$. *Given intervals* $[x_1, x_2] \subset [1, n]$ *and* $[0, q] \subset [0, p - 1]$, *quickly compute the size of the set* $\{x \in [x_1, x_2] | f(x) \in [0, q]\}$.

Note that n could be a large number, since it is the size of the universe of integers in the stream. We present an efficient solution to the above problem. Our solution is a recursive procedure which works by reducing the above problem to another range sampling problem but over a significantly smaller range, whose length is less than half the length of the original range $[x_1, x_2]$. Proceeding thus, we get an algorithm whose time complexity is $O(\log(x_2 - x_1))$. Our range sampling algorithm might be of independent interest and may be useful in the design of other range-efficient algorithms for the data stream model.

The algorithms for computing F_0 found in [AMS99, GT01, BYJK$^+$02] use hash functions defined over $GF(2^m)$, which are different from the ones that we use. Bar-

Yossef et al. [BYKS02], in their range-efficient F_0 algorithm, use the Toeplitz family of hash functions [Gol97], and their algorithm uses specific properties of those hash functions.

3. Range sampling algorithm. In this section, we present a solution to the range sampling problem, its correctness, and time and space complexities. The algorithm *RangeSample*$(r_i = [x_i, y_i], \ell)$ computes the number of points $x \in [x_i, y_i]$ for which $h(x) \in R_\ell$, where $h(x) = (ax + b) \mod p$. This algorithm is later used as a subroutine by the range-efficient algorithm for F_0, which is presented in the next section:

$$\mathit{RangeSample}([x_i, y_i], \ell) = \left|\left\{x \in [x_i, y_i] | h(x) \in \left[0, \left\lfloor \frac{p}{2^\ell} \right\rfloor - 1\right]\right\}\right|.$$

Note that the sequence $h(x_i), h(x_i + 1), \ldots, h(y_i)$ is an arithmetic progression over $\mathbb{Z}_p$ ($\mathbb{Z}_k$ is the ring of integers modulo k) with a common difference a. Thus, we consider the following general problem.

Problem 1. *Let $M > 0$, $0 \leq d < M$, and $0 < L \leq M$. Let $S = \langle u = x_1, x_2, \ldots, x_n \rangle$ be an arithmetic progression over $\mathbb{Z}_M$ with common difference d, i.e, $x_i = (x_{i-1} + d) \mod M$. Let R be a region of the form $[0, L-1]$ or $[-L+1, 0]$. Compute $|S \cap R|$.*

We first informally describe the idea behind our algorithm for Problem 1 and then go on to the formal description. For this informal description, we consider only the case $R = [0, L-1]$.

We first define a total order among the elements of $\mathbb{Z}_m$. Observe that an element of $\mathbb{Z}_m$ has infinitely many representations. For example, if $x \in \mathbb{Z}_m$, then $x, x+m, x+2m, \ldots$ are possible representations of x. For $x \in Z_m$, the standard representation of x is the smallest nonnegative integer $\mathit{std}(x)$ such that $x \equiv \mathit{std}(x)$ in $\mathbb{Z}_m$. If x and y are two elements of $\mathbb{Z}_m$, then we say $x < y$ if $\mathit{std}(x)$ is less than $\mathit{std}(y)$ in the normal ordering of integers. We say $x > y$ if $\mathit{std}(x) \neq \mathit{std}(y)$ and $x \not< y$. In the rest of the paper, we use the standard representation for elements over $\mathbb{Z}_m$. Given an element $x \in \mathbb{Z}_m$, we define $-x$ as $m - x$.

3.1. Intuition. Divide S into subsequences $S_0, S_1, \ldots, S_k, S_{k+1}$ as follows: $S_0 = \langle x_1, x_2, \ldots, x_i \rangle$, where i is the smallest natural number such that $x_i > x_{i+1}$. The subsequences $S_j, j > 0$ are defined inductively. If $S_{j-1} = \langle x_t, x_{t+1}, \ldots, x_m \rangle$, then $S_j = \langle x_{m+1}, x_{m+2}, \ldots, x_r \rangle$, where r is the smallest number such that $r > m+1$ and $x_r > x_{r+1}$; if no such r exists, then $x_r = x_n$. Note that if $S_j = \langle x_t, x_{t+1}, \ldots, x_m \rangle$ then $x_t < d$ and $x_t, x_{t+1}, \ldots, x_m$ are in ascending order. Let f_j denote the first element of S_j, and let e_j denote the last element of S_j.

We treat the computation of $|S_0 \cap R|$ and $|S_{k+1} \cap R|$ as special cases and now focus on the the computation of $|S_i \cap R|$ for $1 \leq i \leq k$. Let $L = d \times q + r$, where $r < d$. Note that the values of q and r are unique. We observe that, for each $i \in [1, k]$, it is easy to compute the number of points of S_i that lie in R. More precisely we make the following observation.

Observation 1. For every $i \in [1, k]$, if $f_i < r$, then $|S_i \cap R| = \lfloor \frac{L}{d} \rfloor + 1$, else $|S_i \cap R| = \lfloor \frac{L}{d} \rfloor$.

Thus Problem 1 is reduced to computing the size of the following set:

$$\{i \mid 1 \leq i \leq k, f_i \in [0, r-1]\}.$$

The following observation is critical.

Observation 2. The sequence $\langle f_1, f_2, \dots, f_k \rangle$ forms an arithmetic progression over $\mathbb{Z}_d$.

We show in Lemma 2 that the common difference of this sequence is $d - r'$, where $r' = M \bmod d$. Thus, we have reduced the original problem (Problem 1) with a common difference of d to a smaller problem, whose common difference is $d - r'$. However, this reduction may not always be useful since $d - r'$ may be not be much smaller than d. However, we now show that it is always possible to get a reduction to a subproblem with a significantly smaller common difference.

We can always view any arithmetic progression over $\mathbb{Z}_d$ with common difference $d - r'$ as an arithmetic progression with common difference $-r'$. The next crucial observation is as follows.

Observation 3. At least one of $d - r'$ or r' is less than or equal to $d/2$, and we can choose to work with the smaller of $d - r'$ or r'.

Thus, we have reduced Problem 1 to a smaller problem, whose common difference is at most half the common difference of Problem 1. Proceeding thus, we get a recursive algorithm whose time complexity is logarithmic in d.

3.2. Formal description. We start with the following useful lemmas.

LEMMA 1. *If $R = [0, L-1]$, then for $1 \le i \le k$,*

$$|S_i \cap R| = \begin{cases} \lfloor \frac{L}{d} \rfloor + 1 & \text{if } f_i \in [0, r-1], \\ \lfloor \frac{L}{d} \rfloor & \text{if } f_i \notin [0, r-1]. \end{cases}$$

If $R = [-L+1, 0]$, then

$$|S_i \cap R| = \begin{cases} \lfloor \frac{L}{d} \rfloor + 1 & \text{if } e_i \in [-r+1, 0], \\ \lfloor \frac{L}{d} \rfloor & \text{if } e_i \notin [-r+1, 0]. \end{cases}$$

Note that each f_i is less than d, and so we can view the f_i's as elements over $\mathbb{Z}_d$. Below we show that the f_i's form an arithmetic progression over $\mathbb{Z}_d$.

LEMMA 2. *Let $M = d \times q' + r'$, where $r' < d$. Then, for $1 \le i < k$, $f_{i+1} = (f_i - r')$ mod d.*

Proof. Recall that $S_i = \langle f_i, f_i + d, \dots, e_i \rangle$ and $M - d \le e_i \le M - 1$. We have two cases.

If $f_i < r'$, then $e_i = f_i + q' \times d$. Thus

$$\begin{aligned} f_{i+1} &= (e_i + d) \mod M \\ &= (f_i + q' \times d + r' + d - r') \mod M \\ &= (f_i + M + (d - r')) \mod M \\ &= (d - (r' - f_i)) \mod M. \end{aligned}$$

Since f_{i+1} is less than d, the final expression for f_{i+1} can be written in $\mathbb{Z}_d$ as follows: $f_{i+1} = (f_i + (d - r')) \mod d = (f_i - r') \mod d$.

In the second case, if $f_i \ge r'$, then $e_i = (f_i + (q' - 1) \times d) \mod M$. Thus, $f_{i+1} = (f_i + q' \times d) \mod M = (f_i - r') \mod M$.

Since f_{i+1} is less than d, the above can be written in $\mathbb{Z}_d$ as $f_{i+1} = (f_i - r')$ mod d. □

Note that similar to the f_i's, the e_i's are also restricted to having a value in a range of length d, since $M - 1 \ge e_i \ge M - d$. However, the e_i's do not directly form an arithmetic progression over Z_d. Thus, we define a function $map : \{M - d, M - d + 1, \dots, M - 1\} \to \mathbb{Z}_d$ such that $map(e_i)$'s form an arithmetic progression over $\mathbb{Z}_d$.

Definition 2. *For $M - d \le x < M$, $map(x) = M - x - 1$.*

We now present the following lemma for $map(e_i)$'s, which is similar to Lemma 2. We omit the proof since it is similar to the proof of Lemma 2.

Lemma 3. *Let $M = d \times q' + r'$, where $r' < d$. Then, for $1 < i \le k$, $map(e_i) = (map(e_{i-1}) + r') \mod d$.*

Next we argue that, for our purposes, any arithmetic progression with common difference $-r'$ can be viewed as a different arithmetic progression with common difference r'. Let $T = \langle y_1, y_2, \ldots, y_k \rangle$ be an arithmetic progression over $\mathbb{Z}_t$ with a common difference $-s$, i.e.,

$$y_i = (y_{i-1} - s) \mod t.$$

Let R be of the form $[0, L-1]$ or $[-L+1, 0]$. Define R' as follows: if $R = [0, L-1]$, then $R' = [-L+1, 0]$, else $R' = [0, L-1]$.

Lemma 4. *Let $T' = \langle y'_1, y'_2, \ldots, y'_k \rangle$ be an arithmetic progression over $\mathbb{Z}_t$ defined as $y'_1 = -y_1$, and for $1 < i < k$, $y'_i = (y'_{i-1} + s) \mod t$. Then,*

$$|T \cap R| = |T' \cap R'|.$$

Proof. If $y_1 - ks = x \mod t$, then $-y_1 + ks = -x \mod t$. Thus $x \in R$ if and only if $-x \in R'$. □

3.2.1. Description of the algorithm. Now we describe our algorithm for Problem 1.

Procedure Hits(M, d, u, n, R).

Precondition: R is of the form $[0, L-1]$ or $[-L+1, 0]$, where $L \le M$, and $u < M, d < M$.

Goal: Compute $|S \cap R|$, where

$$S = \langle u, (u+d) \mod M, \ldots, (u + n \times d) \mod M \rangle.$$

```
1. (a) If $n = 0$, then Return 1 if $u \in R$, else Return 0.
   (b) Let $S = S_0, S_1, \ldots, S_k, S_{k+1}$. Compute $k$.
   (c) $\mathrm{Hits}_0 \leftarrow |S_0 \cap R|$. $\mathrm{Hits}_{k+1} \leftarrow |S_{k+1} \cap R|$.
   (d) If $d = 1$, then
         Return $\mathrm{Hits}_0 + \mathrm{Hits}_{k+1} + (L \times k)$.
2. Compute $r$ and $r'$ such that $L = d \times q + r$, $r < d$ and $M = d \times q' + r'$ and
   $r' < d$.
3. If $R = [0, L-1]$, then
  3.1. Compute $f_1$, where $f_1$ is the first element of $S_1$.
  3.2. $u_{new} \leftarrow f_1$, $M_{new} \leftarrow d, n_{new} \leftarrow k$.
  3.3. If $d - r' \le d/2$, then $R_{new} \leftarrow [0, r-1]$ and $d_{new} \leftarrow d - r'$.
       Comment: In this case, we view the $f_i$'s as arithmetic progression over
       $\mathbb{Z}_d$ with common difference $d - r'$.
  3.4. High = Hits($M_{new}$, $d_{new}$, $u_{new}$, $n_{new}$, $R_{new}$).
       Comment: By making a recursive call to Hits, we are computing High,
       the cardinality of the set $\{i \mid f_i \in [0, r-1], 1 \le i \le k\}$.
  3.5. If $d - r' > d/2$ (so $r' \le d/2$), then $u_{new} \leftarrow -f_1$, $d_{new} \leftarrow r'$, and
       $R_{new} \leftarrow [-r+1, 0]$.
       Comment: In this case we consider the $f_i$'s as an arithmetic progression
       over $\mathbb{Z}_d$ with common difference $-r'$.
  3.6. High = Hits($M_{new}$, $d_{new}$, $u_{new}$, $n_{new}$, $R_{new}$).
```

```
    3.7. Low ← (k − High). Return
```

$$\text{Hits}_0 + \text{Hits}_{k+1} + \left(\text{High} \cdot \left(\left\lfloor \frac{L}{d} \right\rfloor + 1 \right) \right) + \left(\text{Low} \cdot \left\lfloor \frac{L}{d} \right\rfloor \right).$$

```
4. If $R = [-L+1, 0]$ then
    4.1. Compute $e_1$ and $map(e_1)$, where $e_1$ is the last element of $S_1$.
    4.2. $u_{new} \leftarrow map(e_1), M_{new} \leftarrow d, n_{new} \leftarrow k$.
    4.3. If $r' \leq d/2$, then $R_{new} \leftarrow [0, r-1]$, and $d_{new} \leftarrow r'$.
         Comment: In this case we view $map(e_i)$'s as an arithmetic progression
            over $\mathbb{Z}_d$ with common difference $r'$.
    4.4. High = Hits($M_{new}$, $d_{new}$, $u_{new}$, $n_{new}$, $R_{new}$).
    4.5. If $r' > d/2$ (so $d - r' \leq d/2$), then $u_{new} \leftarrow -map(e_1), d_{new} \leftarrow d - r'$,
         and $R_{new} \leftarrow [-r+1, 0]$.
         Comment: In this case we view $map(e_i)$'s as an arithmetic progression
            over $\mathbb{Z}_d$ with common difference $-r'$.
    4.7. High = Hits($M_{new}$, $d_{new}$, $u_{new}$, $n_{new}$, $R_{new}$).
    4.8. Low ← (k − High). Return
```

$$\text{Hits}_0 + \text{Hits}_{k+1} + \left(\text{High} \cdot \left(\left\lfloor \frac{L}{d} \right\rfloor + 1 \right) \right) + \left(\text{Low} \cdot \left\lfloor \frac{L}{d} \right\rfloor \right).$$

3.3. Correctness of Hits.

THEOREM 1. *Given M, d, u, n, R, Algorithm Hits correctly computes the solution to Problem* 1.

Proof. We first consider the case $R = [0, L-1]$. It is easy to verify that when $d = 1$ or when $n = 0$ the algorithm correctly computes the answer. Note that

$$|S \cap R| = |S_0 \cap R| + |S_{k+1} \cap R| + \sum_{i=1}^{i=k} |S_i \cap R|.$$

Step 1 correctly computes $|S_0 \cap R|$ and $|S_{k+1} \cap R|$. By Lemma 1, for $1 \leq i \leq k$, $|S_i \cap R|$ is $\lfloor \frac{L}{d} \rfloor + 1$ if $f_i \in [0, r-1]$ and is $\lfloor \frac{L}{d} \rfloor$ if $f_i \notin [0, r-1]$. Let High denote the number of f_i's for which $f_i \in [0, r-1]$. Given the correct value of High, the algorithm correctly computes the final answer in step 3.4.

Thus the goal is to show that the algorithm correctly computes the value of High. Let $T = \langle f_1, \ldots, f_k \rangle$. At this point, the algorithm considers two cases.

If $d - r' \leq d/2$, then the algorithm is required to compute the number of elements in T that lie in $[0, r-1]$. By Lemma 2, T is an arithmetic progression over $\mathbb{Z}_d$, with common difference $d - r'$. The starting point of this progression is f_1; the number of elements in the progression is k. Thus the algorithm makes a correct recursive call to Hits.

If $d - r' > d/2$, then by Lemma 2 T is an arithmetic progression over $\mathbb{Z}_d$ with the common difference $-r'$. The algorithm is required to compute $|T \cap [0, r-1]|$.

Define a new sequence $T' = \langle f'_1, f'_2, \ldots, f'_k \rangle$ over $\mathbb{Z}_d$ as follows: $f'_1 = -f_1$ and $f'_{i+1} = (f'_i + r') \mod d, 1 \leq i \leq k-1$. By Lemma 4,

$$|T \cap [0, r-1]| = |T' \cap [-r+1, 0]|.$$

Thus the algorithm makes the correct recursive call in step 3.4 to compute $|T' \cap [-r+1, 0]|$ which equals $|T \cap [0, r-1]|$.

Thus the algorithm correctly computes $S \cap R$ when R is of the form $[0, L-1]$. Correctness for the case when R is of the form $[-L+1, 0]$ follows similarly. ∎

3.4. Complexity of Hits. For the time complexity, we assume that all arithmetic operations in Z_k, including addition, multiplication, and division, take unit time.

THEOREM 2. *The time complexity of* Hits(M,d,u,n,R) *is* $O(\min\{\log d, \log n\})$, *and the space complexity is* $O(\log M + \log n)$.

Proof. Time complexity: It is clear that steps 1 and 2 can be completed using a constant number of operations. The algorithm makes a recursive call in step 3 or 4. Also, it is clear that $d_{new} \leq d/2$. Thus if we parametrize the running time of the algorithm with d, then

$$\begin{aligned} T(d) &= T(d_{new}) + O(1) \\ &\leq T(d/2) + O(1) = O(\log d). \end{aligned}$$

We can get a better bound on the running time as follows: The input parameters to the recursive procedure are M, d, u, n, R. When the recursive procedure is called, the parameters are $M_{new} = d$, $d_{new} \leq d/2$, $n_{new} \leq \lceil n \cdot d/M \rceil$.

In every recursive call except (perhaps) the first, it must be true that $d \leq M/2$. Thus, in every recursive call except for the first, $n_{new} \leq n/2$. If we parametrize the running time on n, then the total time of the algorithm is $O(\log(n))$.

Space complexity: Whenever the Hits algorithm makes a recursive call, it needs to store values of a constant number of local variables such as Hits$_0$, Hits$_{k+1}$, n, etc. Since M dominates u, d, and L, each time the algorithm makes a recursive call, it needs $O(\log n + \log M)$ of stack space. Since the depth of the recursion is no more than $\log n$, the total space needed is $O(\log n \cdot (\log n + \log M))$. We can further reduce the space by a careful implementation of the recursive procedure as follows.

In general Hits$(p_1, p_2, \ldots, p_5) = \beta + \gamma$ Hits$(p'_1, p'_2, \ldots, p'_5)$, where β and γ are functions of $p_1, \ldots, p_5$. This is a tail-recursive procedure, which can be implemented without having to allocate space for a new stack for every recursive call and without having to tear down the stack upon a return. Thus, the total space can be reduced to $O(\log n + \log M)$. ∎

4. Algorithm for range-efficient F_0.

We now describe the complete algorithm for estimating F_0, using the range sampling algorithm as a subroutine. We then present its correctness and time and space complexities.

From section 2, recall the following notation: The input stream is $r_1, r_2, \ldots, r_m$, where each stream element $r_i = [x_i, y_i] \subset [1, n]$ is an interval of integers $x_i, x_i + 1, \ldots, y_i$. Integer p is a prime number between $10n$ and $20n$. For each level $\ell = 0, \ldots, \lceil \log p \rceil$, the following hold:

1. $R_\ell = \{0, \ldots, \lfloor \frac{p}{2^\ell} \rfloor - 1\}$.
2. For every $x \in [1, n]$, the sampling function at level ℓ, $S(x, \ell)$, is defined as $S(x, \ell) = 1$ if $h(x) = 1$ and $S(x, \ell) = 0$ otherwise.
3. The sampling probability at level ℓ is $P_\ell = |R_\ell|/p$.

4.1. Algorithm description. A formal description of the algorithm appears in Figure 1. The algorithm does not directly yield an (ϵ, δ)-estimator for F_0 but instead gives an estimator which is within a factor of ϵ of F_0 with a constant probability. Finally, by taking the median $O(\log \frac{1}{\delta})$ of such estimators, we get an (ϵ, δ)-estimator.

The random sample. The algorithm maintains a sample S of the intervals seen so far. S is initially empty. The maximum size of S is $\alpha = \frac{60}{\epsilon^2}$ intervals. The algorithm

```
Initialization.
    1. Choose a prime number $p$ such that $10n \leq p \leq 20n$, and choose two numbers
       $a, b$ at random from $\{0, \dots, p-1\}$.
    2. $S \leftarrow \phi$.
    3. $\ell \leftarrow 0$.
When a new interval $r_i = [x_i, y_i]$ arrives:
    1. If $r_i$ intersects with any interval in $S$, then
        (a) While there is an interval $r \in S$ such that $r \cap r_i \neq \phi$,
            i. $S \leftarrow S - r$;
            ii. $r_i \leftarrow r_i \cup r$.
        (b) $S \leftarrow S \cup r_i$.
    2. Else If $RangeSample(r_i, \ell) > 0$, then
        (a) $S \leftarrow S \cup \{r_i\}$ // insert into sample;
        (b) While $(|S| > \alpha)$ // overflow,
            i. $\ell \leftarrow \ell + 1$.
               If $\ell > \lceil \log p \rceil$, then // maximum level
                                          // reached, algorithm fails
                   return;
            ii. $S \leftarrow \{r \in S | RangeSample(r, \ell) > 0\}$.

When an estimate for $F_0$ is asked for: Return $\sum_{r \in S} RangeSample(r, \ell)/P_\ell$
```

FIG. 1. *The range-efficient F_0 algorithm using the range sampling algorithm as a subroutine.*

also has a current sampling level ℓ, which is initialized to 0. We maintain the following invariants for the random sample S.

Invariant 1. All the intervals in S are disjoint.

Invariant 2. Every interval in S has at least one element selected into the sample at the current sampling level.

When a new interval $r_i = [x_i, y_i]$ arrives. The algorithm first checks if r_i intersects with any currently existing interval in S. If so, it deletes all the intersecting intervals from S, merges all of them with r_i to form a single interval, and inserts the resulting interval into S. If r_i does not intersect with any currently existing interval, it calls the range sampling subroutine to determine if *any* point in $[x_i, y_i]$ belongs in S at level ℓ. If yes, then the whole interval is stored in the sample.

Overflow. It is possible that after adding a new element to the sample, the size of S exceeded α. In such a case, the algorithm increases its sampling level and subsamples the elements of the current sample into the new level. This subsampling is repeated until either the size of the sample becomes less than α or the maximum sampling level is reached (i.e., $\ell = \lceil \log p \rceil$).

Estimating F_0. When an estimate is asked for, the algorithm uses the range sampling routine to determine the size of the current sample and returns this value boosted by $1/P_\ell$, where ℓ is the current sampling level.

4.2. Proof of correctness. The proof follows a parallel structure to the proof by Gibbons and Tirthapura [GT01]. We give the complete correctness proof here because the hash functions used here are different from those in [GT01] and also for the sake of completeness.

FACT 1. *For any $\ell \in [0, \dots, \lceil \log p \rceil]$, $1/2^{\ell+1} \leq P_\ell \leq 1/2^\ell$.*

Fact 2. *For any $\ell \in [0, \ldots, \lceil \log p \rceil]$, the random variables $\{S(x, \ell) | x \in [1, n]\}$ are all pairwise independent.*

Lemma 5. *Invariants* 1 *and* 2 *are true before and after the processing of each interval in the stream.*

Proof. We prove by induction on the number of intervals that have been processed. The base case is clear, since S is initialized to ϕ. Suppose a new interval r arrives. If r intersects with any interval already in S, then our algorithm clearly maintains Invariant 1, and it can be verified that Invariant 2 also holds. If r does not intersect with any element already in S, then it is included in the sample if and only if r has at least one element which would be sampled at the current sampling level. This ensures that Invariant 2 is maintained. It can be easily verified that the steps taken to handle an overflow also maintain the invariants. □

Let random variable Z denote the result of the algorithm. We analyze the algorithm by looking at the following hypothetical process. This process is useful for us only to visualize the proof and is not executed by the algorithm. The stream of intervals R is "expanded" to form the stream I of the constituent integers. For each interval $[x_i, y_i] \in I$, the integer stream I consists of $x_i, x_i + 1, \ldots, y_i$.

Let $D(I)$ denote the set of all distinct elements in I. We want to estimate $F_0 = |D(I)|$. Each element in $D(I)$ is placed in different levels as follows. All the elements of $D(I)$ are placed in level 0. An element $x \in D(I)$ is placed in every level $\ell > 0$ such that $S(x, \ell) = 1$. For level $\ell = 0 \ldots \lceil \log p \rceil$, let X_ℓ denote the number of distinct elements placed in level ℓ.

Lemma 6. *$E[X_\ell] = F_0 P_\ell$, and $Var[X_\ell] = F_0 P_\ell (1 - P_\ell)$.*

Proof. By definition, $X_\ell = \sum_{x \in D(I)} S(x, \ell)$. Thus, $E[X_\ell] = \sum_{x \in D(I)} E[S(x, \ell)] = |D(I)| P_\ell = F_0 P_\ell$. Because the random variables $\{S(x, \ell) | x \in D(I)\}$ are all pairwise independent (Fact 2), the variance of their sum is the sum of their variances, and the expression for the variance follows. □

Let ℓ^* denote the smallest integer $\ell \geq 0$ such that $X_\ell \leq \alpha$. Let ℓ' denote the level at which the algorithm finally ends.

Lemma 7. *The algorithm returns $Z = X_{\ell'} / P_{\ell'}$ and $\ell' \leq \ell^*$.*

Proof. We first show that $\ell' \leq \ell^*$. If the algorithm never increased its sampling level, then $\ell' = 0$ and thus $\ell' \leq \ell^*$ trivially. Suppose $\ell' \geq 1$. Thus the algorithm must have increased its sampling level from $\ell' - 1$ to ℓ'. The increase in level must have been due to an overflow, and the number of (disjoint) intervals at level $\ell' - 1$ must have been more than α. Since, due to Invariant 2, each interval in the sample at level $\ell' - 1$ has at least one point x such that $S(x, \ell' - 1) = 1$, it must be true that $X_{\ell' - 1} > \alpha$, and similarly it follows that $X_\ell > \alpha$ for any $\ell < \ell'$. However, by definition of ℓ^*, it must be true that $X_{\ell^*} \leq \alpha$. Thus it must be true that $\ell^* \geq \ell'$.

Next, we show that the algorithm returns $X_{\ell'} / P_{\ell'}$. Consider the set $S' = \{x \in D(I) | S(x, \ell') = 1\}$. Consider some element $x \in S'$. Integer x must have arrived as a part of some interval $r \in R$. Either r must be in S (if r did not intersect with any range already in the sample and was not later merged with any other interval), or there must be an interval $r' \in S$ such that $r \subset r'$. In both cases, x is included in S. Further, because of Invariant 1, x will be included in exactly one interval in S and will be counted (exactly once) as a part of the sum $\sum_{r \in S} RangeSample(r, \ell')$. Conversely, an element y such that $S(y, \ell') = 0$ will not be counted in the above sum. Thus, the return value of the algorithm is exactly $|S'| / P_\ell$, which is $X_{\ell'} / P_{\ell'}$. □

Define a level ℓ to be *good* if X_ℓ / P_ℓ is within ϵ relative error of F_0. Otherwise, level ℓ is *bad*. For each $\ell = 0, \ldots, \lceil \log p \rceil$, let B_ℓ denote the event that level ℓ is bad, and let T_ℓ denote the event that the algorithm stops at level ℓ.

Theorem 3.

$$\Pr\{Z \in [(1-\epsilon)F_0, (1+\epsilon)F_0]\} \geq 2/3.$$

Proof. Let P denote the probability that the algorithm fails to produce a good estimate. This happens if any of the following is true:

- The maximum level $max = \lceil \log p \rceil$ is reached.
- The level at which the algorithm stops is a bad level, i.e., only for some $\ell \in \{0, \dots, \lceil \log p \rceil - 1\}$, T_ℓ and B_ℓ are both true.

Thus

$$P = \Pr[T_{max}] + \sum_{\ell=0}^{max-1} \Pr[T_\ell \wedge B_\ell].$$

Let ℓ_m denote the lowest numbered level ℓ such that $E[X_\ell] < \alpha/2$. We prove that ℓ_m exists and $0 \leq \ell_m < max$.

Using Lemma 6, we first note that $E[X_{max}] = P_{max}F_0$. Using Fact 1 and $max = \lceil \log p \rceil$, we get $P_{max} \leq 1/p$. Since $F_0 \leq n$, we get $E[X_{max}] \leq n/p \leq 1/10$; since p is a prime number between $10n$ and $20n$:

(1) $$E[X_{max}] \leq \frac{1}{10}.$$

Since $\alpha = 60/\epsilon^2$, we have $E[X_{max}] < \alpha/2$. Thus ℓ_m exists.

Next, we argue that $\ell_m < max$. By definition of ℓ_m, $E[X_{\ell_m - 1}] = P_{\ell_m - 1}F_0 \geq \alpha/2$. From Fact 1, we know that $P_{\ell_m - 1} \leq 4P_{\ell_m}$. Thus, $E[X_{\ell_m}] = P_{\ell_m}F_0 \geq (1/4)\alpha/2 = 60\epsilon^2/8 > 7$:

(2) $$E[X_{\ell_m}] > 7.$$

From (1) and (2), we have $\ell_m < max$.

We now bound P as

$$\begin{aligned} P &= \Pr[T_{max}] + \sum_{\ell=\ell_m+1}^{max-1} \Pr[T_\ell \wedge B_\ell] + \sum_{\ell=0}^{\ell_m} \Pr[T_\ell \wedge B_\ell] \\ &\leq \Pr[T_{max}] + \sum_{\ell=\ell_m+1}^{max-1} \Pr[T_\ell] + \sum_{\ell=0}^{\ell_m} \Pr[B_\ell] \\ &\leq \sum_{\ell=\ell_m+1}^{max} \Pr[T_\ell] + \sum_{\ell=0}^{\ell_m} \Pr[B_\ell]. \end{aligned}$$

In Lemma 8 we show that $\sum_{\ell=0}^{\ell_m} \Pr[B_\ell] < 16/60$, and in Lemma 9 we show that $\sum_{\ell=\ell_m+1}^{max} \Pr[T_\ell] < 1/30$. Putting them together, the theorem is proved. □

Lemma 8. $\sum_{\ell=0}^{\ell_m} \Pr[B_\ell] < 16/60$.

Proof. Let μ_ℓ and σ_ℓ denote the mean and standard deviation of X_ℓ, respectively. Then,

$$\Pr[B_\ell] = \Pr|X_\ell - \mu_\ell| \geq \epsilon\mu_\ell.$$

Using Chebyshev's inequality, $\Pr|X_\ell - \mu_\ell| \geq t\sigma_\ell \leq \frac{1}{t^2}$ and substituting $t = \frac{\epsilon\mu_\ell}{\sigma_\ell}$, we get

(3) $$\Pr[B_\ell] \leq \frac{\sigma_\ell^2}{\epsilon^2\mu_\ell^2}.$$

Substituting values from Lemma 6 into (3), we get

$$\Pr[B_\ell] \le \frac{1-P_\ell}{\epsilon^2 P_\ell F_0} < \frac{1}{F_0\epsilon^2 P_\ell}.$$

Thus,

$$\sum_{\ell=0}^{\ell_m} \Pr[B_\ell] = \frac{1}{F_0\epsilon^2}\sum_{\ell=0}^{\ell_m}\frac{1}{P_\ell}.$$

From Fact 1, we have $1/P_\ell \le 2^{\ell+1}$. Thus, we have

$$\sum_{\ell=0}^{\ell_m} \Pr[B_\ell] \le \frac{1}{F_0\epsilon^2}\sum_{\ell=0}^{\ell_m} 2^{\ell+1} < \frac{1}{F_0\epsilon^2}2^{\ell_m+2}.$$

By definition, ℓ_m is the lowest numbered level ℓ such that $E[X_\ell] < \alpha/2$. Thus, $F_0 P_{\ell_m-1} = E[X_{\ell_m-1}] \ge \alpha/2$. Using Fact 1, we get $F_0/2^{\ell_m-1} \ge \alpha/2$. It follows that

$$\sum_{\ell=0}^{\ell_m} \Pr[B_\ell] \le \frac{4}{\epsilon^2}\frac{2^{\ell_m}}{F_0} \le \frac{4}{\epsilon^2}\frac{4}{\alpha} < \frac{16}{60}$$

by using $\alpha = \frac{60}{\epsilon^2}$. □

LEMMA 9. $\sum_{\ell=\ell_m+1}^{max} \Pr[T_\ell] < 1/30$.

Proof. Let $P_s = \sum_{\ell=\ell_m+1}^{max} \Pr[T_\ell]$. We see that P_s is the probability that the algorithm stops in level ℓ_m+1 or greater. This implies that at level ℓ_m there were at least α intervals in the sample S. Invariant 2 implies that there were at least α elements sampled at that level, so that $X_{\ell_m} \ge \alpha$.

$$\begin{aligned} P_s &\le \Pr[X_{\ell_m} \ge \alpha] \\ &= \Pr\left[X_{\ell_m} - \mu_{\ell_m} \ge \alpha - \frac{\alpha}{2}\right] \\ &\le \frac{\sigma_{\ell_m}^2}{\alpha^2 (1/2)^2}. \end{aligned}$$

From Lemma 6, we get $\sigma_{\ell_m}^2 < E[X_{\ell_m}] < \alpha/2$. Using this in the above expression, we get

$$P_s \le \frac{2}{\alpha} = \frac{2\epsilon^2}{60} < \frac{1}{30}.$$

The last inequality is obtained by using $\epsilon < 1$. □

4.3. Time and space complexity. In this section we prove bounds on the time and space complexity of the algorithm.

LEMMA 10. *The space complexity of the range-efficient* (ϵ, δ)*-estimator for* F_0 *is* $O(\frac{\log 1/\delta \log n}{\epsilon^2})$.

Proof. The workspace required is the space for the sample S plus the workspace for the range sampling procedure *RangeSample*. Sample S contains α intervals, where each interval can be stored using two integers, thus taking $2\log n$ bits of space. The workspace required by the range sampling procedure is $O(\log n)$ bits, so that the total workspace is $O(\alpha \log n + \log n) = O(\frac{\log n}{\epsilon^2})$. Since the we need to run $O(\log 1/\delta)$ instances of this algorithm, the total space complexity is $O(\frac{\log 1/\delta \log n}{\epsilon^2})$. We note that the space complexity is identical to that of the single-item case, where each data item is a single number instead of a range. □

As noted in section 3, we assume that arithmetic operations, including addition, multiplication, and division, in $\mathbb{Z}_k$ take unit time.

LEMMA 11. *The amortized time taken to process an interval $r = [x, y]$ by the range-efficient (ϵ, δ)-estimator for F_0 is $O(\log (n/\epsilon) \log 1/\delta)$.*

Proof. The time to handle a new interval $r = [x, y]$ consists of three parts:

1. time for checking if r intersects any interval in the sample;
2. time required for range sampling (from Theorem 2 this is $O(\log(y-x))$, which is always $O(\log n)$, and perhaps is much smaller than $\Theta(\log n)$).
3. time for handling an overflow in the sample.

We now analyze the time for the first and third parts.

First part. This is the time to check if the interval intersects any of the $O(1/\epsilon^2)$ intervals already in the sample and to merge them, if necessary. This takes $O(1/\epsilon^2)$ time if done naively. This can be improved to $O(\log (1/\epsilon))$ amortized time as follows: Since all the intervals in S are disjoint, we can define a linear order among them in the natural way. We store S in a balanced binary search tree $T(S)$ augmented with in-order links. Each node of $T(S)$ is an interval, and by following the in-order links, we can get the sorted order of S. When a new interval $r = [x, y]$ arrives, we first search for the node in $T(S)$ which contains x. There are three cases possible:

(a) Interval r does not intersect any interval in S. In this case, r is inserted into the tree, which takes $O(\log (1/\epsilon))$ time.

(b) Interval r intersects some interval in S, and there is an interval $t \in S$ which contains x. In such a case, by following the in-order pointers starting from t, we can find all the intervals that intersect r. All these intervals are merged together with r to form a single new interval, say r'. We delete all the intersecting intervals from $T(S)$, and insert r' into $T(S)$. Since each interval is inserted only once and is deleted at most once, the time spent in finding and deleting each intersecting interval can be charged to the insertion of the interval. Thus, the amortized time for handling r is the time for searching for x plus the time to insert r'. Since $|S| = O(1/\epsilon^2)$, this time is $O(\log (1/\epsilon))$.

(c) Interval r intersects some intervals in S, but none of them contain x. This is similar to case (b).

Third part. This is the time for handling an overflow, subsampling to a lower level. For each change of level, we will have to apply the range sampling subroutine $O(1/\epsilon^2)$ times. Each change of level selects roughly half the the number of points belonging in the previous level into the new level. However, since each interval in the sample may contain many points selected in the current level, it is possible that more than one level change may be required to bring the sample size to less than α intervals.

However, we observe that the total number of level changes (over the whole data stream) is less than $\lceil \log p \rceil$ with high probability, since the algorithm does not reach level $\lceil \log p \rceil$ with high probability. Since $\log p = \Theta(\log n)$, the total time taken by level changes over the whole data stream is $O(\frac{\log^2 n}{\epsilon^2} \log \frac{1}{\delta})$. If the length of the data stream dominates the above expression, then the amortized cost of handling the overflow is $O(1)$.

Thus, the amortized time to handle an interval $r = [x, y]$ per instance of the algorithm is $O(\log(y-x) + \log(1/\epsilon))$. Since there are $O(\log 1/\delta)$ instances, the amortized time per interval is $O(\log((y-x)/\epsilon) \log 1/\delta)$ which is also $O(\log(n/\epsilon) \log 1/\delta)$.

It follows that the worst-case processing time per item is $O(\frac{\log^2 n}{\epsilon^2} \log \frac{1}{\delta})$. If our focus was on optimizing the worst-case processing time per item, then we could reduce the above to $O(\frac{\log\log n \log(y-x)}{\epsilon^2} \log \frac{1}{\delta})$ by changing levels using a binary search rather than sequentially when an overflow occurs. □

LEMMA 12. *The algorithm can process a query for F_0 in $O(1)$ time.*

Proof. We first describe how to process a query for F_0 in $O(\log 1/\delta)$ time. At first glance, it seems necessary to apply the range sampling procedure on α intervals and compute the sum. However, we can do better as follows: With each interval in S, store the number of points in the interval which are sampled at the current sampling level. This is first computed either when the interval was inserted into the sample or when the algorithm changes levels. Further, the algorithm also maintains the current value of the sum $\sum_{r \in S} RangeSample(r, \ell)/P_\ell$, where ℓ is the current level and S is the current sample. This sum is updated every time a new interval is sampled or when the algorithm changes level. The above changes do not affect the asymptotic cost of processing a new item. Given this, an F_0 query can be answered for each instance of the algorithm in constant time. Since it is necessary to compute the median of many instances of the algorithm, the time to answer an F_0 query is $O(\log 1/\delta)$.

Given that $O(\log 1/\delta)$ is asymptotically less than the time required to process a new interval, the algorithm can always update the estimate of F_0 while processing a new interval, without affecting the asymptotic cost of processing an interval. If such an estimate of F_0 is maintained continuously, then a query for F_0 can be answered in $O(1)$ time. □

5. Applications and extensions.

5.1. Dominance norms.

We recall the problem here. Given input $\mathcal{I}$ consisting of k streams of m positive integers each, let $a_{i,j}, i = 1, \ldots k, j = 1 \ldots m$, represent the jth element of the ith stream. The max-dominance norm is defined as $\sum_{j=1}^{m} \max_{1 \le i \le k} a_{i,j}$. We can reduce the max-dominance norm of $\mathcal{I}$ to range-efficient F_0 of a stream $\mathcal{O}$ derived as follows: Let n denote an upper bound on $a_{i,j}$. For each element $a_{i,j} \in \mathcal{I}$, the interval $[(j-1)n, (j-1)n + a_{i,j} - 1]$ is generated in $\mathcal{O}$. It is easy to verify the following fact.

FACT 3. *The max-dominance norm of $\mathcal{I}$ equals the number of distinct elements in $\mathcal{O}$.*

Note that the elements of stream $\mathcal{O}$ can take values in the range $[0, nm-1]$. Using the range-efficient algorithm on $\mathcal{O}$, the space complexity is now $O(1/\epsilon^2(\log m + \log n) \log 1/\delta)$. Since the length of the interval in $\mathcal{O}$ corresponding to $a_{i,j} \in \mathcal{I}$ is $a_{i,j}$, from section 4.3 it follows that the amortized time complexity of handling item $a_{i,j}$ is $O(\log \frac{a_{i,j}}{\epsilon} \log \frac{1}{\delta})$ and the worst-case complexity is $O(\frac{\log\log n \log a_{i,j}}{\epsilon^2} \log \frac{1}{\delta})$.

As shown below, our algorithm for dominance norms can easily be generalized to the distributed context.

5.2. Distributed streams.

In the *distributed streams model* [GT01], the data arrives as k independent streams, where for $i = 1 \ldots k$ stream i goes to party i. Each party processes its complete stream and then sends the contents of their workspace to a common referee. Similar to one-round simultaneous communication complexity, there is no communication allowed between the parties themselves. The referee is required to estimate the aggregate F_0 over the *union* of all the data streams

1 to k. The space complexity is the sum of the sizes of all messages sent to the referee.

The range-efficient F_0 algorithm can be readily adapted to the distributed streams model. All the parties share a common hash function h, and each party runs the above-described (single party) algorithm on its own stream, targeting a sample size of $c\alpha$ intervals. Finally, each party sends its sample to the referee.

It is possible that samples sent by different parties are at different sampling probabilities (or, equivalently, are at different sampling levels). The referee constructs a sample of the union of the streams by subsampling each stream to the lowest sampling probability across all the streams. By our earlier analysis, each individual stream is at a sampling probability which will likely give good estimates. Thus the final sampling probability will also give us an (ϵ, δ)-estimator for F_0. The space complexity (total space used across all nodes) of this scheme is $O(k\frac{\log 1/\delta \log n}{\epsilon^2})$, where k is the number of parties, and the time per item is the same as in the single stream algorithm.

5.3. Range-efficiency in every coordinate. If each data point is a vector of dimension d rather than a single integer, then a modified definition of range-efficiency is required. One possible definition was given by [BYKS02], *range-efficiency in every coordinate*, and this proved to be useful in their reduction from the problem of computing the number of triangles in graphs to that of computing F_0 and F_2 of an integer stream.

For vectors of dimension d, define a jth coordinate range $(a_1, \ldots, a_{j-1}, [a_{j,s}, a_{j,e}], a_{j+1}, \ldots, a_d)$ to be the set of all vectors $\hat{x}$ with the ith coordinate $x_i = a_i$ for $i \neq j$ and $x_j \in [a_{j,s}, a_{j,e}]$. An algorithm is said to be range-efficient in the jth coordinate if it can handle a jth coordinate in a range-efficient manner.

The F_0 algorithm can be made range-efficient in every coordinate in the following way: We first find a mapping g between a d-dimensional vector $a = (a_1, a_2 \ldots a_d)$ (where for all $i = 1 \ldots d$, $a_i \in [0, m-1]$) and the one-dimensional line as follows:

$g(a_1, a_2, \ldots a_d) = m^{d-1}a_1 + m^{d-2}a_2 + \cdots + m^0 a_d$.

Fact 4. *Function g has the following properties:*

- *g is an injective function.*
- *A jth coordinate range $(a_1, \ldots, a_{j-1}, [a_{j,s}, a_{j,e}], a_{j+1}, \ldots, a_d)$ maps to an arithmetic progression $g(y_1), \ldots, g(y_n)$, where $y_i \in (a_1, \ldots, a_{j-1}, [a_{j,s}, a_{j,e}], a_{j+1}, \ldots, a_d)$.*

Because of the above, the number of distinct elements does not change when we look at the stream $g(x)$ rather than stream x. Because of Fact 4 and since the range-efficient F_0 algorithm can handle an arithmetic sequence of integers rather than just intervals, it follows that the algorithm can be made range-efficient in every coordinate.

Assuming the arithmetic operations on the integers that were mapped to take $O(d)$ time, the amortized processing time per item is $O(d \log \frac{1}{\delta} \left(\log n + \frac{1}{\epsilon^2}\right))$, and the time for answering a query is $O(d \log 1/\delta)$. The workspace used is $O(d\frac{1}{\epsilon^2} \log \frac{1}{\delta} \log n)$.

Finally, we note that the algorithm presented in this paper can handle streams in which each data item is an arithmetic progression rather than a range of consecutive integers. Let $[x_i, y_i]$ be a data item that represents an arithmetic progression $x_i, x_i + c, x_1 + 2c, \ldots y_i$. Now the range sample algorithm must compute the following:

$$RangleSample([x_i, y_i], \ell) = \left|\left\{x \in [x_i, y_i] \mid h(x) \in \left[0, \left\lfloor \frac{p}{2^\ell} \right\rfloor - 1\right]\right\}\right|.$$

Now the sequence $h(x_i), h(x_i + c), \ldots, h(y_i)$ forms an arithmetic progression over $\mathbb{Z}_p$ with a common difference ac, instead of common difference a. The range sampling algorithm described in section 3 can handle this case.

Acknowledgments. We thank Graham Cormode for pointing out the connection to dominance norms and for suggestions which helped improve the running time of the algorithm. We thank the anonymous referees for their comments that greatly improved the presentation.

REFERENCES

[AJKS02] M. Ajtai, T. S. Jayram, R. Kumar, and D. Sivakumar, *Approximate counting of inversions in a data stream*, in Proceedings of the 37th ACM Symposium on Theory of Computing (STOC), 2002, pp. 370–379.

[AMS99] N. Alon, Y. Matias, and M. Szegedy, *The space complexity of approximating the frequency moments*, J. Comput. System Sci., 58 (1999), pp. 137–147.

[BBD+02] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, *Models and issues in data stream systems*, in Proceedings of the 21st ACM Symposium on Principles of Database Systems (PODS), 2002, pp. 1–16.

[BFL+06] L. S. Buriol, G. Frahling, S. Leonardi, A. Marchetti-Spaccamela, and C. Sohler, *Counting triangles in data streams*, in Proceedings of the ACM Symposium on Principles of Database Systems (PODS), 2006, pp. 253–262.

[BYJK+02] Z. Bar-Yossef, T. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan, *Counting distinct elements in a data stream*, in Proceedings of the 6th International Workshop on Randomization and Approximation Techniques (RANDOM), Lecture Notes in Comput. Sci. 2483, Springer, Berlin, 2002, pp. 1–10.

[BYKS02] Z. Bar-Yossef, R. Kumar, and D. Sivakumar, *Reductions in streaming algorithms, with an application to counting triangles in graphs*, in Proceedings of the 13th ACM-SIAM Symposium on Discrete Algorithms (SODA), 2002, pp. 623–632.

[CDIM02] G. Cormode, M. Datar, P. Indyk, and S. Muthukrishnan, *Comparing data streams using hamming norms (how to zero in)*, IEEE Transactions on Knowledge and Data Engineering, 15 (2003), pp. 529–540.

[CGL+05] A. R. Calderbank, A. Gilbert, K. Levchenko, S. Muthukrishnan, and M. Strauss, *Improved range-summable random variable construction algorithms*, in Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA), 2005.

[CK04] D. Coppersmith and R. Kumar, *An improved data stream algorithm for frequency moments*, in Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms (SODA), 2004, pp. 151–156.

[CLKB04] J. Considine, F. Li, G. Kollios, and J. Byers, *Approximate aggregation techniques for sensor databases*, in Proceedings of the 20th IEEE International Conference on Data Engineering (ICDE), 2004, pp. 449–460.

[CM03] G. Cormode and S. Muthukrishnan, *Estimating dominance norms of multiple data streams*, in Proceedings of the 11th European Symposium on Algorithms (ESA), Lecture Notes in Comput. Sci. 2832, Springer, Berlin, 2003, pp. 148–160.

[CW79] J. L. Carter and M. L. Wegman, *Universal classes of hash functions*, J. Comput. System Sci., 18 (1979), pp. 143–154.

[DGIM02] M. Datar, A. Gionis, P. Indyk, and R. Motwani, *Maintaining stream statistics over sliding windows*, SIAM J. Comput., 31 (2002), pp. 1794–1813.

[FKSV02] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan, *An approximate L^1-difference algorithm for massive data streams*, SIAM J. Comput., 32 (2002), pp. 131–151.

[FM85] P. Flajolet and G. N. Martin, *Probabilistic counting algorithms for database applications*, J. Comput. System Sci., 31 (1095), pp. 182–209.

[GGI+02] A. C. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, and M. Strauss, *Fast, small-space algorithms for approximate histogram maintenance*, in Proceedings of the ACM Symposium on Theory of Computing (STOC), 2002, pp. 389–398.

[GKMS03] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss, *One-pass wavelet decompositions of data streams*, IEEE Trans. Knowl. Data Eng., 15 (2003), pp. 541–554.

[Gol97] O. Goldreich, *A Sample of Samplers: A Computational Perspective on Sampling (Survey)*, Technical report TR97-020, Electronic Colloquium on Computational Complexity, 1997.

[GT01] P. B. Gibbons and S. Tirthapura, *Estimating simple functions on the union of data streams*, in Proceedings of the ACM Symposium on Parallel Algorithms and Architectures (SPAA), 2001, pp. 281–291.

[HNSS95] P. J. Haas, J. F. Naughton, S. Seshadri, and L. Stokes, *Sampling-based estimation of the number of distinct values of an attribute*, in Proceedings of the 21st International Conference on Very Large Data Bases, Morgan Kaufmann, San Mateo, CA, 1995, pp. 311–322.

[IW03] P. Indyk and D. Woodruff, *Tight lower bounds for the distinct elements problem*, in Proceedings of the 44th IEEE Symposium on Foundations of Computer Science (FOCS), 2003, p. 283.

[JG05] H. Jowhari and M. Ghodsi, *New streaming algorithms for counting triangles in graphs*, in Proceedings of the International Conference on Computing and Combinatorics (COCOON), Lecture Notes in Comput. Sci. 3595, Springer, Berlin, 2005, pp. 710–716.

[Mut05] S. Muthukrishnan, *Data Streams: Algorithms and Applications*, Foundations and Trends in Theoretical Computer Science, Now Publishers, 2005.

[NGSA04] S. Nath, P. B. Gibbons, S. Seshan, and Z. Anderson, *Synposis diffusion for robust aggregation in sensor networks*, in Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, ACM Press, New York, 2004, pp. 250–262.

SIAM J. COMPUT.
Vol. 37, No. 2, pp. 380–400

DERANDOMIZATION IN CRYPTOGRAPHY*

BOAZ BARAK[†], SHIEN JIN ONG[‡], AND SALIL VADHAN[‡]

Abstract. We give two applications of Nisan–Wigderson-type (NW-type) ("noncryptographic") pseudorandom generators in cryptography. Specifically, assuming the existence of an appropriate NW-type generator, we construct the following two protocols: (1) a one-message witness-indistinguishable proof system for every language in NP, based on any trapdoor permutation. This proof system does not assume a shared random string or any setup assumption, so it is actually an "NP proof system." (2) a noninteractive bit-commitment scheme based on any one-way function. The specific NW-type generator we need is a hitting set generator fooling *nondeterministic circuits*. It is known how to construct such a generator if $E = DTIME(2^{O(n)})$ has a function of nondeterministic circuit complexity $2^{\Omega(n)}$. Our witness-indistinguishable proofs are obtained by using the NW-type generator to derandomize the ZAPs of Dwork and Naor [*Proceedings of the 41st Annual ACM Symposium on Foundations of Computer Science*, 2000, pp. 283–293]. To our knowledge, this is the first construction of an NP proof system achieving a secrecy property. Our commitment scheme is obtained by derandomizing the interactive commitment scheme of Naor [*J. Cryptology*, 4 (1991), pp. 151–158]. Previous constructions of noninteractive commitment schemes were known only under incomparable assumptions.

Key words. interactive proofs, witness-indistinguishable proofs, commitment schemes, complexity theory, pseudorandom generators

AMS subject classifications. 94A60, 68Q99

DOI. 10.1137/050641958

1. Introduction.

The computational theory of pseudorandomness has been one of the most fertile grounds for the interplay between cryptography and computational complexity. This interplay began when Blum, Micali, and Yao (BMY) [10, 50], motivated by applications in cryptography, placed the study of pseudorandom generators (PRGs) on firm complexity-theoretic foundations. They gave the first satisfactory definition of PRGs along with constructions meeting that definition. Their notion quickly acquired a central position in cryptography, but it turned out that the utility of PRGs was not limited to cryptographic applications. In particular, Yao [50] showed that they could also be used for *derandomization*—efficiently converting randomized algorithms into deterministic algorithms. PRGs and their generalization, pseudorandom functions [21], also found a variety of other applications in complexity theory and the theory of computation (e.g., [43, 49]).

Focusing on derandomization, Nisan and Wigderson (NW) [41] proposed a weakening of the BMY definition of PRGs which still suffices for derandomization. The

*Received by the editors October 4, 2005; accepted for publication (in revised form) November 16, 2006; published electronically May 16, 2007. A preliminary version of this work appeared in Proceedings of the 23rd Annual International Cryptology Conference (CRYPTO 2003), Santa Barbara, CA [5].

http://www.siam.org/journals/sicomp/37-2/64195.html

[†] Department of Computer Science, Princeton University, Princeton, NJ (boaz@cs.princeton.edu). This author's work was supported by a Clore Foundation Fellowship and an Israeli Higher Education Committee Fellowship.

[‡] Division of Engineering and Applied Sciences, Harvard University, Cambridge, MA (shienjin@eecs.harvard.edu, salil@eecs.harvard.edu). The work of the first author was done mainly while an undergraduate at MIT, supported by an MIT Eloranta Fellowship and the MIT Reed UROP Fund. This author's work is currently supported by ONR grant N00014-04-1-0478. The work of the second author was supported by NSF grants CCR-0205423, CCR-0133096, and CNS-0430336 and by a Sloan Research Fellowship.

benefit was that such NW-type PRGs could be constructed under weaker assumptions than the BMY ones (circuit lower bounds for exponential time rather than the existence of one-way functions).[1] Thus, a long body of work developed around the task of constructing increasingly efficient NW-type PRGs under progressively weaker assumptions. One of the highlights of this line of work is the construction of Impagliazzo and Wigderson [33] implying that P = BPP, i.e., the class of lanuages decidable in polynomial time equals the class of languages decidable in probabilistic polynomial time (BPP), under the plausible assumption that E = DTIME($2^{O(n)}$) has a problem of circuit complexity $2^{\Omega(n)}$. More recently, the work on NW-type PRGs has also been found to be intimately related to randomness extractors [48] and has been used to prove complexity-theoretic results which appear unrelated to derandomization (e.g., [31]).

While allowing remarkable derandomization results such as the Impagliazzo–Wigderson result mentioned above, NW-type PRGs have not previously found applications in cryptography (for reasons mentioned below). In this work, we show that a stronger form of NW-type PRGs, namely, ones fooling *nondeterministic circuits* [2, 35, 38, 47], do have cryptographic applications. Using such PRGs (which can be constructed under plausible complexity assumptions), we

1. construct witness-indistinguishable (WI) "NP-proofs" (i.e., one-message[2] proof systems, with a deterministic verifier strategy, and no shared random string or other setup assumptions) for every language in NP, assuming the existence of trapdoor permutations;
2. construct *noninteractive* bit-commitment schemes from any one-way function.

Thus, each of these results requires two assumptions—the circuit complexity assumption for the NW-type PRG (roughly, that E has a function of nondeterministic circuit complexity $2^{\Omega(n)}$) and a "cryptographic" assumption (one-way functions or trapdoor permutations).

Result 1 is the first construction of WI NP-proofs under any assumption whatsoever and shows that interaction is not necessary to achieve secrecy in proof systems. It is obtained by derandomizing the ZAP construction of Dwork and Naor [15]. We note that Dwork and Naor [15] themselves also constructed one-message WI proofs that are *nonuniform* in the sense that the prover and verifier require a polynomial-length string to be hardwired in advance as a nonuniform advice. Those can be viewed as "NP/poly proofs."

Result 2 is not the first construction of noninteractive commitment schemes but is based on assumptions that appear incomparable to previous ones (which were based on the existence of one-to-one one-way functions). We obtain this result by derandomizing the Naor's interactive bit-commitment scheme [39].

These two examples suggest that NW-type PRGs (and possibly other "noncryptographic" tools from the derandomization literature) are actually relevant to the foundations of cryptography, and it seems likely that other applications will be found in the future.

[1] Here and throughout this paper, when we say that assumption X is weaker than assumption Y, we mean that Y is known to imply X, but X is not known to imply Y. Strictly speaking, the assumptions for NW-type generators are weaker only in this sense when considering generators of the same stretch (and when fooling nonuniform circuits). In this paper, the NW-type generators we use have a much greater stretch than the BMY-type generators we use, and hence the assumptions are incomparable.

[2] We use "messages" rather than "rounds," as the latter is sometimes used to refer to a pair of messages.

NW-type generators fooling nondeterministic circuits. The most important difference between BMY-type and NW-type PRGs is that BMY-type PRGs are required to fool even circuits with greater running time than the generator, whereas NW-type PRGs are allowed greater running time than the adversarial circuit. Typically, a BMY-type PRG must run in some fixed polynomial time (say, n^c) and fool all polynomial-time circuits (even those running in time, say, n^{2c}). In contrast, an NW-type PRG may run in time $n^{O(c)}$ (e.g., n^{3c}) in order to fool circuits running in time n^{2c}. BMY-type PRGs are well suited for cryptographic applications, where the generator is typically run by the legitimate parties and the circuit corresponds to the adversary (who is always allowed greater running time). In contrast, NW-type PRGs seem noncryptographic in nature. Nevertheless we are able to use them in cryptographic applications. The key observation is that, in the protocols we consider, (some of) the randomness is used to obtain a string that satisfies some fixed property *which does not depend on the adversary (or its running time)*. Hence, if this property can be verified in some fixed polynomial time, we can obtain the string using an NW-type PRG of a larger fixed polynomial running time. We then eliminate the randomness entirely by enumerating over all possible seeds. This is feasible because NW-type generators can have logarithmic seed length. Also, we show that in our specific applications this enumeration does not compromise the protocol's security.

In the protocols we consider, the properties in question do not seem to be verifiable in polynomial time. However, they are verifiable in *nondeterministic* polynomial time. So we need to use a PRG that fools nondeterministic circuits. Fortunately, it is possible for an *NW-type* PRG to fool nondeterministic circuits, as realized by Arvind and Köbler [2] and Klivans and van Melkebeek [35].[3] Indeed, a sequence of works have constructed such PRGs under progressively weaker complexity assumptions [2, 35, 38, 47]. (Recently, these assumptions were all shown to be equivalent [46].) Our results make use of the Miltersen–Vinodchandran construction [38] (which gives only a "hitting set generator" (HSG) rather than a PRG, but this suffices for our applications) and its "uniform analogue" from [29].

WI NP-proofs. In order to make zero-knowledge proofs possible, the seminal paper of Goldwasser, Micali, and Rackoff [27] augmented the classical notion of an NP proof with two new ingredients—interaction and randomization. Both were viewed as necessary for the existence of zero-knowledge proofs, and indeed it was proven by Goldreich and Oren [25] that without either, zero-knowledge proofs exist only for trivial languages (those in BPP). The role of interaction was somewhat reduced by the introduction of "noninteractive" zero-knowledge (NIZK) proofs [9, 8], but those require a shared random string selected by a trusted third party, which can be viewed as providing a limited form of interaction. Given the aforementioned impossibility results [25], reducing the interaction further seems unlikely. Indeed, a truly noninteractive proof system, in which the prover sends a single proof string to the verifier, seems to be inherently incompatible with the intuitive notion of "zero knowledge": from such a proof, the verifier gains the ability to prove the same statement to others.

Despite this, we show that for a natural weakening of zero knowledge, namely, *WI* [17], the interaction *can* be completely removed (under plausible complexity assumptions). Recall that a WI proof system for a language $L \in \text{NP}$ is an interactive

[3] It is impossible for a BMY-type PRG to fool nondeterministic circuits, as such a circuit can recognize outputs of the PRG by guessing the corresponding seed and evaluating the generator to check. Some attempts to bypass this difficulty can be found in [45].

proof system for L that leaks no knowledge about which witness is being used by the prover (as opposed to leaking no knowledge at all, as in zero-knowledge proofs) [17]. WI suffices for a number of the applications of zero knowledge [17] and also is a very useful intermediate step in the construction of zero-knowledge proofs [16].

Several prior results show that WI proofs do not require the same degree of interaction as zero-knowledge proofs. Feige and Shamir [17] constructed three-message WI proofs for NP (assuming the existence of one-way functions), whereas the existence of three-message zero-knowledge proofs is a long-standing open problem. More recently, the ZAPs of Dwork and Naor [15] achieved witness indistinguishability with just two messages (assuming trapdoor permutations), whereas this is known to be impossible for zero knowledge [25]. As mentioned earlier, Dwork and Naor also showed that the interaction could be further reduced to one message at the price of *nonuniformity* (i.e., if the protocol can use some nonuniform advice of polynomial length); they interpret this as evidence that "*proving* a lower bound of two [messages] is unlikely."

We construct one-message WI proofs for NP in the "plain model," with no use of a shared random string or nonuniformity. Our proof system is obtained by derandomizing the verifier in the ZAPs of Dwork and Naor [15] via an NW-type generator against nondeterministic circuits. The prover, however, remains probabilistic polynomial time given a witness for membership. Since our verifier is deterministic, we actually obtain a standard NP proof system with the WI property. More precisely, for any language $L \in \mathrm{NP}$ with associated NP-relation R, we construct a new NP-relation R' for L. The relation R' has the property that one can efficiently transform any witness with respect to R into a distribution of new witnesses with respect to R', such that the distributions originating from different witnesses of R are computationally indistinguishable.

Converting Arthur–Merlin (AM) proof systems to NP proof systems was actually one of the original applications of NW-type generators versus nondeterministic circuits [2, 35]. The novelty in our result comes from observing that this conversion preserves the WI property.

The randomness requirements of *zero-knowledge* proofs have been examined in previous works. Goldreich and Oren [25] showed that only languages in BPP have zero-knowledge proofs in which either the prover or verifier is deterministic. Thus De Santis, Di Crescenzo, and Persiano [12, 13, 14] have focused on reducing the number of random bits. Specifically, under standard "cryptographic" assumptions, they constructed NIZK proofs with a shared random string of length $O(n^\varepsilon + \log(1/s))$ and two-message WI proofs (actually, ZAPs) in which the verifier uses only $O(n^\varepsilon + \log(1/s))$ random bits, where $\varepsilon > 0$ is any constant and s is the soundness error. They posed the existence of one-message WI proofs for NP as an open problem. One of their main observations in [14] is that combinatorial methods for randomness-efficient error reduction, such as pairwise independence and expander walks, preserve WI. As mentioned above, we make crucial use of an analogous observation about NW-type generators.

Noninteractive bit-commitment schemes. Bit-commitment schemes are one of the most basic primitives in cryptography, used pervasively in the construction of zero-knowledge proofs [24] and other cryptographic protocols. Informally, a bit-commitment scheme is a two-stage protocol between two interacting parties, the *sender* and the *receiver*. In the *commitment stage*, the sender commits to a secret bit b. In the *decommitment stage*, the sender decommits by revealing the bit b together with an additional secret key that enables the receiver to verify that b is consistent with the commitment stage. The commitment stage alone must not reveal any information about b. This is called the *hiding* property. In addition, we require that the commitment be *binding*; i.e., the sender should not be able to successfully decommit to more than one value.

Here we focus on perfectly (or statistically) binding and computationally hiding bit-commitment schemes; i.e., even computationally unbounded senders should not be able to decommit to different values, but the hiding property holds only against efficient (probabilistic polynomial-time) receivers. *Noninteractive* bit-commitment schemes, in which the commitment phase consists of a single message from the sender to the receiver, are generally preferred over interactive schemes. There is a simple construction of noninteractive bit-commitment schemes from any *one-to-one* one-way function [7, 50, 23]. From general one-way functions, the only known construction of bit-commitment schemes, namely, Naor's protocol [39] with the PRG construction of [30], requires interaction.

We show how to use an NW-type PRG against nondeterministic circuits to remove the interaction in Naor's protocol, yielding noninteractive bit-commitment schemes under assumptions that appear incomparable to the existence of one-to-one one-way functions. In particular, ours is a "raw hardness" assumption, not requiring hard functions with any structure such as being one-to-one.

From a different perspective, our result shows that noncryptographic assumptions (nondeterministic circuit lower bounds for E) can reduce the gap between one-way functions and one-to-one one-way functions. In particular, a noninteractive bit-commitment scheme gives rise to a "partially one-to-one one-way function": a polynomial-time computable function $f(x, y)$ such that x is uniquely determined by $f(x, y)$ and x is hard to compute from $f(x, y)$ (for random x, y). It would be interesting to see if this can be pushed further to actually construct one-to-one one-way functions from general one-way functions under a noncryptographic assumption.

Perspective. The assumption used in the construction of NW-type generators is a strong one, but it seems to be plausible (see section 2.6). Perhaps its most significant feature is that it is very different than the assumptions typically used in cryptography (e.g., it is a worst-case assumption); nevertheless, our results show it has implications in cryptography. In our first result, we use it to demonstrate the plausibility of one-message WI proofs for all of NP, which will hopefully lead to efficient constructions for specific problems based on specific assumptions. As for our second result, the plausibility of noninteractive commitment schemes was already established more convincingly based on one-to-one one-way functions [7]. What we find interesting instead is that a noncryptographic assumption can imply new relationships between basic cryptographic primitives and in particular reduce the gap between one-way functions and one-to-one one-way functions.

2. Preliminaries.

Let X be a *random variable* taking values in a finite set T. We write $x \leftarrow X$ to indicate that x is selected according to X. For a finite set S, we write $x \leftarrow S$ to indicate that x is selected uniformly amongst all the elements of S.

We write $\text{neg}(n)$ to denote a *negligible function*, namely, one that vanishes more quickly than any inverse polynomial. That is, for all $c \in \mathbb{N}$, $\text{neg}(n) < n^{-c}$ for all sufficiently large n. We write $\text{poly}(n)$ to denote any polynomially bounded function; i.e., $\text{poly}(n) \leq n^c$ for some $c \in \mathbb{N}$ and for all sufficiently large n.

By *probabilistic polynomial time (PPT)*, we refer to probabilistic algorithms that run in *strict* polynomial time. A *nonuniform* PPT algorithm is a pair $(A, \bar{z})$, where

$\bar{z} = z_1, z_2, \ldots$ is an infinite series of strings where $|z_n| = \text{poly}(n)$ and A is a PPT algorithm that receives pairs of input of the form $(x, z_{|x|})$. (The string z_n is called the *advice string* for A for inputs of length n.) Nonuniform PPT algorithms are equivalent to families of polynomial-sized Boolean circuits.

The *statistical difference* between two random variables A and B over $\{0,1\}^n$ is defined as

$$\Delta(A, B) \overset{\text{def}}{=} \max_{T \subseteq \{0,1\}^n} |\Pr[A \in T] - \Pr[B \in T]| = \frac{1}{2} \sum_{x \in \{0,1\}^n} |\Pr[A \in T] - \Pr[B \in T]|.$$

We say that distributions A and B are *ε-close* if $\Delta(A, B) \leq \varepsilon$.

Let I be a set of strings. A *probability ensemble* of a sequence of random variables indexed by I is denoted as $\{A_x\}_{x \in I}$. Two probability ensembles $\{A_x\}_{x \in I}$ and $\{B_x\}_{x \in I}$ are *computationally indistinguishable* on $I \subseteq \{0,1\}^*$ if, for every PPT D, there exists a negligible function μ such that for all $x \in I$

$$\Big|\Pr[D(x, A_x) = 1] - \Pr[D(x, B_x) = 1]\Big| \leq \mu(|x|).$$

We say that $\{A_x\}_{x \in I}$ and $\{B_x\}_{x \in I}$ are *nonuniformly* computationally indistinguishable if the above holds for all *nonuniform* PPT D. Similarly, we say that $\{A_x\}_{x \in I}$ and $\{B_x\}_{x \in I}$ are *statistically indistinguishable* if the above holds for all functions D (instead of only PPT). Equivalently, $\{A_x\}_{x \in I}$ and $\{B_x\}_{x \in I}$ are statistically indistinguishable iff A_x and B_x are $\mu(|x|)$-close for some negligible function μ and all $x \in I$. Sometimes we will refer to ensembles indexed by natural numbers n, e.g., $\{A_n\}$ and $\{B_n\}$, in which case we apply the above definitions with $x = 1^n$. That is, we allow the distinguisher running time $\text{poly}(n, |A_n|)$ and require the distinguishing gap to be negligible in n.

2.1. Nondeterministic computations. A significant advantage of NW-type generators that we will use is that they can fool *nondeterministic* circuits, because even if such a circuit can guess the seed, it does not have enough time to evaluate the generator on it.

We define a nondeterministic circuit to be a (nonuniform) Boolean circuit that has the additional power of nondeterminism.

DEFINITION 2.1. *A* nondeterministic *Boolean circuit $C(x, y)$ is a circuit that takes x as its primary input and y as a witness. For each $x \in \{0,1\}^*$, we define $C(x) = 1$ if there exists a witness y such that $C(x, y) = 1$.*

A conondeterministic *Boolean circuit $C(x, y)$ is a circuit that takes x as its primary input and y as a witness. For each $x \in \{0,1\}^*$, we define $C(x) = 0$ if there exists a witness y such that $C(x, y) = 0$.*

Denote $S_{\text{N}}(f)$ to be the minimal sized nondeterministic circuit computing f.

Nondeterministic and conondeterministic algorithms can be defined in a similar fashion, with the nonuniform circuit C being replaced by a *uniform algorithm.* Naturally, we measure the running time of a nondeterministic algorithm $A(x, y)$ in terms of the first input x. Therefore NP and coNP are the classes of languages decidable by polynomial-time nondeterministic algorithms and conondeterministic algorithms, respectively.

DEFINITION 2.2. *A* nondeterministic *algorithm $A(x, y)$ is a uniform algorithm that takes x as its primary input and y as a witness. For each $x \in \{0,1\}^*$, we define $A(x) = 1$ if there exists a witness y such that $A(x, y) = 1$.*

Likewise, a conondeterministic *algorithm $A(x, y)$ is a uniform algorithm that takes x as its primary input and y as a witness. For each $x \in \{0,1\}^*$, we define $A(x) = 0$ if there exists a witness y such that $A(x, y) = 0$.*

A nondeterministic (or conondeterministic) algorithm A is said to run in time $t(n)$ if, for every x and y, the running time of $A(x, y)$ is at most $t(|x|)$.

2.2. Interactive proofs. An interactive proof is an interactive protocol in which a prover (with unlimited computational powers) tries to convince a probabilistic polynomial-time verifier of the validity of a certain statement. Since interactive protocols are probabilistic, the soundness and completeness criteria are also probabilistic. The formal definition of interactive proofs follows.

DEFINITION 2.3 (interactive proofs [3, 27]). *An interactive protocol (P, V) is called an* interactive proof system *for a language L if the following conditions hold.*

1. Efficiency: *On common input x, the number and total length of messages exchanged between P and V are bounded by a polynomial in $|x|$, and V is a PPT machine.*
2. Completeness: *If $x \in L$, then* $\Pr[(P, V)(x) = 1] \geq \frac{2}{3}$.
3. Soundness: *If $x \notin L$, then for any P^*,* $\Pr[(P^*, V)(x) = 1] \leq \frac{1}{3}$.

The class of languages possessing interactive proofs is denoted as IP.

We say that an interactive proof system has *perfect completeness* if the completeness condition holds with probability 1 instead of $\frac{2}{3}$. We say that a system has *perfect soundness* if the soundness condition holds with probability 0 instead of $\frac{1}{3}$.

An interactive proof system is called a *public-coin* if the verifier's messages consist only of random strings and acceptance is computed as a deterministic polynomial-time function of the interaction's transcript. An interactive proof system that is not a public-coin is called a *private-coin*.

The number of *rounds* in an interactive proof is the total number of messages exchanged in the interaction (i.e., both prover messages and verifier messages). A proof system with one round is called *noninteractive*.

2.3. The class AM. The class AM [3] has two equivalent formulations. The first is as the class of languages with constant-message interactive proofs. The second is as the class of languages decidable by polynomial-time *probabilistic* nondeterministic algorithms. Formally, a probabilistic nondeterministic algorithm $A(x, r, y)$ takes a random input r in addition to its regular input x and nondeterministic input y. We say A computes a function f if the following two conditions hold.

1. If $f(x) = 1$, then $\Pr_r[\exists y A(x, r, y) = 1] = 1$.
2. If $f(x) = 0$, then $\Pr_r[\exists y A(x, r, y) = 1] \leq 1/2$.

Then AM is the class of languages decidable by such algorithms $A(x, r, y)$ running in time poly($|x|$). The equivalence of the two definitions of AM is due to [3, 28, 18]. More generally, AMTIME($t(n)$) denotes the class of languages that are decided by probabilistic nondeterministic algorithms running in time $t(n)$, and [i.o.–AMTIME]($t(n)$) denotes the class of languages that are decided by probabilistic-time $t(n)$ nondeterministic algorithms for *infinitely many input lengths*. Formally, we say $L \in$ [i.o.–AMTIME]($t(n)$) if there exists an algorithm A running in time $t(n)$ such that for infinitely many $n \in \mathbb{N}$, the following two conditions hold for all x of length n.

1. If $x \in L$, then $\Pr_r[\exists y A(x, r, y) = 1] = 1$.
2. If $x \notin L$, then $\Pr_r[\exists y A(x, r, y) = 1] \leq 1/2$.

Note that the above definition of [i.o.–AMTIME]($t(n)$) is slightly nonstandard in the sense that infinitely often complexity classes are often defined in the following manner: If C is a complexity class, then io–C is the class of all languages L such that there exists a language $L' \in \mathrm{C}$ such that $L \cap \{0,1\}^n = L' \cap \{0,1\}^n$ for infinitely many $n \in \mathbb{N}$. Observe that io–AMTIME$(t(n)) \subseteq$ [i.o.–AMTIME]$(t(n))$. Discussions about the subtle difference between these two classes can be found in [29].

2.4. PRGs. A *PRG* is a deterministic algorithm $G\colon \{0,1\}^\ell \to \{0,1\}^m$, with $\ell < m$. PRGs are used to convert a short random string into a longer string that looks random to any efficient observer.

Definition 2.4 (PRG). *We say that $G\colon \{0,1\}^\ell \to \{0,1\}^m$ is a (s,ε)-*PRG against circuits *if, for all circuits $C\colon \{0,1\}^m \to \{0,1\}$ of size at most s, it holds that $|\Pr[C(G(U_\ell)) = 1] - \Pr[C(U_m) = 1]| < \varepsilon$, where U_k denotes the uniform distribution over $\{0,1\}^k$.*

BMY-type versus NW-type generators. As mentioned above, there are two main types of PRGs: the BMY-type [10, 50] and the NW-type [41] generator. Both can be defined for a wide range of parameters, but here we focus on the "classic" settings that we need. A BMY-type generator is the standard kind of PRG used in cryptography.

Definition 2.5 (BMY-type generators). *A function $G = \bigcup_m G_m : \{0,1\}^\ell \to \{0,1\}^m$ is a* BMY-type PRG *with seed length $\ell = \ell(m)$ if G is computable in time* poly(ℓ) *and, for every constant c, G_m is a $(m^c, 1/m^c)$-PRG against circuits for all sufficiently large m.*

Note that a BMY-type generator is required to have running time that is a fixed polynomial but must fool circuits whose running time is an arbitrary polynomial. Hástad et al. [30] proved that BMY-type PRGs with seed length $\ell(m) = m^\delta$ for every $\delta > 0$ exist iff one-way functions exist.

NW-type generators differ from BMY-type generators most significantly in the fact that the generator has a greater running time than the circuits it fools.

Definition 2.6 (NW-type generators). *A function $G = \bigcup_m G_m : \{0,1\}^\ell \to \{0,1\}^m$ is an* NW-type PRG *with seed length $\ell = \ell(m)$ if G is computable in time $2^{O(\ell)}$ and G_m is a $(m^2, 1/m^2)$-PRG against circuits for all m.*[4]

We will be interested in the "high end" NW-type generators, which have seed length $\ell(m) = O(\log m)$ and thus have a running time which is a fixed polynomial in m. The running time of such a generator, though polynomial, is allowed to be greater than the size of the circuits it fools. Impagliazzo and Wigderson [33] proved that such a generator exists if $\mathrm{E} = \mathrm{DTIME}(2^{O(n)})$ has a function of circuit complexity $2^{\Omega(n)}$. Note that when the seed length is $\ell = O(\log m)$, all 2^ℓ seeds can be enumerated in time poly(m), and hence the generator can be used for complete derandomization. In particular, the existence of such a generator implies that BPP = P.

2.5. Hitting set generators. A HSG is a deterministic algorithm $H(1^m, 1^s)$ that outputs a *set* of strings of length m. HSGs are weaker notions of PRGs.

Definition 2.7 (HSGs). *We say that H is an ε-HSG against circuits if, for every $m, s \in \mathbb{N}$ and circuit $C\colon \{0,1\}^m \to \{0,1\}$ of size at most s, the following holds:*

$$\Pr[C(U_m) = 1] > \varepsilon \implies \exists y \in H(1^m, 1^s) \text{ such that } C(y) = 1.$$

HSGs against nondeterministic *and* conondeterministic *circuits are defined analogously, replacing circuits with nondeterministic and conondeterministic circuits, respectively. In addition, we say that H is an ε-HSG against* conondeterministic uniform

[4] One can replace m^2 in this definition with any *fixed* polynomial in m.

algorithms *if, for every time-constructible function* $s(\cdot)$ *and every conondeterministic uniform algorithm* $A\colon \{0,1\}^* \to \{0,1\}$ *running in time at most* $s(m)$ *on inputs of length* m*, only the following holds for all sufficiently large* m*:*

$$\Pr[A(U_m) = 1] > \varepsilon \implies \exists y \in H(1^m, 1^{s(m)}) \text{ such that } A(y) = 1.$$

The construction of a one-message WI proof system in section 3 requires a HSG against conondeterministic circuits. However, we will need only a (weaker) HSG against conondeterministic uniform algorithms for the construction of a noninteractive commitment scheme in section 4.

Definition 2.8. *We say HSG* $H(1^m, 1^s)$ *is* efficient *if its running time is polynomial in* m *and* s*.*

Note that the running time of an efficient HSG, though a fixed polynomial, can be greater than the size of the circuits it fools. Hence, the HSGs defined correspond to NW-type generators. Furthermore, observe that a PRG $G\colon \{0,1\}^\ell \to \{0,1\}^m$ fooling circuits of size s gives rise to a HSG, by taking the set of outputs of G over all seeds. The HSG will be efficient if G is computable in time $\operatorname{poly}(s, m)$ and has logarithmic seed length $\ell = O(\log m + \log s)$. In this sense HSGs are weaker than PRGs. Indeed, HSGs can be directly used to derandomize algorithms with one-sided error (i.e., RP algorithms), whereas PRGs can be used to derandomize circuits with two-sided error (BPP algorithms). Since the error in AM proof systems can be made one-sided [18], i.e., the existence of an efficient 1/2-HSG against conondeterministic circuits implies that AM = NP.

The first constructions of efficient HSG (in fact PRGs) against conondeterministic circuits was given by Arvind and Köbler [2]. Their construction was based on the assumption that there are languages in E that are hard on average for nondeterministic circuits of size $2^{\Omega(n)}$. Klivans and van Melkebeek [35] gave a construction based on a *worst-case* hardness assumption, namely, the existence of languages in E that are worst-case circuits of size $2^{\Omega(n)}$ even with oracle gates for Satisfiability (SAT). Miltersen and Vinodchandran [38] managed to relax the hardness condition to nondeterministic circuits (but obtained only a HSG rather than a PRG). We state the main result of Miltersen and Vinodchandran [38].

Theorem 2.9 (see [38]).[5] *If there exists a function* $f \in E$ *such that* $S_{\mathrm{N}}(f) = 2^{\Omega(n)}$*, then there exists an efficient* $1/2$*-HSG against conondeterministic circuits. In particular, under this assumption* AM = NP*.*

Shaltiel and Umans [47] subsequently extended Theorem 2.9 in two ways: First, they obtained a PRG rather than a HSG. Second, they obtained analogous results for a quantitatively weaker assumption (e.g., when the $S_{\mathrm{N}}(f)$ is only superpolynomial rather than exponential) yielding correspondingly less efficient generators. However, we will not need these extensions in our paper.

Uniform HSGs. Gutfreund, Shaltiel, and Ta-Shma [29] extended Theorem 2.9 to give a HSG against conondeterministic *uniform algorithms* from *uniform* hardness assumptions. They used the same HSG as Miltersen and Vinodchandran but augmented the analysis.

[5] Miltersen and Vinodchandran [38] actually use a seemingly weaker assumption, needing a function only of exponential "single-valued" nondeterministic circuit complexity. But, it was recently shown in [46] that all of the assumptions used in this line of work [2, 35, 38] are in fact equivalent. In addition, [38] presented the HSG as a $(1-\delta)$-HSG for $\delta = 2^{m^\gamma}/2^m$, but it can be converted into a 1/2-HSG using dispersers as done implicitly in their paper.

Theorem 2.10 (see [29]). *If* $\mathrm{E} \not\subseteq \text{io-AMTIME}(2^{\delta n})$ *for some* $\delta > 0$*, then an efficient* $1/2$*-HSG against conondeterministic uniform algorithms exists.*

The assumption used in Theorem 2.10 is weaker than the assumption used in Theorem 2.9 since nonuniformity can be used to replace randomness.

2.6. Discussions.

Are the assumptions reasonable? Our two results rely on the existence of HSGs as constructed in Theorems 2.9 and 2.10, which in turn make assumptions about E containing functions of high nondeterministic complexity. In our opinion, these assumptions are plausible. The two most common reasons to believe a hardness assumption are empirical evidence and philosophical (or structural) considerations. The widely held $\mathrm{P} \neq \mathrm{NP}$ assumption is supported by both. Empirically, much effort has been invested into finding efficient algorithms for NP problems. Philosophically, it seems unlikely that proofs should always be as easy to find as they are to verify. Other hardness assumptions, such as the hardness of factoring, are supported mainly by empirical evidence. Some, like $\mathrm{E} \not\subseteq \mathrm{NP}$ (equivalently, $\mathrm{EXP} \neq \mathrm{NP}$), are supported mainly by philosophical considerations: it seems unlikely that it should *always* be possible to prove the correctness of exponentially long computations with polynomial-sized proofs. The assumptions of Theorems 2.9 and 2.10 are natural strengthenings of this assumption, where we extend NP both by letting the running time grow from polynomial to subexponential and by allowing nonuniformity or randomization.

How do we find the function f? Once we accept the existence of *some* function $f \in \mathrm{E}$ such that $S_{\mathrm{N}}(f) = 2^{\Omega(n)}$, can we find a *specific* function f satisfying that condition? The answer is yes. It is not hard to show that if there exists a function f satisfying the condition of Theorem 2.9, then *every* function that is E-complete via linear-time reductions also satisfies that condition. In particular, we can take the bounded halting function $\mathrm{BH}(\cdot)$ defined as follows: $\mathrm{BH}(M, x, t) = 1$ if the Turing machine M outputs 1 on input x after at most t steps (where t is given in binary form), and $\mathrm{BH}(M, x, t) = 0$, otherwise.

3. WI NP-proofs.

In this section we use efficient HSGs against conondeterministic circuits to derandomize the ZAP construction of Dwork and Naor [15] and to obtain a *noninteractive* WI proof system for any language in NP. We call this an "NP proof system" because it consists of a single message from the prover to the verifier, as is the case in the trivial NP proof of simply sending the witness to the verifier.

As in the trivial NP proof system, our verifier algorithm will be deterministic. Our prover algorithm, however, will be *probabilistic* to make our proof system WI. It remains open as to whether a probabilistic prover strategy is necessary to achieve the WI property. We stress that our proof system is in the *plain model*, without assumptions of a shared random string or nonuniformity. As far as we know, this is the first noninteractive proof system for NP in the plain model that satisfies a secrecy property.

3.1. Definitions.

Witness relation. Let $W \subseteq \{0,1\}^* \times \{0,1\}^*$ be a relation. Define $W(x) = \{w \mid (x, w) \in W\}$ and $L(W) = \{x \mid \exists w \text{ such that } (x, w) \in W\}$. If $w \in W(x)$, then we say that w is a *witness* for x. Recall that the class NP is the class of languages L such that $L = L(W)$ for a relation W that is decidable in time polynomial in the first input. If $L = L(W)$ is an NP language, then we say that W is a *witness relation* corresponding to L.

Efficient provers. Recall the notion of interactive proofs as defined in section 2.2. Let L be an NP language with witness relation W. We say that an interactive proof for L has an *efficient prover* if the honest prover strategy can be implemented by a PPT algorithm given $w \in W(x)$ as auxiliary input. In this paper we will be interested only in interactive proofs for NP that have efficient provers.

NP proof systems. An NP *proof system* is an interactive proof system that is degenerate in that it (a) consists of only a single message from the prover to the verifier, (b) satisfies both perfect completeness and perfect soundness, and (c) has a deterministic verifier. The prover, however, is allowed to be PPT given a witness of membership. Because the verifier is deterministic, an NP proof system for a language L induces a witness relation W corresponding to L by setting $W(x)$ to contain all the prover messages accepted by the verifier.

WI. We recall the notion of WI, as defined by Feige and Shamir [17].

Definition 3.1 (WI, [17]). *Let L be an* NP *language with witness relation W_L. Let (P, V) be a proof system for L, where P is an efficient (PPT) prover that gets a witness as auxiliary input.*

We say that (P, V) is WI *if, for every nonuniform polynomial-time verifier V^*, every $x \in L$, and every $w, w' \in W_L(x)$, the interaction of V^* with $P(w)$ is computationally indistinguishable from its interaction with $P(w')$. That is, the ensembles $\{\mathsf{output}_{V^*}(P(w), V^*)(x)\}_{x \in L, w, w' \in W_L(x)}$ and $\{\mathsf{output}_{V^*}(P(w'), V^*)(x)\}_{x \in L, w, w' \in W_L(x)}$ are (nonuniformly) computationally indistinguishable, where $\mathsf{output}_{V^*}(P(w), V^*)(x)$ is a random variable denoting the output of V^* after interacting with P when both receive common input x and P receives the witness w as a private auxiliary input. Without loss of generality, V^* outputs its* view *of the interaction, which consists of its coin tosses and the messages exchanged.*

Feige and Shamir also proved that WI is closed under concurrent composition [17]. We will need only the special case of parallel composition, defined as follows: For an interactive proof system (P, V) and a polynomial m, we define the *m-fold parallel execution* $(P_m(w), V_m)(x)$ to be the protocol whereby P_m and V_m execute $m(|x|)$ parallel copies of $(P(w), V)(x)$ with each party using independent coin tosses in each execution. Here "parallel" means that the ith message for each of the $m(|x|)$ executions are all sent simultaneously. Note that, while the honest parties are defined to behave independently in each of the $m(|x|)$ executions, an adversary need not do so. Indeed, this is why zero knowledge is not preserved under parallel composition [22, 17]. Nevertheless, WI is maintained.

Theorem 3.2 ([17]). *If (P, V) is WI, then (P_m, V_m) is WI for every polynomial m.*

ZAPs. A *ZAP* [15] is a two-message public-coin interactive proof system that is WI. Dwork and Naor proved the following theorem.

Theorem 3.3 (see [15]). *If (nonuniformly secure) trapdoor permutations exist,*[6] *then every language in* NP *has a ZAP.*

We note that the construction of ZAPs by [15] is actually based on the weaker assumption that NIZK (in the shared random string model) systems exist for every language in NP. Thus, our construction can also be based on this weaker assumption.

[6] We refer the reader to [19, sect. 2.4.4] for the definition of trapdoor permutations. Actually, the definition we use is what is called by Goldreich [20, App. C.1] an *enhanced* trapdoor permutation collection. Such a collection is known to exist based on the hardness of factoring assumption (by a minor modification of the Rabin function [42] as noted in [20, App. C.1]).

3.2. Our result. The main theorem of this section follows.

THEOREM 3.4. *Assume that there exists an efficient* $1/2$*-HSG against conondeterministic circuits and that (nonuniformly secure) trapdoor permutations exist. Then every language in* NP *has a WI* NP*-proof system.*

3.3. Proof of Theorem 3.4. We prove Theorem 3.4 by converting the ZAPs for languages in NP into WI NP-proof systems. Let L be an NP language with witness relation W_L, and let (P, V) be the ZAP for L. We denote the first message in a ZAP (the verifier's random coins sent to the prover) by r, and we denote the second message (sent by the prover to the verifier) by π. We let $\ell(n)$ denote the length of the verifier's first message in a proof for statements of length n. Let $x \in \{0,1\}^n \setminus L$. We say that $r \in \{0,1\}^{\ell(n)}$ is *sound* with respect to x if there does not exist a prover message π such that the transcript (x, r, π) is accepting. The statistical soundness of the ZAP scheme implies that, for every $x \in \{0,1\}^n \setminus L$, the probability that $r \leftarrow \{0,1\}^{\ell(n)}$ is sound with respect to x is very high, and in particular it is larger than $\frac{1}{2}$.

Our construction is based on the following observation. Let $q(n)$ be a polynomial that bounds the running time of the honest ZAP verifier in a proof of statements of length n. For every $x \in \{0,1\}^n \setminus L$, there exists a *conondeterministic* circuit C_x of size less than $p(n) < q(n)^2$ that outputs 1 iff a string r is sound with respect to x. We stress that the time to verify the soundness of a string r depends only on the running time of the honest verifier (in our case it is $p(n)$).

On input r, the circuit C_x will output 1 if there does not exist a prover message π such that the transcript (x, r, π) is accepting, and it will output 0 otherwise. Note that $\Pr[C_x(U_{\ell(n)}) = 1] > \frac{1}{2}$. Since H is a 1/2-HSG against conondeterministic circuits, we have that, for every $x \in \{0,1\}^n \backslash L$, there exists $r \in H(1^{\ell(n)}, 1^{p(n)})$ such that $C_x(r) = 1$. In other words, for every $x \in \{0,1\}^n \setminus L$, there exists a string $r \in H(1^{\ell(n)}, 1^{p(n)})$ such that r is sound with respect to x.

Our construction is as follows.

PROTOCOL 3.5. *One-message WI* NP*-proof for* $L \in$ NP.
On common input $x \in \{0,1\}^n$ and auxiliary input w for the prover, such that $(x, w) \in W_L$, do the following.

Prover's Message

1. Compute $(r_1, \ldots, r_m) \stackrel{\text{def}}{=} H(1^{\ell(n)}, 1^{p(n)})$.
2. Using the auxiliary input (witness) w and the ZAP prover algorithm, compute, for every $i \in [1, m]$, a string π_i that is the prover's response to the verifier's message r_i in a ZAP proof for x.
3. Send to verifier $(\pi_1, \ldots, \pi_m)$.

Verifier's Test

1. Compute $(r_1, \ldots, r_m) \stackrel{\text{def}}{=} H(1^{\ell(n)}, 1^{p(n)})$.
2. Given prover's message $(\pi_1, \ldots, \pi_m)$, run the ZAP verifier on the transcript (x, r_i, π_i) for every $i \in [1, m]$.
3. Accept if the ZAP verifier accepts *all* these transcripts.

Note that Protocol 3.5 is indeed a one-message system with a deterministic verifier, and it satisfies the perfect completeness property. Thus, to prove Theorem 3.4, we need to prove that it has perfect soundness and is WI.

LEMMA 3.6. *Protocol* 3.5 *is a perfectly sound proof system for L.*

Proof. Let $x \notin L$, with $|x| = n$. Since H is a HSG, there exists an $r_i \in H(1^{\ell(n)}, 1^{p(n)})$ that is sound with respect to x. This means that no prover's message π_i will make the ZAP verifier accept the transcript (x, r_i, π_i). Therefore, no string $\pi = (\pi_1, \dots, \pi_m)$ will make the verifier of Protocol 3.5 accept. □

LEMMA 3.7. *Protocol* 3.5 *is a WI proof system for L.*

Proof. This follows from the fact that the prover algorithm of Protocol 3.5 simply invokes m times the prover algorithm for the ZAP on m different verifier messages. Since the m-fold parallel composition of the ZAP is also WI (by Theorem 3.2), it follows that Protocol 3.5 is WI. More precisely, the output (or view) of the verifier after an execution of Protocol 3.5, where the prover uses witness w, can be simulated perfectly by a cheating verifier V_m^* that sends the m-tuple of messages $(r_1, \dots, r_m) = H(1^{\ell(n)}, 1^{p(n)})$ in the m-fold parallel composition $(P_m(w), V_m^*)$. Since the output of V_m^* is computationally indistinguishable for any two witnesses used by the prover P_m, so is the output of the verifier in Protocol 3.5. □

3.4. Applications of noninteractive WI proofs.

One-out-of-two oblivious transfer. As an application of ZAPs, Dwork and Naor [15] constructed a *three-message* one-out-of-two *oblivious transfer* (OT) protocol based on the quadratic residuosity assumption.[7] Informally, an OT protocol consists of two parties, a sender and a receiver. The sender has two secret input bits b_0 and b_1. The goal of the receiver is to select an input bit of the sender without letting the sender know which bit it has selected. The goal of the sender is to allow the chooser to learn only its selected input bit.

The first two rounds of the Dwork–Naor OT protocol consist of a ZAP (two-message WI proof) of a certain NP statement. Replacing the ZAP with our WI NP-proofs, we prove that same NP statement in only one message, thus allowing for a *two-message* OT with the same security properties.

We begin with the formal definition of OT that we use. We use the notation $\mathsf{output}_S(S(b_0, b_1; r_S), R(c, r_R))$ to represent the output of sender S (on inputs b_0 and b_1 and private randomness r_S) after interacting with receiver R (on inputs the choice bit c and private randomness r_R). Both parties are also given the security parameter k, but we omit it from the notation. We define $\mathsf{output}_R(S(b_0, b_1; r_S), R(c, r_R))$ in an analogous manner.

DEFINITION 3.8. *A* one-out-of-two OT protocol *(with security parameter k) consists of a polynomial-time sender S and a polynomial-time receiver R, satisfying the following conditions.*

1. Completeness: *For all b_0, b_1, $c \in \{0, 1\}$, we have the following condition:* $\Pr_{r_S, r_R}[\mathsf{output}_R(S(b_0, b_1; r_S), R(c, r_R)) = b_c] > 1 - \mathrm{neg}(k)$.
2. Computational privacy of receiver: *For all PPT cheating S^*, we have that* $\mathsf{output}_{S^*}(S^*, R(0; r_R))$ *is computationally indistinguishable from* $\mathsf{output}_{S^*}(S^*, R(1; r_R))$.
3. Statistical privacy of sender: *For every deterministic receiver strategy R^*, one of the two following conditions holds:*
   (a) *for every $b \in \{0, 1\}$,* $\mathsf{output}_{R^*}(S(0, b; r_S), R^*)$ *is statistically indistinguishable from* $\mathsf{output}_{R^*}(S(1, b; r_S), R^*)$*, or*

[7] For further information on the quadratic residuosity assumption, we refer the reader to [26, sect. 2.5.1].

(b) *for every* $b \in \{0,1\}$, $\mathsf{output}_{R^*}(S(b,0;r_S),R^*)$ *is statistically indistinguishable from* $\mathsf{output}_{R^*}(S(b,1;r_S),R^*)$.

Condition 3.8 intuitively says that the receiver obtains no information about at least one of the sender's inputs. Unlike simulation-based definitions, however, it does not guarantee that a cheating receiver "knows" which of the two inputs it is learning. Similar definitions have been used in previous works on OT with few rounds.

As mentioned above, we obtain a two-message OT protocol by using noninteractive WI proofs in the Dwork–Naor [15] protocol. The computational assumptions we make are the existence of a HSG against conondeterministic circuits and the quadratic residuosity assumption, the latter being inherited from [15]. (We can drop the assumption of trapdoor permutations in Theorem 3.4, because it is implied by the quadratic residuosity assumption.) The formal theorem is stated below.

THEOREM 3.9. *Suppose that there exists an efficient* $1/2$*-HSG against conondeterministic circuits and that the quadratic residuosity assumption holds. Then there exists a* two-message *one-out-of-two OT protocol.*

There are two points that we would like to note. First, our protocol does *not* use any *public key.* If we allow the sender to publish a public key, the Dwork–Naor OT protocol can be reduced to two messages by having the sender S publish the random string of the ZAP in the public key (this random string corresponds to the first message of the ZAP).

Second, there were several previous constructions of two-message OT protocols satisfying similar security properties as Definition 3.8. Naor and Pinkas [40] and Aiello, Ishai, and Reingold [1] independently constructed two-message OT protocols based on the decisional Diffie–Hellman (DDH) assumption. Recently and independently of our work, Kalai [34] constructed two-message OT protocols based on a variant of "smooth projective hash families" [11].

Weak zero knowledge. The standard notions of zero knowledge require at least three rounds of interaction for languages outside BPP [25, 4]. Subsequent to this work, Barak and Pass [6] proposed a weak form of zero-knowledge protocols for all languages in NP that consist only of a *single* round, i.e., a single message from the prover to the verifier. Like our one-message WI, their prover is randomized, and their verifier is deterministic. Their construction of such protocols utilizes noninteractive WI proofs, as constructed in this paper. The properties achieved are weaker in the following sense: The weak zero-knowledge condition allows the simulator to run in *quasi-polynomial* time instead of polynomial time, and the computational soundness is guaranteed only against *uniform* PPT cheating provers.

4. Noninteractive bit commitment.

Bit-commitment schemes are basic primitives in cryptography. Informally, a bit-commitment scheme is a protocol that consists of two interacting parties, the sender and the receiver. The first step of the protocol involves the sender giving the receiver a commitment to a secret bit b. In the next step, the sender decommits the bit b by revealing a secret key. The commitment alone (without the secret key) must not reveal any information about b. This is called the *hiding* property. In addition, we require that the commitment to b be *binding*; i.e., the sender should not be able to decommit to a different bit $\bar{b}$. Note that, given a bit-commitment scheme, a string-commitment scheme can be obtained by independently committing to the individual bits of the string (cf. [19]).

In an *interactive bit-commitment scheme*, the sender and the receiver are allowed to interact during the commitment and decommitment steps. The formal definition of an interactive bit-commitment scheme can be found in [19]. Often, however, *nonin-*

teractive bit-commitment schemes are preferred or even crucial. For these, a simpler definition can be given.

Definition 4.1 (noninteractive bit commitment). *A* noninteractive bit-commitment scheme *is a polynomial-time algorithm S which takes a bit $b \in \{0,1\}$ and a random key $K \leftarrow \{0,1\}^{\mathrm{poly}(k)}$, where k is the security parameter, and outputs a commitment $C = S(b;K)$. The algorithm S must satisfy the following two conditions:*

1. *Binding: There do not exist keys K and K' such that $S(0;K) = S(1;K')$.*
2. *Hiding: The commitments to 0 and 1 are computationally indistinguishable. This means that the probability distributions $\{S(0;K)\}_{K\leftarrow\{0,1\}^{\mathrm{poly}(k)}}$ and $\{S(1;K)\}_{K\leftarrow\{0,1\}^{\mathrm{poly}(k)}}$ are computationally indistinguishable by PPT algorithms.*

We say that a bit-commitment scheme is nonuniformly *secure if the probability distributions $\{S(0;K)\}_{K\leftarrow\{0,1\}^{\mathrm{poly}(k)}}$ and $\{S(1;K)\}_{K\leftarrow\{0,1\}^{\mathrm{poly}(k)}}$ are nonuniformly computationally indistinguishable. This means that even nonuniform polynomial-sized circuits cannot distinguish between a commitment to 0 and a commitment to 1.*

There is a well-known construction by Blum [7] of a noninteractive bit-commitment scheme based on any *one-to-one* one-way function (using the function's hard-core predicate [50, 23]). Naor [39] gave a construction of an *interactive* bit-commitment scheme based on any one-way function (using PRGs [30]).

For completeness, we briefly describe a noninteractive bit-commitment protocol based on any *one-to-one* one-way function. First, note that a *one-to-one* one-way function can be transformed into another *one-to-one* one-way function with an associated hard-core predicate [50, 23]. Then, let f be a *one-to-one* one-way function, and let h be the hard-core predicate for f. A commitment to a bit $b \in \{0,1\}$ is just $\langle f(K), h(K) \oplus b\rangle$, where K is a randomly chosen key. The injectivity property of f seems crucial to guarantee the binding property of the commitment scheme.

4.1. Our result.

The main result of this section is the following theorem.

Theorem 4.2. *Assume that there exists an efficient 1/2-HSG against conondeterministic* uniform *algorithms. Then, noninteractive bit-commitment schemes exist iff one-way functions exist.*

The first condition is true if $\mathrm{E} \not\subseteq [\text{i.o.–AMTIME}](2^{\Omega(n)})$, by Theorem 2.10. We stress that the assumption of an efficient 1/2-HSG against conondeterministic *uniform* algorithms is sufficient, even if one wants to obtain a commitment scheme that is nonuniformly secure (i.e., commitments that are indistinguishable by polynomial-sized circuits). However, to get such schemes it will be necessary to assume that the one-way function is secure against *nonuniform* polynomial-sized circuits.

If we assume that the one-way function is only secure against uniform PPT adversaries, then we obtain commitment schemes secure against (uniform) PPT algorithms.

Our result is incomparable to the previous results on bit-commitment schemes. This is because the assumptions used in constructing our noninteractive commitments are on one hand stronger than Naor's [39], which only requires one-way functions, but Naor's scheme is interactive. On the other hand, our assumptions seem incomparable to assuming the existence of *one-to-one* one-way functions.

"Raw" hardness versus hardness with structure. Note that unlike assuming the existence of *one-to-one* one-way functions, we do not assume in Theorem 4.2 that there exists a hard function with a particular structure. Rather, we assume only that there exist functions with "raw hardness" (i.e., a one-way function and a function in E with high AM-complexity).

Even if one is told that one-to-one one-way functions exist, it is necessary to know a *particular* one-to-one one-way function to instantiate Blum's noninteractive commitment scheme. In contrast, we can construct a single noninteractive commitment scheme that is secure as long as there exists a one-way function and a function $f \in \mathrm{E} \setminus [\text{i.o.–AMTIME}](2^{\Omega(n)})$. This is because we can instantiate our scheme with a universal one-way function[8] [36] and a function that is E-complete via linear-time reductions such as the function $\mathrm{BH}(\cdot)$ (see the discussion in section 2.6).

4.2. Proof of Theorem 4.2. Our construction is based on derandomizing Naor's [39] *interactive* bit-commitment scheme using a HSG.

Let $G\colon \{0,1\}^k \to \{0,1\}^{3k}$ be a BMY-type PRG computable in time k^d for some constant d. Such a generator can be constructed based on any one-way function [30]. Naor [39] gave the following protocol for an interactive bit-commitment scheme, based on the existence of such a generator.

Protocol 4.3. *Interactive bit-commitment scheme* [39].
Input to receiver R: 1^k, where k is the security parameter.
Input to sender S: 1^k and a bit $b \in \{0,1\}$.
Commitment stage:
- **R:** Select a random $r \leftarrow \{0,1\}^{3k}$, and send r to S.
- **S:** Select a random $s \leftarrow \{0,1\}^k$. If $b = 0$, send $\alpha = G(s)$ to R. Else if $b = 1$, send $\alpha = G(s) \oplus r$ to R.

Decommitment stage:
- **S:** Reveal s and b.
- **R:** Accept if $b = 0$ and $\alpha = G(s)$, or $b = 1$ and $\alpha = G(s) \oplus r$.

Observe that when the sender commits to 0, the sender's message α is distributed according to $G(U_k)$. When the sender commits to 1, α is distributed according to $G(U_k) \oplus r$. The following lemma shows that Protocol 4.3 has the hiding property.

Lemma 4.4 (hiding property). *For every $r \in \{0,1\}^{3k}$, the distributions $G(U_k)$ and $G(U_k) \oplus r$ are computationally indistinguishable.*[9]

Proof. For any efficient adversary A, the pseudorandomness of G guarantees that

$$|\Pr[A(G(U_k)) = 1] - \Pr[A(U_{3k}) = 1]| < \varepsilon$$

and, for any given $r \in \{0,1\}^{3k}$,

$$|\Pr[A(G(U_k) \oplus r) = 1] - \Pr[A(U_{3k}) = 1]| < \varepsilon',$$

where ε and ε' are negligible. Hence, by the triangle inequality,

$$|\Pr[A(G(U_k) \oplus r) = 1] - \Pr[A(G(U_k)) = 1]| < \varepsilon + \varepsilon' = \mathrm{neg}(k).$$

This shows that no efficient adversary can distinguish between $G(U_k)$ and $G(U_k) \oplus r$. □

[8] The construction of such a universal one-way function can also be found in [19, sect. 2.4.1]. It uses the observation that if there exists a one-way-function, then there exists a one-way function that is computable in time n^2.

[9] To be exact, the condition of the lemma ("for every r") holds for nonuniform computational indistinguishability (assuming that G is a PRG against nonuniform circuits). For the uniform setting, the lemma still holds if the string r comes from any polynomial-time sampleable distribution. That is, $r \leftarrow R(1^k)$, where R is a PPT algorithm.

Define a string $r \in \{0,1\}^{3k}$ to be *good* for G if, for all $s, s' \in \{0,1\}^k$, we have $G(s) \neq G(s') \oplus r$. We have the following lemma.

LEMMA 4.5 (binding property). *The probability* $\Pr_{r \leftarrow \{0,1\}^{3k}}[r$ *is* good$] \geq 1 - 2^{-k}$.

Proof. Note that $G(s) \neq G(s') \oplus r$ iff $G(s) \oplus G(s') \neq r$. The total number of pairs (s, s'), with $s, s' \in \{0,1\}^k$, is 2^{2k}. For each pair, only one r is not good, namely, $r = G(s) \oplus G(s')$. Hence, the number of $r \in \{0,1\}^{3k}$ which are not good is at most 2^{2k}. This implies that the fraction of good $r \in \{0,1\}^{3k}$ is at least $1 - 2^{2k}/2^{3k} = 1 - 2^{-k}$. □

If the receiver selected a good r in the first step of the commitment stage of Protocol 4.3, then there do not exist $s, s' \in \{0,1\}^k$ such that $G(s) = G(s') \oplus r$, so no commitment α can be opened as both a 0 and 1. The probability of selecting a good r is high; hence, Protocol 4.3 is binding.

Our noninteractive bit-commitment scheme. Observe that the only interaction involved in Protocol 4.3 is in the receiver sending a random $r \in \{0,1\}^{3k}$ to the sender. However, one can see that the receiver does not have to send a random string, and it is enough to send a *good* string. This is because a good string r will make the distributions $G(U_k)$ and $G(U_k) \oplus r$ disjoint. As we show in the proof of Lemma 4.8, testing whether r is good can be done by a (uniform) *conondeterministic* algorithm running in time at most $3k^d$. Since the fraction of good r's is large, an efficient HSG against conondeterministic algorithms H can be used to select a candidate list of r's such that at least one element $r \in H$ is good. Thus, our protocol will be obtained by running the sender of Naor's protocol on each r in the hitting set. The resulting protocol follows.

PROTOCOL 4.6. *Noninteractive bit-commitment scheme.*
Input to receiver R: 1^k, where k is the security parameter.
Input to sender S: 1^k and a bit $b \in \{0,1\}$.

Commitment stage:

1. Compute $(r_1, \ldots, r_{p(k)}) \stackrel{\text{def}}{=} H(1^{3k}, 1^{3k^d})$.
2. Choose $s_1, \ldots, s_{p(k)}$ at random from $\{0,1\}^k$.
3. If $b = 0$, send $\alpha = \langle G(s_1), \ldots, G(s_{p(k)}) \rangle$.
   If $b = 1$, send $\alpha = \langle G(s_1) \oplus r_1, \ldots, G(s_{p(k)}) \oplus r_{p(k)} \rangle$.

Decommitment stage:

S reveals b and $\langle s_1, \ldots, s_{p(k)} \rangle$. R accepts if either of the following holds:

1. The bit $b = 0$ and $\alpha = \langle G(s_1), \ldots, G(s_{p(k)}) \rangle$.
   or
2. The bit $b = 1$ and $\alpha = \langle G(s_1) \oplus r_1, \ldots, G(s_{p(k)}) \oplus r_{p(k)} \rangle$.

To show that Protocol 4.6 constitutes a bit-commitment scheme (and hence to prove Theorem 4.2), we prove the following two lemmas.

LEMMA 4.7. *Protocol* 4.6 *has the hiding property of Definition* 4.1.

Proof. By Lemma 4.4, we know that for an $r \in \{0,1\}^{3k}$ generated by a polynomial-time algorithm, the distributions $G(U_k)$ and $G(U_k) \oplus r$ are computationally indistinguishable. Furthermore given r, the distributions $G(U_k)$ and $G(U_k) \oplus r$ are polynomial-time sampleable. Hence, by a hybrid/statistical-walk argument, the distributions $\langle G(U_k^1), G(U_k^2), \ldots, G(U_k^{p(k)})\rangle$ and $\langle G(U_k^1) \oplus r_1, G(U_k^2) \oplus r_2, \ldots, G(U_k^{p(k)}) \oplus r_{p(k)}\rangle$ are computationally indistinguishable for $\big(r_1, r_2, \ldots, r_{p(k)}\big) = H(1^{3k}, 1^{3k^d})$. ☐

LEMMA 4.8. *Protocol* 4.6 *has the binding property of Definition* 4.1.

Proof. Define the conondeterministic (uniform) algorithm A such that $A(r) = 1$ if, for all s, s' $G(s) \oplus G(s') \neq r$. Note that $A(r) = 1$ iff r is good. Therefore $\Pr[A(U_{3k}) = 1] \geq 1 - 2^{-k} > 1/2$. In addition, the running time of A (on inputs of length k) is bounded by $3k^d$. Hence, there exists an $r_i \in H(1^{3k}, 1^{3k^d})$ such that, for all s, s' $G(s) \oplus G(s') \neq r_i$. Therefore, there do not exist $s_1, \ldots, s_{p(k)}$ and $s'_1, \ldots, s'_{p(k)}$ such that

$$\langle G(s_1), \ldots, G(s_{p(k)})\rangle = \langle G(s'_1) \oplus r_1, \ldots, G(s'_{p(k)}) \oplus r_{p(k)}\rangle.$$

In other words, no commitment α can be opened as both a 0 and 1. Thus, Protocol 4.6 is perfectly binding. ☐

If one-way functions exists, then PRGs exist [30] and hence noninteractive bit-commitment schemes exist. Conversely, noninteractive bit-commitment schemes (as in Definition 4.1) imply the existence of one-way functions [32]. These facts, together with Lemmas 4.7 and 4.8, establish Theorem 4.2.

4.3. Partially one-to-one one-way functions. Another interpretation of our result is closing the gap between one-to-one and general one-way functions under a "noncryptographic" assumption.

DEFINITION 4.9. *A function* $f = \cup_k f_k \colon \{0,1\}^k \times \{0,1\}^k \to \{0,1\}^*$ *is a* partially one-to-one one-way function *if the following hold:*

1. *Easy to evaluate:* f *can be evaluated in polynomial time.*
2. *Partially one-to-one: If* $f(x, y) = f(x', y')$, *then* $x = x'$.
3. *Hard to invert: For every PPT algorithm* A, *the probability of inversion* $\Pr\left[A(1^k, f(X, Y)) = X\right]$ *is negligible in* k, *where the probability is taken over* X *and* Y *chosen independent and uniformly from* $\{0,1\}^k$, *and the coin tosses of* A.

LEMMA 4.10. *Partially one-to-one one-way functions exist iff noninteractive bit-commitment schemes exist.*

Proof. If f is a partially one-to-one one-way function, we can obtain a noninteractive bit-commitment scheme using the Goldreich–Levin hardcore bit [23]. Specifically, define $\texttt{Commit}(b; x, y, r) = (f(x, y), r, \langle x, r\rangle \oplus b)$, where $\langle \cdot, \cdot \rangle$ denotes inner product mod 2.

If a noninteractive bit-commitment scheme exists, we can obtain a partially one-to-one one-way function by first converting the bit-commitment scheme to a string-commitment scheme (by committing independently to each bit) and then defining $f(x, y) = \texttt{Commit}(x; y)$. (The fact that x and y may be of different lengths is inconsequential and can be fixed by padding.) ☐

Thus, a restatement of Theorem 4.2 is the following.

COROLLARY 4.11. *Assume that there exists an efficient* $1/2$*-HSG against conondeterministic uniform algorithms. Then one-way functions imply partially one-to-one one-way functions.*

5. Future work. Given the two examples we have presented here, it is natural to look for more applications of NW-type generators (and related notions in complexity theory) to cryptography. In parallel to this work, Barak, Lindell, and Vadhan [4] have used NW-type generators to obtain *negative* results about zero-knowledge proofs.

To facilitate the search for additional applications, we summarize the properties of the protocols (ZAPs and Naor's bit commitment) that enabled our derandomizations to work.

1. In order for the protocol to be secure, the random string r need satisfy only some fixed property that depends on only the algorithms of the "honest parties." In particular, it should be possible to verify this property by a nondeterministic algorithm that runs in a fixed polynomial time. (Algorithms even higher in the polynomial hierarchy can also be derandomized under stronger complexity assumptions [35].)
2. The protocol must remain secure under parallel repetition (with multiple choices of r, at least one of which satisfies the property above).

Another intriguing question is whether it can be shown that under a "noncryptographic" assumption, one-way functions imply truly one-to-one one-way functions (rather than just partially one-to-one one-way functions).

Finally, given our plausibility result, it is natural to look for additional constructions of nontrivial one-message WI proofs. Either constructions for specific problems based on specific assumptions or general constructions for all of NP based on alternative assumptions would be interesting. In addition to complexity-theoretic assumptions, it may also be useful to use assumptions from number theory, such as the extended Riemann hypothesis, which has been used for derandomization in the past (e.g., [37]).

Acknowledgments. We thank Oded Goldreich and the anonymous SIAM J. Comput. and CRYPTO 2003 reviewers for helpful comments.

REFERENCES

[1] W. Aiello, Y. Ishai, and O. Reingold, *Priced oblivious transfer: How to sell digital goods*, in Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT 2001), Lecture Notes in Comput. Sci. 2045, Springer, Berlin, 2001, pp. 119–135.

[2] V. Arvind and J. Köbler, *On pseudorandomness and resource-bounded measure*, Theoret. Comput. Sci., 255 (2001), pp. 205–221.

[3] L. Babai and S. Moran, *Arthur-Merlin games: A randomized proof system and a hierarchy of complexity classes*, J. Comput. System Sci., 36 (1988), pp. 254–276.

[4] B. Barak, Y. Lindell, and S. Vadhan, *Lower bounds for non-black-box zero knowledge*, J. Comput. System Sci., 72 (2006), pp. 321–391.

[5] B. Barak, S. J. Ong, and S. Vadhan, *Derandomization in cryptography*, in Proceedings of the 23rd Annual International Cryptology Conference (CRYPTO 2003), Lecture Notes in Comput. Sci. 2729, Springer, Berlin, 2003, pp. 299–315.

[6] B. Barak and R. Pass, *On the possibility of one-message weak zero-knowledge*, in Proceedings of the First Theory of Cryptography Conference (TCC 2004), Lecture Notes in Comput. Sci. 2961, Springer, Berlin, 2004, pp. 121–132.

[7] M. Blum, *Coin flipping by telephone—a protocol for solving impossible problems*, SIGACT News, 15 (1983), pp. 23–27.

[8] M. Blum, A. De Santis, S. Micali, and G. Persiano, *Noninteractive zero-knowledge*, SIAM J. Comput., 20 (1991), pp. 1084–1118.

[9] M. Blum, P. Feldman, and S. Micali, *Non-interactive zero-knowledge and its applications (extended abstract)*, in Proceedings of the 20th Annual ACM Symposium on Theory of Computing, 1988, pp. 103–112.

[10] M. Blum and S. Micali, *How to generate cryptographically strong sequences of pseudo-random bits*, SIAM J. Comput., 13 (1984), pp. 850–864.

[11] R. Cramer and V. Shoup, *Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption*, in Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2002), Lecture Notes in Comput. Sci. 2332, Springer, Berlin, 2002, pp. 45–64.

[12] A. De Santis, G. Di Crescenzo, and G. Persiano, *Randomness-efficient non-interactive zero-knowledge (extended abstract)*, in Proceedings of the 24th International Colloquium on Automata, Languages and Programming, Springer, Berlin, 1997, pp. 716–726.

[13] A. De Santis, G. Di Crescenzo, and G. Persiano, *Non-interactive zero-knowledge: A low-randomness characterization of NP*, in Proceedings of the 26th International Colloquium on Automata, Languages and Programming, Springer, Berlin, 1999, pp. 271–280.

[14] A. De Santis, G. Di Crescenzo, and G. Persiano, *Randomness-optimal characterization of two NP proof systems*, in Proceedings of the 6th International Workshop on Randomization and Approximation Techniques in Computer Science, Springer, Berlin, 2002, pp. 179–193.

[15] C. Dwork and M. Naor, *Zaps and their applications*, in Proceedings of the 41st Annual ACM Symposium on Foundations of Computer Science, 2000, pp. 283–293.

[16] U. Feige, D. Lapidot, and A. Shamir, *Multiple non-interactive zero knowledge proofs under general assumptions*, SIAM J. Comput., 29 (1999), pp. 1–28.

[17] U. Feige and A. Shamir, *Witness indistinguishable and witness hiding protocols*, in Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, Baltimore, Maryland, 1990, pp. 416–426.

[18] M. Furer, O. Goldreich, Y. Mansour, M. Sipser, and S. Zachos, *On completeness and soundness in interactive proof systems*, Adv. Comput. Res., 5 (1989), pp. 429–442.

[19] O. Goldreich, *Foundations of Cryptography: Basic Tools*, Cambridge University Press, Cambridge, 2001.

[20] O. Goldreich, *Foundations of Cryptography II: Basic Applications*, Cambridge University Press, Cambridge, 2004.

[21] O. Goldreich, S. Goldwasser, and S. Micali, *How to construct random functions*, J. ACM, 33 (1986), pp. 792–807.

[22] O. Goldreich and H. Krawczyk, *On the composition of zero-knowledge proof systems*, SIAM J. Comput., 25 (1996), pp. 169–192.

[23] O. Goldreich and L. A. Levin, *A hard-core predicate for all one-way functions*, in Proceedings of the 21st Annual ACM Symposium on Theory of Computing, 1989, pp. 25–32.

[24] O. Goldreich, S. Micali, and A. Wigderson, *Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems*, J. ACM, 38 (1991), pp. 691–729.

[25] O. Goldreich and Y. Oren, *Definitions and properties of zero-knowledge proof systems*, J. Cryptology, 7 (1994), pp. 1–32.

[26] S. Goldwasser and M. Bellare, *Lecture Notes on Cryptography*, http://www.cs.ucsd.edu/users/mihir/papers/gb.html (August 2001).

[27] S. Goldwasser, S. Micali, and C. Rackoff, *The knowledge complexity of interactive proof systems*, SIAM J. Comput., 18 (1989), pp. 186–208.

[28] S. Goldwasser and M. Sipser, *Private coins versus public coins in interactive proof systems*, Adv. Comput. Res., 5 (1989), pp. 73–90.

[29] D. Gutfreund, R. Shaltiel, and A. Ta-Shma, *Uniform hardness versus randomness tradeoffs for Arthur-Merlin games*, Comput. Complexity, 12 (2003), pp. 85–130.

[30] J. Hastad, R. Impagliazzo, L. A. Levin, and M. Luby, *A pseudorandom generator from any one-way function*, SIAM J. Comput., 28 (1999), pp. 1364–1396.

[31] R. Impagliazzo, V. Kabanets, and A. Wigderson, *In search of an easy witness: Exponential time vs. probabilistic polynomial time*, J. Comput. System Sci., 65 (2002), pp. 672–694.

[32] R. Impagliazzo and M. Luby, *One-way functions are essential for complexity based cryptography*, in Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science, Research Triangle Park, NC, 1989, pp. 230–235.

[33] R. Impagliazzo and A. Wigderson, *$P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma*, in Proceedings of the 29th Annual ACM Symposium on Theory of Computing, 1997, pp. 220–229.

[34] Y. T. Kalai, *Smooth projective hashing and two-message oblivious transfer*, in Proceedings of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2005), Lecture Notes in Comput. Sci. 3494, Springer, Berlin, 2005, pp. 78–95.

[35] A. R. Klivans and D. van Melkebeek, *Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses*, SIAM J. Comput., 31 (2002), pp. 1501–1526.

[36] L. A. Levin, *One-way functions and pseudorandom generators*, Combinatorica, 7 (1987), pp. 357–363.

[37] G. L. Miller, *Riemann's hypothesis and tests for primality*, J. Comput. System Sci., 13 (1976), pp. 300–317.

[38] P. B. Miltersen and N. V. Vinodchandran, *Derandomizing Arthur-Merlin games using*

hitting sets, Comput. Complexity, 14 (2005), pp. 256–279.

[39] M. Naor, *Bit commitment using pseudorandomness*, J. Cryptology, 4 (1991), pp. 151–158.

[40] M. Naor and B. Pinkas, *Efficient oblivious transfer protocols*, in Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-01), New York, 2001, pp. 448–457.

[41] N. Nisan and A. Wigderson, *Hardness vs. randomness*, J. Comput. System Sci., 49 (1994), pp. 149–167.

[42] M. Rabin, *Digitalized signatures and public-key functions as intractable as factorization*, Technical report MIT/LCS/TR-212, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1979.

[43] A. A. Razborov and S. Rudich, *Natural proofs*, J. Comput. System Sci., 55 (1997), pp. 24–35.

[44] R. L. Rivest, A. Shamir, and L. M. Adleman, *A method for obtaining digital signatures and public key cryptosystems*, Commun. ACM, 21 (1978), pp. 120–126.

[45] S. Rudich, *Super-bits, demi-bits, and* $\widetilde{\mathrm{NP}}$*/qpoly-natural proofs*, in Proceedings of the 1st International Workshop on Randomization and Approximation Techniques in Computer Science, Springer, Berlin, 1997, pp. 85–93.

[46] R. Shaltiel and C. Umans, *Pseudorandomness for approximate counting and sampling*, in Proceedings of the IEEE Conference on Computational Complexity, 2005, pp. 212–226.

[47] R. Shaltiel and C. Umans, *Simple extractors for all min-entropies and a new pseudorandom generator*, J. ACM, 52 (2005), pp. 172–216.

[48] L. Trevisan, *Extractors and pseudorandom generators*, J. ACM, 48 (2001), pp. 860–879.

[49] L. G. Valiant, *A theory of the learnable*, Commun. ACM, 27 (1984), pp. 1134–1142.

[50] A. C. Yao, *Theory and applications of trapdoor functions*, in Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science, 1982, pp. 80–91.

SIAM J. COMPUT.
Vol. 37, No. 2, pp. 401–412

A $\frac{3}{2}$-APPROXIMATION ALGORITHM FOR SCHEDULING INDEPENDENT MONOTONIC MALLEABLE TASKS*

GREGORY MOUNIE†, CHRISTOPHE RAPINE‡, AND DENIS TRYSTRAM†

Abstract. A *malleable task* is a computational unit that may be executed on any arbitrary number of processors, whose execution time depends on the amount of resources allotted to it. This paper presents a new approach for scheduling a set of independent malleable tasks which leads to a worst case guarantee of $\frac{3}{2} + \varepsilon$ for the minimization of the parallel execution time for any fixed $\varepsilon > 0$. The main idea of this approach is to focus on the determination of a good allotment and then to solve the resulting problem with a fixed number of processors by a simple scheduling algorithm. The first phase is based on a dual approximation technique where the allotment problem is expressed as a knapsack problem for partitioning the set of tasks into two shelves of respective heights 1 and $\frac{1}{2}$.

Key words. scheduling, malleable tasks, polynomial approximation, performance guarantee

AMS subject classifications. 90B35, 68W25, 68W40, 68Q85

DOI. 10.1137/S0097539701385995

1. Introduction.

The implementation of applications on parallel and distributed systems requires sophisticated algorithms for scheduling the tasks of the parallel programs. There exists a very large literature addressing this problem. It corresponds to determining a date for each task to start its execution together with a processor location. Usually, the tasks correspond to indivisible pieces of the application that are executed sequentially on a processor. The standard communication model for scheduling the tasks of a parallel program is the delay model introduced by Rayward-Smith [21] for UET-UCT (unit execution times and unit communication times) task graphs and extended by Papadimitriou and Yannakakis [18]. In this model, the communications between tasks executed on different processors are considered explicitly by the time for transferring a message between them. The communication times between tasks within the same processor are neglected. The scheduling UET-UCT problem is known to be $\mathcal{NP}$-hard in the strong sense [21], and it is not approximable within a factor of 5/4 of the optimum by any polynomial algorithm [10], unless $\mathcal{P} = \mathcal{NP}$. The best known approximation result is due to Hanen and Munier [8], whose algorithm is within a factor of 7/3 of the optimum for small communication delays. Among the various possible approaches, the most commonly used is to consider the tasks of the program at the finest level of granularity and apply some adequate clustering heuristics to reduce the relative communication overhead [22, 6, 17]. The main drawback of such an approach is that the communications are taken into account explicitly: they are expressed assuming a model of the underlying architecture of the system. A good alternative is to consider the *malleable tasks* model (denoted MT), where the communication times are considered implicitly by a function representing the parallel execution time with the penalty due to the management of the parallelism (communication, synchronization, etc.). A malleable task is a computational unit which may

*Received by the editors March 5, 2001; accepted for publication (in revised form) December 8, 2006; published electronically May 16, 2007.
http://www.siam.org/journals/sicomp/37-2/38599.html

†ID-IMAG, 51 avenue Jean Kuntzman, 38 330 Montbonnot Saint Martin, France (Denis.Trystram@imag.fr, Gregory.Mounie@imag.fr).

‡GILCO-INPG, 46 avenue Felix Viallet, 38 031 Grenoble cedex, France (Christophe.Rapine@gilco.inpg.fr).

be executed on several processors with a running time that depends on the number of processors allotted to it.

In this paper, we are interested in scheduling a set of n independent malleable tasks on a multiprocessor system composed of m identical processors. An instance of the problem is a set $\mathcal{T} = \{T_1, \dots, T_n\}$ of tasks, together with a set of n functions $t_i : p \rightarrow t_{i,p}$ which represents the processing time of task T_i when executed on p processors. A solution (*scheduling*) consists in finding for each task T_i a starting time st_i and a subset $\mathcal{P}_i$ of the processors to execute it, with the following constraints.

- Task T_i starts its execution simultaneously on all the processors of $\mathcal{P}_i$ and occupies them without interruption until its completion time $C_i = st_i + t_{i,|\mathcal{P}_i|}$.
- A processor executes at most one task at a time.

A task will be represented as a rectangle in the Gantt chart. This study is restricted to algorithms that provide consecutive processors. It is clear that the optimal allotment could use processors that are not consecutive. However, since we make use of a lower bound of the true optimum in our proof of the approximation bound, we see that this restriction does not have a substantial impact on what results can be achieved.

The objective is to minimize the *makespan* defined as the maximum completion time over all the tasks. Our main contribution is to propose a new method for scheduling independent monotonic malleable tasks. The analysis leads to a performance guarantee of $\frac{3}{2} + \varepsilon$ for any constant $\varepsilon > 0$, in time $\mathcal{O}(nm \log(n/\varepsilon))$. This bound improves all existing practical results for solving this problem. Such a method should be used as a basis for other scheduling problems like MT with precedence constraints or multiobjective scheduling analysis.

The organization of this paper is the following: we first briefly survey related work and recall the model of MT and its main properties. Then, we discuss the principle of our approach and present the algorithm and analyze its performance guarantee.

2. Preliminaries on malleable tasks.

2.1. Related work. The problem of scheduling independent malleable tasks has been extensively studied in the last decade. Among other reasons, interest in this problem was motivated by the problem of scheduling jobs in batch processing. Classical scheduling problems (i.e., with sequential tasks) are a particular case of the MT scheduling, and hence their complexity results apply directly to MT problems. It implies that scheduling independent MT is an $\mathcal{NP}$-hard problem [5], in the ordinary sense if m is fixed. Du and Leung [4] studied more precisely the complexity for MT scheduling problems, establishing that the problem with arbitrary precedence constraints is strongly $\mathcal{NP}$-hard for 2 processors, and scheduling independent MT is strongly $\mathcal{NP}$-hard for 5 processors.

Srinivasa Prasanna and Musicus [20] developed an approach based on optimal control theory for a continuous version of malleable tasks, leading to optimal solution assuming the same particular parallel time function for all the tasks.

Jansen and Porkolab [11] proposed a polynomial approximation scheme based on a linear programming formulation for scheduling independent malleable tasks. The complexity of the scheme, although linear in the number of tasks, is high independently of the accuracy of the approximation due to an exponential factor in the number of processors. Thus, even though the result is of great theoretical interest, this algorithm cannot be considered for a practical use.

We are interested in efficient, low complexity heuristics with a good performance guarantee. Most previous work is based on a two-phase approach proposed by Turek, Wolf, and Yu [24]. The basic idea is to select first an allotment (the number of processors allotted to each task) and in a second step to solve the resulting nonmalleable scheduling problem, which is a classical scheduling problem of multiprocessor tasks. As far as the makespan criterion is concerned, this problem is related to a 2-dimensional strip-packing problem [1, 3, 12] for independent tasks. It is clear that applying an approximation of guarantee λ for the nonmalleable problem on the allotment of an optimal solution provides the same guarantee λ for the malleable problem if ever an optimal allotment can be found. Two complementary ways for solving the problem have been proposed, focusing either on the allotment (first phase) or on the scheduling (second phase).

- Turek, Wolf, and Yu proposed a polynomial selection algorithm for the allotment problem such that any λ-approximation algorithm of complexity $\mathcal{O}(f(n,m))$ for the nonmalleable (multiprocessor) problem can be adapted into a λ-approximation algorithm of complexity $\mathcal{O}(mnf(n,m))$ for the malleable problem. Ludwig [13, 14] improved the complexity of the allotment selection in the special case of monotonic tasks. Based on this result and on the 2-dimensional strip-packing algorithm of guarantee 2 proposed by Steinberg [23], he presented a 2-approximation algorithm for scheduling independent MT. The power of this approach is also its main limitation: any improvement in the approximation of the strip-packing problem directly applies to the MT problem, but the performance guarantee of the approach is limited by the best known result for strip-packing.
- The other way corresponds to choosing an allotment such that the resulting nonmalleable problem is not a general instance of strip-packing, and hence better specific approximation algorithms can be applied. Using Knapsack as an auxiliary problem for the selection of the allotment, this technique leads to a $(\sqrt{3}+\varepsilon)$-approximation for monotonic tasks [16].

We focus in this paper on the second approach and show how a $(\frac{3}{2}+\varepsilon)$-approximation algorithm can be obtained for any $\varepsilon > 0$. The basic idea is to determine an allotment such that the tasks can be partitioned into two shelves of respective heights d and $d/2$ for some deadline d to be determined.

2.2. Notation and basic properties. The aim of this work is to construct an MT schedule for a set of n independent malleable tasks that minimizes the maximum completion time over all the m processors. Recall that we assume that a processor can compute only one task at a time (no time sharing) and that the number of processors dedicated to a task remains constant during all its execution. In addition we are looking for nonpreemptive schedules with *contiguous* allocation, which means that for each task the set of the subscripts of the processors allotted to it is an interval of $[1,m]$. Their performance guarantee is established with respect to an optimal solution, which may be contiguous or not.

2.2.1. Monotonic assumptions. We define the *work function* w_i of a task T_i, which corresponds to its computational area in the Gantt chart representation of a schedule, as $w_i : \ p \mapsto w_{i,p} = p \times t_{i,p}$ for $p \leq m$. According to the usual behavior of parallel programs, we will assume that the tasks are *monotonic:* allocating more processors to a task decreases its execution time and increases its work.

DEFINITION 2.1 (monotony).

- *The time monotony is achieved by a set of tasks $\mathcal{T}$ when t_i is a decreasing function for any task T_i.*
- *The work monotony is achieved by a set of tasks $\mathcal{T}$ when w_i is an increasing function for any task T_i.*

A set of tasks is monotonic if the two previous conditions are fulfilled.

Notice that an instance of the MT problem can always be transformed to fulfill the time monotony property, replacing the functions t_i by $t'_i: \ p \mapsto min\{t_{i,q} | q = 1, \ldots, p\}$. This transformation does not affect the optimal solution of the scheduling.

Due to cache effects or scheduling anomalies described by Graham [7], the work monotony cannot be asserted for all the applications. However, it is a quite reasonable hypothesis, which is expected for most large actual parallel applications, mainly due to the communication overhead. From the parallel computing point of view, this monotonic assumption may be interpreted by the well-known Brent's lemma [2], which states that the parallel execution of a task achieves some speedup if it is large enough, but does not lead to superlinear speedups.

We give below one useful definition for the presentation of our algorithm, together with two basic properties implied by the monotonic behavior of the tasks.

Definition 2.2 (canonical number of processors). *Given a real number h, we define for each task T_i its canonical number of processors $\gamma(i,h)$ as the minimal number of processors needed to execute task T_i in time at most h. If T_i cannot be executed in time less than h on m processors, we set by convention $\gamma(i,h) = +\infty$.*

Notice that if the set of tasks is monotonic, the canonical number of processors can be found in time $\mathcal{O}(\log m)$ by bisection search. In addition $w_{i,\gamma(i,h)}$ is also the minimal work area needed to execute T_i in time less than h.

Property 1. *Given a real number h, if $\gamma(i,h) < +\infty$, the execution time of task T_i on its canonical number of processors satisfies the inequality*

$$h \ \geq \ t_{i,\gamma(i,h)} \ > \ \frac{\gamma(i,h) - 1}{\gamma(i,h)} \ h.$$

Proof. For short let us denote by p the canonical number of processors of task T_i for the given deadline h. If $p = 1$, the inequality is clearly satisfied. Otherwise the monotonic behavior of the tasks implies that $w_{i,p} \geq w_{i,p-1}$, i.e., $p \times t_{i,p} \geq (p-1) \times t_{i,p-1}$. By definition of the canonical number of processors, $t_{i,p-1} > h$, which proves Property 1. □

As a corollary, if the canonical number of processors is at least 2 for a task, we have the following simplified property.

Property 2. *Given a real number h, if $\gamma(i,h) \in [2, m]$, we have*

$$2t_{i,\gamma(i,h)} \ \geq \ t_{i,\gamma(i,h)-1} \ > \ h \ \geq \ t_{i,\gamma(i,h)} \ > \ \frac{1}{2} \ h.$$

3. Description of the scheduling algorithm.

3.1. Principle. The principle of the algorithm is to use the dual approximation technique [9]. A λ-dual approximation algorithm for the MT scheduling problem takes a real number d as an input and

- either delivers a schedule of length at most λd, or
- answers correctly that there exists no schedule of length at most d.

Ludwig and Tiwari [14] proposed a lower bound ω that can be computed in time $\mathcal{O}(mn \log n)$ such that the optimal makespan d^* verifies $\omega \leq d^* \leq 2\omega$. Hence a λ-dual approximation running in time $f(n,m)$ can be converted, by bisection search, in a $\lambda(1+\varepsilon)$-approximation running in time $\mathcal{O}(mn \log n + \log(1/\varepsilon) f(n,m))$ for any $\varepsilon > 0$.

We are interested in this article in finding a 3/2-dual approximation. Let d be the current real number input for our dual approximation. In the following we assert that an MT schedule of length lower than d exists: thus we have to show how it is possible to build a schedule of length at most $3d/2$. The idea of the algorithm is to partition the set of tasks into two shelves, one of height d and the other of height $d/2$. Since the tasks are independent in both shelves, the scheduling strategy is straightforward after the allotment of the tasks has been determined and yields directly a solution of length at most $3d/2$. The main problem we face is to choose the tasks in each shelf in order to obtain a feasible solution. The way to determine the partition will be described in section 3.5.

3.2. Structure of an optimal schedule. To take advantage of the dual approximation paradigm, we have to make explicit the consequences of our assumption that there exists a schedule of length at most d. We state below some straightforward properties of such a schedule. They should give the insight for the construction of our solution.

Remark 1. In an optimal solution, the execution time of each task is at most d, and the total work is at most md.

Remark 2. In an optimal solution, if there exist two successive tasks (i.e., tasks executed successively on a common processor), at least one of these tasks has an execution time at most $d/2$.

The basic idea of the solution that we propose comes from the analysis of the shape of an optimal schedule. From Remark 2 the tasks whose execution times are strictly greater than $d/2$ do not use more than m processors, and hence can be executed concurrently. The other tasks can be executed in time at most $d/2$. Thus, we are looking for a schedule in two shelves: S_1 of height d and S_2 of height $d/2$.

3.3. Algorithm. Starting from the idea of constructing a schedule in two shelves, we sketch below the main steps of the dual approximation algorithm; the full details follow in the next sections. The input consists in a set of tasks ($\mathcal{T}$), the number of processors (m), and a guess of the optimal makespan (d).

```
$3/2$ Dual Approximation ($\mathcal{T}$, $m$, $d$)
   1. Determine the set $\mathcal{T_S} = \{T_i | t_{i,1} \leq d/2\}$ of tasks whose
      sequential time is at most $d/2$.  Compute $\mathcal{W_S} = \sum_{T_i \in \mathcal{T_S}} t_{i,1}$.
   2. Search for an allotment for $\mathcal{T} \backslash \mathcal{T_S}$ such that
         • the total work is at most $md - \mathcal{W_S}$,
         • each task has an execution time at most $d$,
         • the tasks whose execution times are strictly greater
           than $d/2$ require at most $m$ processors altogether.
      If such an allotment does not exist, return NO.
   3. Convert the allotment into a feasible 2-Shelves schedule
      of length at most $3d/2$ by calling algorithm BuildFeasible
      (see section 3.6).
   4. Insert the set of tasks $\mathcal{T_S}$ in the schedule using a greedy
      algorithm (see section 3.4) and return YES and the
      schedule.
```

Sections 3.4–3.6 detail the implementation of these steps. Steps 1 and 4, presented together in section 3.4, simply mean that we can forget about sufficiently small tasks to build the 2-Shelves schedule. They can be implemented in time complexity $\mathcal{O}(nm)$.

Step 2 is the critical point of the algorithm. It consists in choosing a first allotment that fulfills some properties to be a good candidate for the 2-Shelves schedule. We need

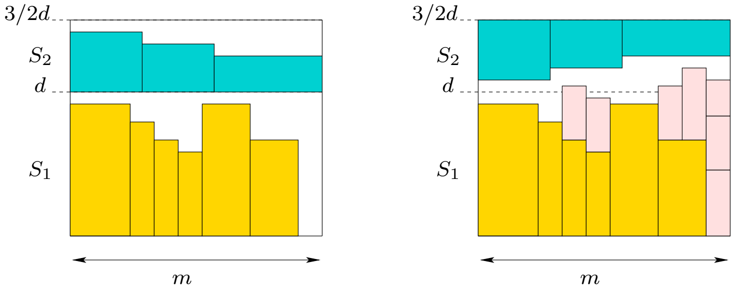


FIG. 3.1. *A* 2*-Shelves schedule for* $\mathcal{T}\backslash\mathcal{T}_{\mathcal{S}}$ *(left) and the final schedule after the insertion of tasks of* $\mathcal{T}_{\mathcal{S}}$ *(right).*

an oracle to either find such an allotment or deliver a certificate that no schedule of length at most d exists.

This oracle is implemented in time $\mathcal{O}(nm)$ using a knapsack formulation of the problem described in section 3.5.

Finally section 3.6 presents the algorithm `BuildFeasible` which applies several simple transformations to the initial allotment in order to build a feasible schedule with two shelves of respective heights d and $d/2$. Its time complexity is also in $\mathcal{O}(nm)$, which leads to an overall complexity in $\mathcal{O}(nm)$ for the dual approximation scheme.

3.4. Forgetting about small tasks. Recall that we are looking for an MT schedule of length at most $3d/2$, assuming that there exists a schedule of length at most d. Since we are interested in an approximation of the optimal solution, we can "forget" about some small tasks which do not affect the final performance of the algorithm. These small tasks are the set $\mathcal{T}_{\mathcal{S}}$ of the tasks whose sequential execution time is at most $d/2$. Let us denote by $\mathcal{W}_{\mathcal{S}}$ the sum of the execution times of $\mathcal{T}_{\mathcal{S}}$. We remark that $\mathcal{W}_{\mathcal{S}}$ is a lower bound of the work area of execution of $\mathcal{T}_{\mathcal{S}}$ in any feasible schedule.

LEMMA 3.1. *If there exists a* 2*-Shelves schedule of length* $3d/2$ *for* $\mathcal{T}\backslash\mathcal{T}_{\mathcal{S}}$ *with a work area at most* $md - \mathcal{W}_{\mathcal{S}}$*, then an MT schedule of length at most* $3d/2$ *can be derived for* $\mathcal{T}$ *in time* $\mathcal{O}(nm)$.

Proof. Consider a 2-Shelves schedule composed of shelves S_1 and S_2. We can modify the starting time of the tasks of S_2, which is currently d, to require that they all finish exactly at time $3d/2$. It creates on each processor an idle time interval between the completion of the tasks of S_1 and the starting of the tasks of S_2. We define the *load* of a processor as the sum of the execution times of the tasks computed by it. By definition the load is equal to $3d/2$ minus the length of the idle time interval on the processor. Now consider the following algorithm to schedule the tasks of $\mathcal{T}_{\mathcal{S}}$ (see Figure 3.1):

- Consider the tasks in an arbitrary order $\mathcal{T}_{\mathcal{S}} = \{T_1, \ldots, T_k\}$.
- Allocate task T_i to the least loaded processor, at the earliest possible date. Update its load.

The only problem that may occur is that a task T_i cannot be completed before the tasks of S_2. But at each step, the least loaded processor has a load at most d; otherwise it would contradict the fact that the total work area of the tasks is bounded by md. Hence, the idle time interval on this processor has a length at least $d/2$ and can contain the task T_i. □

3.5. Partitioning the tasks into two shelves. In this section, we detail how to fill both shelves S_1 and S_2 by specifying an initial allotment of processors for the tasks. According to Lemma 3.1, we assume that only tasks with a sequential execution time strictly greater than $d/2$ remain in $\mathcal{T}$.

In order to obtain a 2-Shelves schedule, we look for an allotment satisfying the following three constraints:

(C1) The total work area of the allotment is at most $\mathcal{W} = md - \mathcal{W}_{\mathcal{S}}$.

(C2) The set $\mathcal{T}_1$ of tasks with an execution time strictly greater than $d/2$ in the allotment uses a total of at most m processors. These tasks are intended to be scheduled in S_1.

(C3) The set $\mathcal{T}_2$ of tasks with an execution time at most $d/2$ in the allotment uses a total of at most m processors. These tasks are intended to be scheduled in S_2.

Such an allotment clearly defines a 2-Shelves schedule of length at most $3d/2$ which would allow us to build a solution for the MT problem according to Lemma 3.1. Unfortunately, we have no certitude on the existence of such an allotment. Therefore, we relax the allotment problem looking for a solution which verifies only constraints (C1) and (C2), but might violate (C3).

Due to the monotonic assumption, we have only two allotments to consider for a task. If it is selected to belong to $\mathcal{T}_1$, clearly $\gamma(i,d)$ is a dominant allotment; otherwise $\gamma(i,d/2)$ is. According to Remark 1, we note that $\gamma(i,d)$ is at most m for all the tasks. To determine if such a relaxed allotment exists, we can solve the following optimization problem:

$$\text{find } \; W^* = \min_{\mathcal{T}_1 \subseteq \mathcal{T}} \left(\sum_{i \in \mathcal{T}_1} w_{i,\gamma(i,d)} + \sum_{i \notin \mathcal{T}_1} w_{i,\gamma(i,d/2)} \right)$$

$$\text{under the constraint } \; \sum_{i \in \mathcal{T}_1} \gamma(i,d) \le m.$$

This problem is in fact a well-known knapsack problem. Let us recall it briefly: given a set of n items, each one associated to an integral weight ω_i and a profit v_i, and a knapsack with a total weight capacity W, find a subset of the tasks which can be contained by the knapsack with the maximal profit. This problem is $\mathcal{NP}$-hard [5]; however, it admits [15, 19] a pseudopolynomial algorithm, using dynamic programming, that solves it exactly in time complexity in $\mathcal{O}(nW)$.

Here, imagine that all the tasks are initially allotted to their canonical number of processors to respect the $d/2$ threshold. The problem is then to determine the set $\mathcal{T}_1$ using at most m processors such that the total weight is minimal. Hence, the profit of an item-task will correspond to the work saving obtained by executing the task just to respect the threshold d instead of $d/2$, i.e., $v_i = w_{i,\gamma(i,d/2)} - w_{i,\gamma(i,d)}$. The weight of an item-task will be its canonical number of processors needed to respect the threshold d, $\omega_i = \gamma(i,d)$, while the capacity W of the knapsack is m. Using this notation, the problem can be rewritten as the following knapsack problem:

$$\text{find } W^* = \sum_{i \in \mathcal{T}} w_{i,\gamma(i,d/2)} - \max_{\mathcal{T}_1 \subseteq \mathcal{T}} \sum_{i \in \mathcal{T}_1} v_i$$

$$\text{under the constraint } \; \sum_{i \in \mathcal{T}_1} \omega_i \le m.$$

If the work area W^* is greater than $\mathcal{W} = md - \mathcal{W}_{\mathcal{S}}$, then there exists no solution with a makespan at most d, and the algorithm answers "NO" to the dual approximation. Otherwise, we will describe in detail in the next section how to construct a feasible solution with a makespan at most $3d/2$. Lemma 3.2 establishes the correctness of this dual approximation.

Lemma 3.2. *Assuming that there exists a schedule of length at most d, the knapsack formulation of the problem delivers an allotment satisfying constraints* (C1) *and* (C2) *in time $\mathcal{O}(nm)$.*

Proof. Consider an optimal schedule. As already noted, the total work of tasks of $\mathcal{T}_{\mathcal{S}}$ in this schedule is at least $\mathcal{W}_{\mathcal{S}}$; hence the remaining tasks occupy an area bounded by $\mathcal{W} = md - \mathcal{W}_{\mathcal{S}}$. The allotment of the optimal solution partitions these tasks into two sets $\mathcal{T}_1'$ and $\mathcal{T}_2'$, where $\mathcal{T}_1'$ groups the tasks with execution time strictly greater than $d/2$. By definition any task T_i is allotted to at least $\gamma(i, d)$ processors if it belongs to $\mathcal{T}_1'$, and at least $\gamma(i, d/2)$ processors if it belongs to $\mathcal{T}_2'$. Finally, as a direct consequence of Remark 2, the tasks of $\mathcal{T}_1'$ use at most m processors. It follows that $\mathcal{T}_1'$ is a feasible solution for the knapsack procedure, with a resulting work area at most $\mathcal{W}$. By definition it implies that the optimum W^* of the knapsack procedure is at most $\mathcal{W}$. □

3.6. Satisfying constraint (C3). Starting from the allotment found by the knapsack procedure, we can construct a solution with the tasks of $\mathcal{T}_1$ in S_1 and the others, $\mathcal{T}_2 = \mathcal{T} \backslash \mathcal{T}_1$, in S_2. No more than m processors are used to schedule the tasks in S_1, but it may happen that more than m processors are needed in S_2. We then apply three possible transformations that will reduce this number to less than m. These transformations are applied until the resulting schedule becomes a feasible solution on m processors. These transformations modify the shape of the 2-Shelves solution we are looking for, creating a new area S_0 whose processors are continuously busy in the time interval $[0, d]$, as depicted in Figure 3.2. The three transformations are the following (note that these transformations can be applied in any order):

(1) If a task T in S_1 has an execution time at most $3d/4$ and is allotted to $p > 1$ processors, allocate T to $p - 1$ processors in S_0.

(2) If T and T' in S_1 have an execution time lower than $3d/4$ and are each allotted to 1 processor, allocate T and T' to the same processor in S_0. A special case happens if T is the only remaining sequential task of execution time at most $3d/4$.

(3) Let q denote the number of idle processors in S_1. If there exists a task T_i in S_2 whose execution time on q processors is bounded by $3d/2$, allocate T_i on $\gamma(i, 3d/2)$ processors. According to the resulting execution time, T_i is either scheduled in S_0 if $t \geq d$ or in S_1 otherwise.

Finally, the algorithm to build a feasible solution of length at most $3d/2$ is the following:

```
Algorithm BuildFeasible
    • Start from the solution delivered by the Knapsack,
      $S_0 = \emptyset$, $S_1 = \mathcal{T}_1$, $S_2 = \mathcal{T}_2$.
    • While the solution is not feasible
          apply one of the transformations (1), (2), or (3).
```

The end of the section is devoted to the proof of Lemma 3.3.

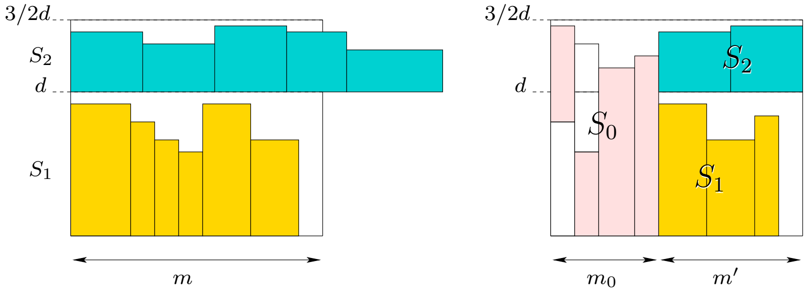


Fig. 3.2. *The* 2*-Shelves schedule obtained from the allotment phase (left) and the final schedule given by* `BuildFeasible` *(right) with the new area* S_0.

Lemma 3.3. *The algorithm* `BuildFeasible` *delivers a feasible schedule of length at most* $3d/2$ *in time complexity* $\mathcal{O}(nm)$.

Notice that the transformations ensure by construction that the makespan remains bounded by $3d/2$ at each step of the algorithm (for transformation (1), this is a direct consequence of Property 2). In addition, a transformation never increases the number of processors allotted to a task. Due to the monotony, it asserts that the total work area of the schedule remains bounded by $\mathcal{W} = md - \mathcal{W}_{\mathcal{S}}$ at any step of `BuildFeasible`.

Let m_0 be the number of processors used to schedule the tasks of set S_0 in the final solution. We denote by $m' = m - m_0$ the remaining processors for the 2-Shelves schedule composed of S_1 and S_2. By construction any processor in S_0 completes after deadline d, which implies a work area greater than $m_0 d$. Since the total work area is bounded by $\mathcal{W}$, it is straightforward to remark that the total work area of tasks in S_1 and S_2 is bounded by $m'd - \mathcal{W}_{\mathcal{S}}$. In addition the set S_1 requires less than m' processors for the concurrent execution of its tasks. Hence, to prove Lemma 3.3, we are going to show that while the second shelf S_2 requires more than m' processors, one of the three transformations can be applied. It is clear that the schedule restricted to S_1 and S_2 on m' processors, if feasible, verifies the conditions of application of Lemma 3.1. Thus, we can conclude that the algorithm is a 3/2-dual approximation.

3.6.1. Algorithm `BuildFeasible` delivers a feasible schedule. Suppose that none of the transformations can be applied to the current schedule. We have to prove that this solution is feasible, i.e., requires at most m processors, which is equivalent by construction to proving that S_2 requires at most m' processors.

Let q be the number of idle processors in the first shelf S_1. Assume for the sake of contradiction that the second shelf S_2 requires $m_2 > m'$ processors. We have the following structure for the current schedule:

1. The total work area of tasks in $S_1 \cup S_2$ is bounded by $\mathcal{W}' = m'd - \mathcal{W}_{\mathcal{S}}$.
2. Any task in S_1 has a duration strictly greater than $3d/4$, except possibly one sequential task whose execution time can be in the range $]d/2, 3d/4]$.
3. Any task in S_2 has a duration strictly greater than $d/4$.
4. Any task in S_2 has a work area greater than $3qd/2$, and hence is allotted to at least $3q + 1$ processors.

The second point is a direct consequence of the fact that neither transformation (1) nor (2) can be applied. The third point is a corollary of Property 2: since any task of $\mathcal{T} \backslash \mathcal{T}_{\mathcal{S}}$ has a sequential execution time strictly greater than $d/2$, all the tasks in S_2 are allotted to $\gamma(i, d/2) \geq 2$ processors. The last point comes from the fact that, due to transformation (3), any task in S_2 has a duration greater than $3d/2$ when allotted to q processors. Due to the monotonic assumption, the task's current work is at least its work on q processors, which is strictly greater than $q(3d/2)$. In particular, we have $3qt_{i,3q} \geq w_{i,q} > q(3d/2)$, which implies that the time duration on $3q$ processors is strictly greater than $d/2$. Thus by definition $\gamma(i, d/2) > 3q$.

To obtain a contradiction to the assumption that the current schedule is not a feasible solution, we will derive some lower bounds of the work area in S_1 and S_2 which will contradict the fact that their sum is bounded by $m'd$. We start by giving the lower bound used for the tasks in S_1, together with a very simple first lower bound for S_2.

Lemma 3.4. *If the schedule is not feasible, the overall work area $\mathcal{W}_1$ of S_1 is greater than $3d(m'-q)/4$, while the overall work area $\mathcal{W}_2$ of S_2 is at least $d(m'+1)/4$.*

Proof. The lower bound on $\mathcal{W}_2$ is straightforward, since any task in S_2 has an execution time strictly greater than $d/4$, and the tasks of S_2 use at least $m' + 1$ processors. The same argument holds for $\mathcal{W}_1$ if no sequential task of duration at most $3d/4$ exists in the shelf. Otherwise let T be this unique task, with a sequential time t in the range $]d/2, 3d/4]$.

Let us first establish that T cannot be the only task scheduled in S_1. Indeed, assume for the sake of contradiction that it is the case. If there is no idle processor in S_1 ($q = 0$), we simply have $m' = 1$. Hence at least $\mathcal{W}_1 > d/2$ while the lower bound on S_2 can be rewritten as $\mathcal{W}_2 > d/2$. It contradicts the fact that $\mathcal{W}_1 + \mathcal{W}_2$ is bounded by $m'd = d$. If there exist some idle processors, then we have $q = m' - 1 > 0$. As S_2 contains at least one task, $\mathcal{W}_2 > 3qd/2 = 3(m' - 1)d/2$. We obtain $m'd > d/2 + 3(m' - 1)d/2 = (3m' - 2)d/2$, which implies $2 > m'$, contradicting $q > 0$.

Hence, at least another task T' is partially scheduled in S_1 together with T. Since transformation (2) cannot be applied, task T' has an execution time t' strictly greater than $3d/2-t$. Thus, considering the processor executing T' and the processor executing T, their average load is strictly greater than $3d/4$. Since any other non-idle processor is occupied by a task in S_1 with an execution time greater than $3d/4$, we obtain $\mathcal{W}_1 > 3(m' - q)/4$. □

To conclude that a nonfeasible schedule leads to a contradiction, we distinguish between two cases, depending on whether or not there exist some idle processors in S_1.

Case 1. Assume $q = 0$. In this case Lemma 3.4 leads directly to a contradiction. Indeed we have $m'd \geq \mathcal{W}_1 + \mathcal{W}_2 > 3m'd/4 + (m' + 1)d/4 > m'd$.

Case 2. Assume $q > 0$. We need a more accurate lower bound on $\mathcal{W}_2$ to complete the proof. Let k be the number of tasks in S_2. By construction we have

$$\mathcal{W}_2 = \sum_{i \in S_2} w_{i,\gamma(i,d/2)}.$$

We can express the work of each task in two different ways. First, using the fact that this work is greater than $3qd/2$, we obtain

$$\mathcal{W}_2 > \frac{3}{2}qdk. \tag{3.1}$$

Second, due to monotony, the work of each task T_i when allotted on fewer processors never increases: $w_{i,\gamma(i,d/2)} \geq w_{i,\gamma(i,d/2)-1}$. By definition, the execution time on $\gamma(i, d/2) - 1$ processors is strictly greater than $d/2$, which implies that $w_{i,\gamma(i,d/2)-1} > (\gamma(i, d/2) - 1)d/2$. We have

$$\mathcal{W}_2 \ > \ \left(\sum_{i \in S_2} \gamma\left(i, \frac{d}{2}\right) - k\right)\frac{d}{2} \ \geq \ \frac{1}{2}(m' + 1 - k)d. \tag{3.2}$$

Using the lower bound established in Lemma 3.4 on $\mathcal{W}_1$ we can rewrite the upper bound $m'd$ on the total work area as $W_2 < m'd/4 + 3qd/4$. Using (3.1), we obtain

$$6qk < m' + 3q \iff 3q(2k-1) < m'.$$

Using (3.2), we get

$$2(m' + 1 - k) < m' + 3q \iff m' < 3q + (2k-2).$$

By transitivity we obtain the following strict inequality:

$$3q(2k-1) < 3q + (2k-2) \iff 3q(2k-2) < (2k-2) \iff (3q-1)(k-1) < 0.$$

But both k and q are greater than 1, which contradicts the previous inequality. This concludes the proof of Lemma 3.3 by contradiction.

3.6.2. Time complexity of `BuildFeasible`. We finally establish the time complexity of the algorithm. Each of the three transformations either moves a task from S_2 to S_1 or S_0, or from S_1 to S_0. Hence at most two transformations can be applied to any task. Let N be the number of tasks to deal with in our problem, after the elimination of the "small" tasks of $\mathcal{T}_\mathcal{S}$. If we look at the time complexity of each of the three transformations at a step of the algorithm, we have the following:

- The first two transformations can be implemented in time complexity $\mathcal{O}(N)$ by a simple scan of the tasks in S_1.
- Transformation (3) can also be implemented in time $\mathcal{O}(N)$ scanning the tasks of S_2. The determination of $\gamma(i, 3d/2)$ for the elected task T_i can be computed in time $\mathcal{O}(\log m)$ by a bisection search.

Since at most $2N$ transformations can be applied, algorithm `BuildFeasible` has an overall complexity in $\mathcal{O}(N^2 + N \log m)$. To obtain a time complexity in $\mathcal{O}(nm)$, simply notice that N is bounded by both n and $2m$: indeed any of the N tasks has a sequential execution time greater than $d/2$ for a total work bounded by md. Monotony implies that $N \leq 2m$.

4. Conclusion.

We have presented in this paper a new algorithm for scheduling a set of independent malleable tasks. It improves significantly the best bound known at this time, with a performance guarantee of $\frac{3}{2} + \varepsilon$ for any $\varepsilon > 0$ in time complexity $\mathcal{O}(nm \log(n/\varepsilon))$. The basic idea was to focus on the first phase of allotment using a knapsack formulation of the problem.

The natural continuation of this work is to study the scheduling of other structures of precedence graphs with malleable tasks. We believe that a similar analysis in two phases with a sophisticated allotment algorithm should lead to good approximation algorithms.

Another promising feature of MT is its intrinsic hierarchical behavior which should help in developing good scheduling algorithms for cluster computing. This issue is under investigation.

REFERENCES

[1] B. S. Baker, E. G. Coffman, Jr., and R. L. Rivest, *Orthogonal packings in two dimensions*, SIAM J. Comput., 9 (1980), pp. 846–855.

[2] R. P. Brent, *The parallel evaluations of general arithmetic expressions*, J. ACM, 21 (1974), pp. 201–206.

[3] E. G. Coffman, Jr., M. R. Garey, D. S. Johnson, and R. E. Tarjan, *Performance bounds for level-oriented two-dimensional packing algorithms*, SIAM J. Comput., 9 (1980), pp. 808–826.

[4] J. Du and J. Y-T. Leung *Complexity of scheduling parallel task systems*, SIAM J. Discrete Math., 2 (1989), pp. 473–487.

[5] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, New York, 1979.

[6] A. Gerasoulis and T. Yang, *PYRROS: Static scheduling and code generation for message passing multiprocessors*, in Proceedings of the 6th ACM International Conference on Supercomputing, ACM, 1992, pp. 428–437.

[7] R. L. Graham, *Bounds on multiprocessing timing anomalies*, SIAM J. Appl. Math., 17 (1969), pp. 416–429.

[8] C. Hanen and A. Munier, *An approximation algorithm for scheduling dependent tasks on m processors with small communication delays*, Discrete Appl. Math., 108 (2001), pp. 239–257.

[9] D. S. Hochbaum and D. B. Shmoys, *Using dual approximation algorithms for scheduling problems: Theoretical and practical results*, J. ACM, 34 (1987), pp. 144–162.

[10] J. Hoogeveen, J.-K. Lenstra, and B. Veltman, *Three, four, five, six, or the complexity of scheduling with communication delays*, Oper. Res. Lett., 16 (1994), pp. 129–137.

[11] K. Jansen and L. Porkolab, *Linear-time approximation schemes for scheduling malleable parallel tasks*, in Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 99), Baltimore, MD, 1999, pp. 490–498.

[12] C. Kenyon and E. Eric Rémila, *Approximate strip packing*, in 37th Annual Symposium on Foundations of Computer Science, (Burlington, VT 1996), IEEE Computer Society Press, Los Alamitos, CA, 1996, pp. 31–36.

[13] W. T. Ludwig, *Algorithms for Scheduling Malleable and Nonmalleable Parallel Tasks*, Ph.D. thesis, Department of Computer Sciences, University of Wisconsin-Madison, Madison, WI, 1995.

[14] W. Ludwig and P. Tiwari, *Scheduling malleable and nonmalleable parallel tasks*, in Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, D. D. Sleator, ed., Arlington, VA, 1994, pp. 167–176.

[15] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, Wiley, New York, 1990.

[16] G. Mounié, C. Rapine, and D. Trystram, *Efficient approximation algorithms for scheduling malleable tasks*, in Proceedings of the Eleventh ACM Symposium on Parallel Algorithms and Architectures (SPAA'99), ACM Press, New York, 1999, pp. 23–32.

[17] M. A. Palis, J.-C. Liou, and D. S. L. Wei, *Task clustering and scheduling for distributed memory parallel architectures*, IEEE Trans. Parallel and Distributed Systems, 7 (1996), pp. 46–55.

[18] C. H. Papadimitriou and M. Yannakakis, *Towards an architecture-independent analysis of parallel algorithms*, SIAM J. Comput., 19 (1990), pp. 322–328.

[19] C. H. Papadimitriou, *Computational Complexity*, Addison–Wesley, Reading, MA, 1994.

[20] G. N. Srinivasa Prasanna and B. R. Musicus, *Generalised multiprocessor scheduling using optimal control*, in Proceedings of the Third Annual ACM Symposium on Parallel Algorithms and Architectures, ACM Press, New York, 1991, pp. 216–228.

[21] V. J. Rayward-Smith, *UET scheduling with unit interprocessor communication delays*, Discrete Appl. Math., 18 (1987), pp. 55–71.

[22] V. Sarkar, *Partitioning and Scheduling Parallel Programs for Multiprocessors*, Pitman, London, 1989.

[23] A. Steinberg, *A strip-packing algorithm with absolute performance bound* 2, SIAM J. Comput., 26 (1997), pp. 401–409.

[24] J. Turek, J. Wolf, and P. Yu, *Approximate algorithms for scheduling parallelizable tasks*, in Proceedings of the Fourth Annual ACM Symposium on Parallel Algorithms and Architectures, 1992, pp. 323–332.

SIAM J. COMPUT.
Vol. 37, No. 2, pp. 413–424

QUANTUM ALGORITHMS FOR THE TRIANGLE PROBLEM*

FRÉDÉRIC MAGNIEZ†, MIKLOS SANTHA†, AND MARIO SZEGEDY‡

Abstract. We present two new quantum algorithms that either find a triangle (a copy of K_3) in an undirected graph G on n nodes, or reject if G is triangle free. The first algorithm uses combinatorial ideas with Grover Search and makes $\tilde{O}(n^{10/7})$ queries. The second algorithm uses $\tilde{O}(n^{13/10})$ queries and is based on a design concept of Ambainis [in *Proceedings of the* 45*th IEEE Symposium on Foundations of Computer Science*, 2004, pp. 22–31] that incorporates the benefits of quantum walks into Grover Search [L. Grover, in *Proceedings of the Twenty-Eighth ACM Symposium on Theory of Computing*, 1996, pp. 212–219]. The first algorithm uses only $O(\log n)$ qubits in its quantum subroutines, whereas the second one uses $O(n)$ qubits. The Triangle Problem was first treated in [H. Buhrman et al., *SIAM J. Comput.*, 34 (2005), pp. 1324–1330], where an algorithm with $O(n + \sqrt{nm})$ query complexity was presented, where m is the number of edges of G.

Key words. 05C85, 68R10, 68W99

AMS subject classifications. quantum algorithm, query complexity, triangle problem, quantum walk

DOI. 10.1137/050643684

1. Introduction.

Quantum computing is an extremely active research area (for introductions, see, e.g., [22, 20]), where a growing trend is the study of quantum query complexity. The quantum query model was implicitly introduced by Deutsch [15], Deutsch and Jozsa [16], Simon [25], and Grover [18], and explicitly by Beals et al. [9]. In this model, as in its classical counterpart, we pay for accessing the oracle (the black box), but unlike in the classical case, the machine can use the power of quantum parallelism to make queries in superpositions. While no significant lower bounds are known in quantum time complexity, the black box constraint sometimes enables us to prove such bounds in the query model.

For promise problems quantum query complexity indeed can be exponentially smaller than the randomized query complexity; a prominent example for that is the Hidden Subgroup Problem [25, 17]. On the other hand, Beals et al. [9] showed that for total functions the deterministic and the quantum query complexities are polynomially related. In this context, a large axis of research pioneered by Grover [18] was developed around search problems in unstructured, structured, or partially structured databases.

The classical query complexity of graph properties has made its fame through the notoriously hard evasiveness conjecture of Aanderaa and Rosenberg [24] which states that every nontrivial and monotone boolean function on graphs whose value remains invariant under the permutation of the nodes has deterministic query complexity exactly $\binom{n}{2}$, where n is the number of nodes of the input graph. Though this conjecture

*Received by the editors October 27, 2005; accepted for publication (in revised form) December 4, 2006; published electronically May 16, 2007. A preliminary version of this paper appeared in *Proceedings of the Sixteenth ACM-SIAM Symposium on Discrete Algorithms*, 2005, pp. 1109–1117.
http://www.siam.org/journals/sicomp/37-2/64368.html

†CNRS–LRI, UMR 8623 Université Paris–Sud, 91405 Orsay, France (magniez@lri.fr, santha@lri.fr). The research of this author was partially supported by the EU 5th and 6th framework programs RESQ IST-2001-37559, RAND-APX IST-1999-14036 and QAP IST, and by ACI Cryptologie CR/02 02 0040 and ACI Sécurité Informatique 03 511 grants of the French Research Ministry.

‡Department of Computer Science, Rutgers University, Piscataway, NJ 08854 (szegedy@cs.rutgers.edu). This author's research was supported by NSF grant 0105692 and the EU 5th framework program RESQ IST-2001-37559. The research was done while the author was visiting LRI.

is still open, an $\Omega(n^2)$ lower bound has been established by Rivest and Vuillemin [23]. In randomized bounded error complexity the general lower bounds are far from the conjectured $\Omega(n^2)$. The first nonlinear lower bound was shown by Yao [30]. For a long time Hajnal's $\Omega(n^{4/3})$ bound [19] was the best, until it was slightly improved in [13] to $\Omega(n^{4/3}\log^{1/3} n)$. The question of the quantum query complexity of graph properties was first raised in [11], where it is shown that in the exact case an $\Omega(n^2)$ lower bound still holds. In the bounded error quantum query model, the $\Omega(n^2)$ lower bound does not hold anymore in general. An $\Omega(n^{2/3}\log^{1/6} n)$ lower bound, first observed by Yao [31], can be obtained combining Ambainis' technique [4] with the above randomized lower bound.

We address the Triangle Problem in this setting. In a graph G, a complete subgraph on three vertices is called a *triangle*. In this write-up we study the following oracle problem:

Triangle
Oracle Input: The adjacency matrix f of a graph G on n nodes.
Output: A triangle if there is any, otherwise reject.

Triangle has been studied in various contexts, partly because of its relation to matrix multiplication [3]. Its quantum query complexity was first raised in [12], where the authors show that in the case of sparse graphs the trivial (that is, using Grover Search) $O(n^{3/2})$ upper bound can be improved. Their method breaks down when the graph has $\Theta(n^2)$ edges.

The quantum query complexity of Triangle as well as of many of its kins with small one-sided certificate size is notoriously hard to analyze, because one of the main lower bounding methods breaks down near the square root of the instance size [27, 21, 32, 26]: *If the* 1*-certificate size of a boolean function on N boolean variables is K, then even the most general variants* [8, Theorem 4], [5], [21] *of Ambainis' quantum adversary technique* [4] *can prove only a lower bound of* $\Omega(\sqrt{NK})$. Indeed only the $\Omega(n)$ lower bound is known for Triangle, which, because of the remark above, cannot be improved using any quantum adversary technique ($N = n^2$ and $K = 3$). Problems with small certificate complexity include various collision type problems such as the 2-1 Collision Problem and the Element Distinctness Problem. The first polynomial lower bound for the 2-1 Collision Problem was shown by Aaronson and Shi [1] using the polynomial method of Beals et al. [9]. For the Element Distinctness Problem, a randomized reduction from the 2-1 Collision Problem gives $\Omega(n^{2/3})$.

In this paper we present two different approaches that give rise to new upper bounds. First, using combinatorial ideas, we design an algorithm for Triangle (Theorem 3.5) whose quantum query complexity is $\tilde{O}(n^{10/7})$. Surprisingly, its quantum parts consist in only Grover Search subroutines. Indeed, Grover Search coupled with the Szemerédi lemma [28] already gives an $o(n^{3/2})$ bound. We exploit this fact using a simpler observation that leads to the $\tilde{O}(n^{10/7})$ bound. Moreover, our algorithm uses only small quantum memory, namely $O(\log n)$ qubits (and $O(n^2)$ classical bits). Then, we generalize the new elegant method used by Ambainis [6] for solving the Element Distinctness Problem in $O(n^{2/3})$, to solve a general Collision Problem by a dynamic quantum query algorithm (Theorem 4.1). The solution of the general Collision Problem will be used in our second algorithm for Triangle. As an intermediate step, we introduce the Graph Collision Problem, which is a variant of the Collision Problem, and solve it in $\tilde{O}(n^{2/3})$ query complexity (Theorem 4.4). Whereas a reduction of Triangle to the Element Distinctness Problem does not give a better algorithm than $O(n^{3/2})$, using a recursion of our dynamic version of Ambainis' method we prove the $\tilde{O}(n^{13/10})$ query complexity for Triangle (Theorem 4.5). We end by generalizing this result for finding the copy of any given graph (Theorem 4.6) and then for every graph property with small 1-certificates (Corollary 4.7).

2. Preliminaries.

2.1. Query model. In the query model of computation each query adds one to the complexity of an algorithm, but all other computations are free. The state of the computation is represented by three registers, the query register x, the answer register a, and the work register z. The computation takes place in the vector space spanned by all basis states $|x, a, z\rangle$. In the *quantum query model* the state of the computation is a complex combination of all basis states which has unit length in the norm l_2.

The query operation O_f maps the basis state $|x, a, z\rangle$ into the state $|x, a \oplus f(x), z\rangle$ (where $\oplus$ is bitwise XOR). Nonquery operations are independent of f. A *k-query algorithm* is a sequence of $(k + 1)$ operations $(U_0, U_1, \ldots, U_k)$, where U_i is unitary. Initially the state of the computation is set to some fixed value $|0, 0, 0\rangle$, and then the sequence of operations $U_0, O_f, U_1, O_f, \ldots, U_{k-1}, O_f, U_k$ is applied.

2.2. Notation. We denote the set $\{1, 2, \ldots, n\}$ by $[n]$. A simple undirected graph is a set of edges $G \subseteq \{(a, b) \mid \ a, b \in [n];\ a \neq b\}$ with the understanding that $(a, b) \overset{\text{def}}{=} (b, a)$. Let $t(G)$ denote the number of triangles in G. The complete graph on a set $\nu \subseteq [n]$ is denoted by ν^2. The neighborhood of a $v \in [n]$ in G is denoted by $\nu_G(v)$, and it is defined by $\nu_G(v) = \{b \mid \ (v, b) \in G\}$. We denote $|\nu_G(v)|$ by $\deg_G v$. For sets $A, B \subseteq [n]$ let $G(A, B) = \{(a, b) \mid \ a \in A;\ b \in B;\ (a, b) \in G\}$.

The following function will play a major role in our proof. We denote the number of paths of length 2 from $a \in [n]$ to $b \in [n]$ in G with $t(G, a, b)$: $t(G, a, b) = |\{x \mid (a, x) \in G;\ (b, x) \in G\}|$. For a graph $G \subseteq [n]^2$ and an integer $k \geq 0$, we define $G^{\langle k\rangle} = \{(a, b) \in [n]^2 \mid \ t(G, a, b) \leq k\}$.

2.3. Quantum subroutines. We will use a safe version of Grover Search [18], namely Safe Grover Search(t), based on t iterations of Grover Search and followed by a checking process for markedness of output instances.

FACT 2.1. *Let $c > 0$. Safe Grover Search$(\Theta(c \log N))$ on a database of N items has quantum query complexity $O(c\sqrt{N} \log N)$, and it always rejects if there is no marked item; otherwise it finds a marked item with probability at least $1 - \frac{1}{N^c}$.*

For quantum walks on graphs we usually define two operators: *coin flip* and *shift.* The state of the walk is held in a pair of registers, the *node* and the *coin.* The coin flip operator acts only on the coin register and is the identity on the node register. The shift operation changes only the node register, but it is controlled by the content of the coin register (see [29, 2, 7]). Often the coin flip is actually the diffusion operator.

DEFINITION 2.2 (diffusion over T). *Let T be a finite set. The* diffusion *operator over T is the unitary operator on the Hilbert space $\mathbf{C}^T$ that acts on a basis element $|x\rangle$, $x \in T$ as: $|x\rangle \mapsto -|x\rangle + \frac{2}{|T|} \sum_{y \in T} |y\rangle$.*

In [6] a new walk is described that plays a central role in our result. Let S be a finite set of size n. The node register holds a subset A of S of size either r or $r + 1$ for some fixed $0 < r < n$, and the coin register holds an element $x \in S$. Thus the basis states are of the form $|A\rangle|x\rangle$, where we also require that if $|A| = r$, then $x \notin A$, and if $|A| = r + 1$, then $x \in A$. We also call the node register the *set register.*

```
Quantum Walk.
    1. Diffuse the coin register over $S - A$.
    2. Add $x$ to $A$.
    3. Diffuse the coin register over $A$.
    4. Remove $x$ from $A$.
```

Ambainis [6] showed that, inside some specific stable subspaces, $\Theta(\sqrt{r})$ iterations of Quantum Walk can play the role of the diffusion over $\{(A,x) : A \subset S, |A| = r, x \notin S\}$. This nice result leads to a more efficient Grover Search for some problems such as the Element Distinctness Problem [6]. We will describe this in a general setting in section 4.1.

3. Combinatorial approach.

3.1. Preparation. The algorithm presented here is based on three combinatorial observations. Throughout this section we do not try to optimize $\log n$ factors and we will hide time in the $\tilde{O}$ notation. The first observation is based on the amplitude amplification technique of Brassard et al. [10].

LEMMA 3.1. *For any known graph $E \subseteq [n]^2$, a triangle with at least one edge in E can be detected with $\tilde{O}(\sqrt{E} + \sqrt{n|G \cap E|})$ queries and probability $1 - \frac{1}{n}$.*

Perhaps the most crucial observation to the algorithm is the following simple one.

LEMMA 3.2. *For every $v \in [n]$, using $\tilde{O}(n)$ queries, we either find a triangle in G or verify that $G \subseteq [n]^2 \setminus \nu_G(v)^2$ with probability $1 - \frac{1}{n^3}$.*

Proof. We query all edges incident to v classically using $n-1$ queries. This determines $\nu_G(v)$. With Safe Grover Search we find an edge of G in $\nu_G(v)^2$ if there is any. □

This lemma, with the observation that hard instances have to be dense, already enables us to show that the quantum query complexity of TRIANGLE is $o(n^{3/2})$, using the Szemerédi lemma [28]. However, another fairly simple observation can help us to decrease the exponent.

LEMMA 3.3. *Let $0 < \varepsilon < 1$, $k = \lceil 4n^{\varepsilon} \log n \rceil$, and let $v_1, v_2, \dots, v_k$ be randomly chosen from $[n]$ (with no repetitions). Let $G' = [n]^2 \setminus \cup_{i=1}^{k} \nu_G(v)^2$. Then $\Pr_{v_1,v_2,\dots,v_k}(G' \subseteq G^{\langle n^{1-\varepsilon} \rangle}) > 1 - \frac{1}{n}$.*

Let us first remind the reader about the following lemma that is useful in many applications.

LEMMA 3.4. *Let X be a fixed subset of $[n]$ of size pn and Y be a random subset of $[n]$ of size qn, where $p + q < 1$. Then the probability that $X \cap Y$ is empty is $(1-pq)^{n(1 \pm O(p^3+q^3+1/n))}$.*

Proof. The probability we are looking for is estimated using the Stirling formula as

$$\begin{aligned}\frac{\binom{n(1-p)}{nq}}{\binom{n}{nq}} &= \frac{[n(1-p)]![nq]![n(1-q)]!}{[nq]![n(1-p-q)]!n!} \\ &= \sqrt{\frac{(1-p)(1-q)}{1-p-q}} \left[\frac{(1-p)^{1-p}(1-q)^{1-q}}{(1-p-q)^{1-p-q}}\right]^n (1 \pm o(1)) \\ &= (1-pq)^{n(1 \pm O(p^3+q^3+1/n))}. \qquad \square\end{aligned}$$

Proof of Lemma 3.3. Consider now a fixed edge (a,b) such that $t(G,a,b) \geq n^{1-\varepsilon}$. The probability that $(a,b) \in G'$ is the same as the probability that the set $X = \{x \in [n] : (x,a) \in G \text{ and } (x,b) \in G\}$ is disjoint from the random set $\{v_1, v_2, \dots, v_k\}$. Notice that $|X| = t(G,a,b)$. By Lemma 3.4 we can now estimate this probability as, for sufficiently large n,

$$\left(1 - \frac{4n^{\varepsilon}\log n}{n} \times \frac{n^{1-\varepsilon}}{n}\right)^{n(1+o(1))} = \left(1 - \frac{4\log n}{n}\right)^{n(1+o(1))} < e^{-3\log n} = n^{-3}.$$

Then the lemma follows from the union bound, since the number of possible edges (a, b) is at most n^2. □

3.2. Algorithm and analysis. We now describe our algorithm where all searches are done using Safe Grover Search. We delay details of Step 6 for a while.

```
Combinatorial Algorithm($\varepsilon, \delta, \varepsilon'$).
    1. Let $k = \lceil 4n^{\varepsilon} \log n \rceil$.
    2. Randomly choose $v_1, \dots, v_k$ from $[n]$ (with no repetition).
    3. Compute every $\nu_G(v_i)$.
    4. If $G \cap \nu_G(v_i)^2 \neq \emptyset$, for some $i$, then output the triangle induced by $v_i$.
    5. Let $G' = [n]^2 \setminus \cup_i(\nu_G(v_i)^2)$.
    6. Classify the edges of $G'$ into $T$ and $E$ such that
       – $T$ contains only $O(n^{3-\varepsilon'})$ triangles,
       – $E \cap G$ has size $O(n^{2-\delta} + n^{2-\varepsilon+\delta+\varepsilon'})$.
    7. Search for a triangle in $G$ among all triangles inside $T$.
    8. Search for a triangle of $G$ intersecting with $E$.
    9. Output a triangle if it is found; otherwise reject.
```

THEOREM 3.5. *Combinatorial Algorithm*$(\varepsilon, \delta, \varepsilon')$ *rejects with probability one if there is no triangle in* G*; otherwise, it returns a triangle of* G *with probability* $1 - O(\frac{1}{n})$. *Moreover, it has query complexity* $\tilde{O}(n^{1+\varepsilon} + n^{1+\delta+\varepsilon'} + \sqrt{n^{3-\varepsilon'}} + \sqrt{n^{3-\min(\delta,\varepsilon-\delta-\varepsilon')}})$.

With $\varepsilon = \frac{3}{7}$, $\varepsilon' = \delta = \frac{1}{7}$ this gives $\tilde{O}(n^{1+\frac{3}{7}})$ for the total number of queries.

We require every probabilistic step to be correctly performed with probability $1 - O(\frac{1}{n^3})$, so that the overall probability of a correct execution is $1 - O(\frac{1}{n})$, using the union bound and since the number of such steps is at most $O(n^2)$. Thus we will always assume that an execution is correct. Since an incorrect execution might increase the query complexity of the algorithm, we also assume there is a counter so that the algorithm rejects and stops when a threshold is exceeded. This threshold is defined as the maximum of query complexities over all correct executions.

The main step of Combinatorial Algorithm is step 6, which we implement in the following way.

```
Classification($G', \delta, \varepsilon'$).
    1. Set $T = \emptyset$, $E = \emptyset$.
    2. While $G' \neq \emptyset$ do
        (a) While there is an edge $(v, w) \in G'$ s.t. $t(G', v, w) < n^{1-\varepsilon'}$,
              add $(v, w)$ to $T$, and delete it from $G'$.
        (b) Pick a vertex $v$ of $G'$ with nonzero degree and decide
            1. low degree hypothesis: $|\nu_G(v)| \leq 10 \times n^{1-\delta}$;
            2. high degree hypothesis: $|\nu_G(v)| \geq \frac{1}{10} \times n^{1-\delta}$.
        (c) If Hypothesis 1, add all edges $(v, w)$ of $G'$ to $E$ and delete them
            from $G'$.
        (d) If Hypothesis 2, then
              i. Compute $\nu_G(v)$;
             ii. If $G \cap \nu_G(v)^2 \neq \emptyset$, output the triangle induced by $v$ and stop;
            iii. Add all edges in $G'(\nu_G(v), \nu_{G'}(v))$ to $E$ and delete them from
                 $G'$.
```

In step 2(b), we use an obvious sampling strategy:

> Set a counter C to 0. Query $\lceil n^{\delta} \rceil$ random edge candidates from $v \times [n]$. If there is an edge of G among them, add one to C. Repeat this process $K = c_0 \log n$ times, where c_0 is a sufficiently large; constant. Accept the low degree hypothesis if by the end $C < K/2$; otherwise accept the large degree hypothesis.

Observe that one could use here a quantum procedure based on Grover Search. Since the cost of this step is negligible from that of others, this would not give any better bound.

FACT 3.6. *When c_0 is large enough in Step* 2(b),

1. *the probability that* $\deg_G(v) > 10 \times n^{1-\delta}$ *and the low degree hypothesis is accepted is $O(\frac{1}{n^3})$;*
2. *the probability that* $\deg_G(v) < \frac{1}{10} \times n^{1-\delta}$ *and the high degree hypothesis is accepted is $O(\frac{1}{n^3})$.*

Proof. Indeed, using Lemma 3.4, considering a single round of sampling, the probability that our sample set does not contain an edge from G even though $\deg_G(v) > 10 \times n^{1-\delta}$ is, for sufficiently large n,

$$\left(1 - \frac{10n^{1-\delta}}{n} \times \frac{n^{\delta}}{n}\right)^{n(1+o(1))} = \left(1 - \frac{10}{n}\right)^{n(1+o(1))} < 0.1.$$

Similarly, the probability that our sample set contains an edge from G even though $\deg_G(v) < \frac{1}{10} \times n^{1-\delta}$ is

$$1 - \left(1 - \frac{n^{1-\delta}}{10n} \times \frac{n^{\delta}}{n}\right)^{n(1+o(1))} = 1 - \left(1 - \frac{1}{10n}\right)^{n(1+o(1))} < 0.2.$$

Now for $K = c_0 \log n$ rounds, where c_0 is large enough, the Chernoff bound gives the claim. □

LEMMA 3.7. *If $G \subseteq G' \subseteq G^{\langle n^{1-\varepsilon} \rangle}$, then Classification$(G', \varepsilon', \delta)$ outputs the desired partition (T, E) of G with probability $1 - O(\frac{1}{n})$ and has query complexity $\tilde{O}(n^{1+\delta+\varepsilon'})$.*

Proof of Theorem 3.5. Clearly, if there is no triangle in the graph, the algorithm rejects since the algorithm outputs a triplet only after checking that it is a triangle in G. Therefore the correctness proof requires us only to calculate the probability with which the algorithm outputs a triangle, if there is any, and the query complexity of the algorithm.

Assume that the execution is without any error. Using the union bound, we can indeed upper bound the probability of incorrect execution by $O(\frac{1}{n})$.

By Lemma 3.2, we already know that the construction of G' requires $\tilde{O}(n^{\varepsilon} \times n)$ queries. Moreover, either $G \subseteq G'$ or a triangle is found, with probability $1 - O(\frac{1}{n})$. From Lemma 3.3, we also know that $G' \subseteq G^{\langle n^{1-\varepsilon} \rangle}$, with probability $1 - O(\frac{1}{n})$.

Assume that G' lends all its edges to T and E; that is, no triangle is found at the end of Classification. Since $G \subseteq G'$, every triangle in G either has to be contained totally in T or has to have a nonempty intersection with E. Using Lemma 3.7, we know that the partition (T, E) is correct with probability $1 - O(\frac{1}{n})$. Assume this is the case. T is a graph that is known to us, and so we can find out if one of these triangles belongs to G with $\tilde{O}(\sqrt{n^{3-\varepsilon'}})$ queries, using Safe Grover Search. By Lemma 3.1, the complexity of finding a triangle in G that contains an edge from E is $\tilde{O}(n + \sqrt{n^{3-\min(\delta, \varepsilon-\delta-\varepsilon')}})$.

From the analysis we conclude that the total number of queries is upper bounded by

$$\tilde{O}\left(n^{1+\varepsilon} + n^{1+\delta+\varepsilon'} + \sqrt{n^{3-\varepsilon'}} + \sqrt{n^{3-\min(\delta,\varepsilon-\delta-\varepsilon')}}\right). \qquad \square$$

In the rest of this section we prove Lemma 3.7 using a sequence of facts. Then the proof derives directly. For the query complexity, we detail the analysis using Fact 3.8. Only steps 2(b) and 2(d) of Classification have nonzero query complexity. As explained before, step 2(b) can be implemented with query complexity $\tilde{O}(n^{\delta})$, and it is iterated at most n times. Step 2(d) has three substeps, with only the first two having nonzero query complexity. The first has query complexity $O(n)$, and the second can be implemented using Safe Grover Search with query complexity $\tilde{O}(\sqrt{n^2}) = \tilde{O}(n)$. Using Fact 3.8, we upper bound the number of iterations of step 2(d) by $O(n^{\delta+\varepsilon'})$, which gives a total amount of queries in $\tilde{O}(n^{1+\delta+\varepsilon'})$.

FACT 3.8. *During a correct execution, there are at most $O(n^{\delta+\varepsilon'})$ iterations of step* 2(d).

Proof. We will estimate the number of executions of step 2(d) by lower bounding $|G'(A, A')|$, where $A = \nu_G(v)$ and $A' = \nu_{G'}(v)$. For each $x \in A$ we have $t(G', v, x) \geq n^{1-\varepsilon'}$; otherwise in step 2(a) we would have classified (v, x) into T. A triangle (v, x, y) contributing to $t(G', v, x)$ contributes with the edge (x, y) to $G'(A, A')$. Two different triangles (v, x, y) and (v, x', y') can give the same edge in $G'(A, A')$ only if $x = y'$ and $y = x'$. Thus,

$$|G'(A, A')| \geq \frac{1}{2} \sum_{x \in \nu_G(v)} t(G', v, x) \geq \frac{|A| n^{1-\varepsilon'}}{2}. \tag{3.1}$$

Since we executed step 2(d) only under the large degree hypothesis on v, if the hypothesis is correct, the right-hand side of (3.1) is at least $\frac{1}{10} \times n^{1-\delta} \times n^{1-\varepsilon'}/2 = \Omega(n^{2-\delta-\varepsilon'})$. Since G' has at most $\binom{n}{2}$ edges, it can execute step 2(d) at most $O(n^{\delta+\varepsilon'})$ times. $\square$

FACT 3.9. *During a correct execution, there are at most $O(n)$ iterations of step* 2(c).

Proof. We claim that each vertex is processed in step 2(c) at most once. Indeed, if a vertex v gets into step 2(c), its incident edges are all removed, and its degree in G' becomes 0, making it ineligible for being processed in step 2(c) again. $\square$

Now we state that T contains $O(n^{3-\varepsilon'})$ triangles using the following quite general fact.

FACT 3.10. *Let H be a graph on $[n]$. Assume that a graph T is built by a process that starts with an empty set and at every step either discards some edges from H or adds an edge (a, b) of H to T for which $t(H, a, b) \leq \tau$ holds. For the T created by the end of the process we have $t(T) \leq \binom{n}{2}\tau$.*

Proof. Let us denote by $T[i]$ the edge of T that T acquired when it was incremented for the ith time, and let us use the notation H^i for the current version of H before the very moment when $T[i] = (a_i, b_i)$ was copied into T. Since $\{T[i], T[i+1], \dots\} \overset{\text{def}}{=} T^i \subseteq H^i$, we have $t(T^i, a_i, b_i) \leq t(H^i, a_i, b_i) \leq \tau$. Now the fact follows from $t(T) = \sum_i t(T^i, a_i, b_i) \leq \binom{n}{2}\tau$, since i can go up to at most $\binom{n}{2}$. $\square$

FACT 3.11. *During a correct execution, $E \cap G$ has size $O(n^{2-\delta} + n^{2-\varepsilon+\delta+\varepsilon'})$.*

Proof. In order to estimate $E \cap G$ observe that we added edges to E only in steps 2(c) and 2(d). In each execution of step 2(c), we added at most $10n^{1-\delta}$ edges to E, and we had $O(n)$ such executions (Fact 3.9) that give a total of $O(n^{2-\delta})$ edges. The number of executions of step 2(d) is $O(n^{\delta+\varepsilon'})$ (Fact 3.8). Our task is now to bound the number of edges of G that each such execution adds to E.

We estimate $|G \cap G'(A, A')|$ from the A' side, where $A = \nu_G(v)$ and $A' = \nu_{G'}(v)$. This is the only place where we use the fact that $G' \subseteq G^{\langle n^{1-\varepsilon}\rangle}$: For every $x \in A'$ we have $t(G, v, x) \leq n^{1-\varepsilon}$. On the other hand, when $y \in A$ and $x \in A'$, every edge $(y, x) \in G'$ creates a (v, x)-based triangle. Thus

$$|G \cap G'(A, A')| \leq |A'| n^{1-\varepsilon} \leq n^{2-\varepsilon}.$$

Therefore the total number of edges of G step 2(d) contributes to E is $n^{2-\varepsilon+\delta+\varepsilon'}$. In conclusion,

$$|G \cap E| \leq O(n^{2-\delta} + n^{2-\varepsilon+\delta+\varepsilon'}). \qquad \square$$

4. Quantum Walk approach.

4.1. Dynamic quantum query algorithms. The algorithm of Ambainis in [6] is somewhat similar to the brand of classical algorithms, where a database is used (as in heapsort) to quickly retrieve the value of those items needed for the run of the algorithm. Of course, this whole paradigm is placed into the context of query algorithms. We shall define a class of problems that can be tackled very well with the new type of algorithm. Let S be a finite set of size n and let $0 < k < n$.

k-Collision
Oracle Input: A function f which defines a relation $\mathcal{C} \subseteq S^k$.
Output: A k-tuple $(a_1, \ldots, a_k) \in \mathcal{C}$ if it is nonempty; otherwise reject.

By carefully choosing the relation $\mathcal{C}$, k-Collision can be a useful building block in the design of different algorithms. For example, if f is the adjacency matrix of a graph G, and the relation $\mathcal{C}$ is defined as "being an edge of a triangle of G," then the output of Collision yields a solution for Triangle with $O(\sqrt{n})$ additional queries (Grover Search for the third vertex).

Unique k-Collision: The same as k-Collision with the promise that $|\mathcal{C}| = 1$ or $|\mathcal{C}| = 0$.

The type of algorithm we study will use a database D associating some data $D(A)$ to every set $A \subseteq S$. From $D(A)$ we would like to determine if $A^k \cap \mathcal{C} \neq \emptyset$. We expedite this using a quantum query procedure Φ with the property that $\Phi(D(A))$ rejects if $A^k \cap \mathcal{C} = \emptyset$ and, otherwise, both accepts and outputs an element of $A^k \cap \mathcal{C}$, which is a *collision*. When operating with D, the following three types of costs are incurred, all measured in the number of queries to the oracle f.

Setup cost $s(r)$: The cost to set up $D(A)$ for a set of size r.

Update cost $u(r)$: The cost to update D for a set of size r, i.e., moving from $D(A)$ to $D(A')$, where A' results from A by adding an element, or moving from $D(A'')$ to $D(A)$, where A results from A'' by deleting an element.

Checking cost $c(r)$: The query complexity of $\Phi(D(A))$ for a set of size r.

Next we describe the algorithm of Ambainis [6] in general terms. The algorithm has three registers $|A\rangle|D(A)\rangle|x\rangle$. The first is called the *set register*, the second the *data register*, and the last the *coin register*.

```
**Generic Algorithm**$(r, D, \Phi)$.
  1. Create the state $\sum_{A \subset S : |A| = r} |A\rangle$ in the set register.
  2. Set up $D$ on $A$ in the data register.
  3. Create a uniform superposition over elements of $S - A$ in the coin register.
  4. Do $\Theta(n/r)^{k/2}$ times:
     (a) If $\Phi(D(A))$ accepts, then do the phase flip; otherwise do nothing
     (b) Do $\Theta(\sqrt{r})$ times **Quantum Walk**, updating the data register.
  5. If $\Phi(D(A))$ rejects, then reject; otherwise output the collision given by $\Phi(D(A))$.
```

THEOREM 4.1 (see [6]). Generic Algorithm *solves* UNIQUE k-COLLISION *with some positive constant probability and has query complexity*

$$O(s(r) + (\tfrac{n}{r})^{k/2} \times (c(r) + \sqrt{r} \times u(r))).$$

Moreover, it turns out that, when UNIQUE k-COLLISION has no solution, Generic Algorithm always rejects, and when UNIQUE k-COLLISION has a solution c, Generic Algorithm outputs c with probability $p = \Omega(1)$ which depends only on k, n, and r. Thus using quantum amplification, one can modify Generic Algorithm to an exact quantum algorithm.

COROLLARY 4.2. UNIQUE k-COLLISION *can be solved with probability* 1 *in quantum query complexity*

$$O(s(r) + (\tfrac{n}{r})^{k/2} \times (c(r) + \sqrt{r} \times u(r))).$$

One can make a random reduction from COLLISION to UNIQUE COLLISION if the definition on Φ is slightly generalized. We add to the input of the checking procedure a relation $\mathcal{R} \subseteq S^k$ which restricts the collision set $\mathcal{C}$ to $\mathcal{C} \cap \mathcal{R}$. The reduction goes in the standard way using a logarithmic number of randomly chosen relations $\mathcal{R}$, and hence an additional logarithmic factor appears in the complexity. If the collision relation is robust in some sense, one can improve this reduction by removing the log factors (see, for example, the reduction used by Ambainis in [6]).

COROLLARY 4.3. COLLISION *can be solved in quantum query complexity*

$$\tilde{O}(s(r) + (\tfrac{n}{r})^{k/2} \times (c(r) + \sqrt{r} \times u(r))).$$

The tables below summarize the use of the above formula for various problems (GRAPH COLLISION(G) is defined in section 4.2).

Problem	Collision relation
ELEMENT DISTINCTNESS	$(u, v) \in \mathcal{C}$ iff $u \neq v$ and $f(u) = f(v)$
GRAPH COLLISION(G)	$(u, v) \in \mathcal{C}$ iff $f(u) = f(v) = 1$ and $(u, v) \in G$
TRIANGLE	$(u, v) \in \mathcal{C}$ iff there is a triangle (u, v, w) in G

Problem	Setup cost $s(r)$	Update cost $u(r)$	Checking cost $c(r)$
ELEMENT DISTINCTNESS	r	1	0
GRAPH COLLISION(G)	r	1	0
TRIANGLE	$O(r^2)$	r	$O(r^{2/3}\sqrt{n})$

4.2. Graph Collision Problem. Here we deal with an interesting variant of COLLISION which will be also useful for finding a triangle. The problem is parametrized by some graph G on n vertices which is given explicitly.

GRAPH COLLISION(G)
Oracle Input: A boolean function f on $[n]$ which defines the relation $\mathcal{C} \subseteq [n]^2$ such that $\mathcal{C}(u,u')$ if and only if $f(u) = f(u') = 1$ and $(u,u') \in E$.
Output: A pair $(u,u') \in \mathcal{C}$ if it is nonempty; otherwise reject.

Observe that an equivalent formulation of the problem is to decide if the set of vertices of value 1 form an independent set in G.

THEOREM 4.4. GRAPH COLLISION(G) *can be solved with positive constant probability in quantum query complexity* $\tilde{O}(n^{2/3})$.

Proof. We solve GRAPH COLLISION(G) using Corollary 4.3, with $S = [n]$ and $r = n^{2/3}$. For every $U \subseteq [n]$ we define $D(U) = \{(v, f(v)) : v \in U\}$ and let $\Phi(D(U)) = 1$ if there are $u, u' \in U$ that satisfy the required property. Observe that $s(r) = r$, $u(r) = 1$, and $c(r) = 0$. Therefore we can solve the problem in quantum query complexity $\tilde{O}(r + \frac{n}{r}(\sqrt{r}))$ which is $\tilde{O}(n^{2/3})$ when $r = n^{2/3}$. □

4.3. Triangle Problem.

THEOREM 4.5. TRIANGLE *can be solved with positive constant probability in quantum query complexity* $\tilde{O}(n^{13/10})$.

Proof. We use Corollary 4.3, where $S = [n]$, $r = n^{2/3}$, and $\mathcal{C}$ is the set of triangle edges. We define D for every $U \subseteq [n]$ by $D(U) = G|_U$, and Φ by $\Phi(G|_U) = 1$ if a triangle edge is in $G|_U$. Observe that $s(r) = O(r^2)$ and $u(r) = r$. We claim that $c(r) = \tilde{O}(\sqrt{n} \times r^{2/3})$.

To see this, let U be a set of r vertices such that $G|_U$ is explicitly known, and let v be a vertex in $[n]$. We define an input oracle for GRAPH COLLISION($G|_U$) by $f(u) = 1$ if $(u,v) \in E$. The edges of $G|_U$ which together with v form a triangle in G are the solutions of GRAPH COLLISION($G|_U$). Therefore finding a triangle edge, if it is in $G|U$, can be done in quantum query complexity $\tilde{O}(r^{2/3})$ by Theorem 4.4. Now using quantum amplification [10], we can find a vertex v, if it exists, which forms a triangle with some edge of $G|_U$, using only $\tilde{O}(\sqrt{n})$ iterations of the previous procedure, and with a polynomially small error (which has no influence in the whole algorithm).

Therefore, we can solve the problem in quantum query complexity $\tilde{O}(r^2 + \frac{n}{r}(\sqrt{n} \times r^{2/3} + \sqrt{r} \times r))$ which is $\tilde{O}(n^{13/10})$ when $r = n^{3/5}$. □

4.4. Monotone graph properties with small certificates. Let now consider the property of having a copy of a given graph H with $k > 3$ vertices. Using directly Ambainis' algorithm, one gets an algorithm whose query complexity is $\tilde{O}(n^{2-2/(k+1)})$. In fact we can improve this bound to $\tilde{O}(n^{2-2/k})$. Note that only the trivial $\Omega(n)$ lower bound is known. This problem was independently considered by Childs and Eisenberg [14] whenever H is a k-clique. Beside the direct Ambainis algorithm, they obtained an $\tilde{O}(n^{2.5-6/(k+2)})$ query algorithm. For $k = 4, 5$, this is faster than the direct Ambainis algorithm, but slower than ours.

THEOREM 4.6. *Finding in a graph a copy of a given graph H, with $k > 3$ vertices, can be done with quantum query complexity* $\tilde{O}(n^{2-2/k})$.

Proof. We follow the structure of the proof of Theorem 4.5. We distinguish an arbitrary vertex of H. Let d be the degree of this vertex in H.

We say that a vertex v and a set K of $(k-1)$ vertices of G are *H-compatible* if the subgraph induced by $K \cup \{v\}$ in G contains a copy of H, in which v is the distinguished vertex. We also say that the set K is an *H-candidate* when there exists a vertex v such that v and K are H-compatible. Our algorithm will essentially find a set that contains an H-candidate.

We define an instance of $(k-1)$-Collision, where $S = [n]$ and $\mathcal{C}$ is the set of H-candidates. We define D for every $U \subseteq [n]$ by $D(U) = G|_U$, and Φ by $\Phi(G|_U) = 1$ if U contains an H-candidate. Again $s(r) = O(r^2)$ and $u(r) = r$. We now claim that $c(r) = \tilde{O}(\sqrt{n} \times r^{d/(d+1)})$.

The checking procedure uses a generalization of Graph Collision to d-ary relations. If some vertex v of G is fixed, then we say that a subset $W \subseteq U$ of size d is in relation if there exists $W \subseteq K \subseteq U$ such that v and K are H-compatible in G, and v is connected to every vertex of W. Following the arguments of the proof of Theorem 4.4 (where the function f takes the value 1 on a vertex $u \in U$ if (u, v) is an edge in G), we find a d-collision in quantum query complexity $\tilde{O}(r^{d/(d+1)})$ when it exists. The checking procedure searches for a vertex v for which this generalized Graph Collision has a solution using a standard Grover Search.

The overall parametrized query complexity is therefore

$$\tilde{O}\left(r^2 + \left(\frac{n}{r}\right)^{(k-1)/2}\left(\sqrt{n} \times r^{d/(d+1)} + \sqrt{r} \times r\right)\right).$$

By optimizing this expression (that is, by balancing the first and third terms), it turns out that the best upper bound does not depend on d. Precisely, the expression is optimal with $r = n^{1-1/k}$, which gives the announced bound. However, one can imagine a different algorithm for the checking procedure where the choice of d might be crucial.

To conclude, note that once a set U of size r that contains an H-candidate is found, one can obtain a copy of H in G in the complexity of the checking cost $c(r)$. □

We conclude by extending this result for monotone graph properties which might have several small 1-certificates.

Corollary 4.7. *Let φ be a monotone graph property whose 1-certificates have at most $k > 3$ vertices. Then deciding φ and producing a certificate whenever φ is satisfied can be done with quantum query complexity to the graph in $\tilde{O}(n^{2-2/k})$.*

Acknowledgment. We would like to thank Andris Ambainis for useful discussions and for sending us a preliminary version of [6].

REFERENCES

[1] S. Aaronson and Y. Shi, *Quantum lower bounds for the collision and the element distinctness problems*, J. ACM, 51 (2004), pp. 595–605.

[2] D. Aharonov, A. Ambainis, J. Kempe, and U. Vazirani, *Quantum walks on graphs*, in Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing, 2001, pp. 50–59.

[3] N. Alon, R. Yuster, and U. Zwick, *Finding and counting given length cycles*, Algorithmica, 17 (1997), pp. 209–223.

[4] A. Ambainis, *Quantum lower bounds by quantum arguments*, J. Comput. System Sci., 64 (2002), pp. 750–767.

[5] A. Ambainis, *Polynomial degree vs. quantum query complexity*, J. Comput. System Sci., 72 (2006), pp. 220–238.

[6] A. Ambainis, *Quantum walk algorithm for element distinctness*, in Proceedings of the 45th IEEE Symposium on Foundations of Computer Science, 2004, pp. 22–31.

[7] A. Ambainis, E. Bach, A. Nayak, A. Vishwanath, and J. Watrous, *One-dimensional quantum walks*, in Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing, 2001, pp. 37–49.

[8] H. Barnum, M. Saks, and M. Szegedy, *Quantum decision trees and semidefinite programming*, in Proceedings of the 18th IEEE Annual Conference on Computational Complexity, 2003, pp. 179–193.

[9] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf, *Quantum lower bounds by polynomials*, J. ACM, 48 (2001), pp. 778–797.

[10] G. Brassard, P. Høyer, M. Mosca, and A. Tapp, *Quantum amplitude amplification and estimation*, in Quantum Computation and Quantum Information: A Millennium Volume, Contemp. Math. 305, AMS, Providence, RI, 2002, pp. 53–74.

[11] H. Buhrman, R. Cleve, R. de Wolf, and C. Zalka, *Bounds for small-error and zero-error quantum algorithms*, in 40th Annual Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 1999, pp. 358–368.

[12] H. Buhrman, C. Dürr, M. Heiligman, P. Høyer, F. Magniez, M. Santha, and R. de Wolf, *Quantum algorithms for element distinctness*, SIAM J. Comput., 34 (2005), pp. 1324–1330.

[13] A. Chakrabarti and S. Khot, *Improved lower bounds on the randomized complexity of graph properties*, in Automata, Languages and Programming, Lecture Notes in Comput. Sci. 2076, Springer-Verlag, Berlin, 2001, pp. 285–296.

[14] A. Childs and J. Eisenberg, *Quantum Algorithms for Subset Finding*, Tech. report, Massachusetts Institute of Technology, Cambridge, MA, 2003; available online from http://www.arxiv.org/abs/quant-ph/0311038.

[15] D. Deutsch, *Quantum theory, the Church-Turing principle and the universal quantum computer*, Proc. Roy. Soc. London Ser. A, 400 (1985), pp. 97–117.

[16] D. Deutsch and R. Jozsa, *Rapid solution of problems by quantum computation*, Proc. Roy. Soc. London Ser. A, 439 (1992), pp. 553–558.

[17] M. Ettinger, P. Høyer, and E. Knill, *The quantum query complexity of the hidden subgroup problem is polynomial*, Inform. Process. Lett., 91 (2004), pp. 43–48.

[18] L. Grover, *A fast quantum mechanical algorithm for database search*, in Proceedings of the Twenty-Eighth ACM Symposium on Theory of Computing, 1996, pp. 212–219.

[19] P. Hajnal, *An $n^{4/3}$ lower bound on the randomized complexity of graph properties*, Combinatorica, 11 (1991), pp. 131–143.

[20] A. Kitaev, A. Shen, and M. Vyalyi, *Classical and Quantum Computation*, Grad. Stud. Math. 47, AMS, Providence, RI, 2002.

[21] S. Laplante and F. Magniez, *Lower bounds for randomized and quantum query complexity using Kolmogorov arguments*, in Proceedings of 19th IEEE Annual Conference on Computational Complexity, 2004, pp. 214–304.

[22] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, Cambridge, UK, 2000.

[23] R. Rivest and J. Vuillemin, *On recognizing graph properties from adjacency matrices*, Theoret. Comput. Sci., 3 (1976), pp. 371–384.

[24] A. Rosenberg, *On the time required to recognize properties of graphs: A problem*, SIGACT News, 5 (1973), pp. 15–16.

[25] D. R. Simon, *On the power of quantum computation*, SIAM J. Comput., 26 (1997), pp. 1474–1483.

[26] R. Špalek and M. Szegedy, *All quantum adversary methods are equivalent*, in Automata, Languages and Programming, Lecture Notes in Comput. Sci. 3580, Springer-Verlag, Berlin, 2005, pp. 1299–1311.

[27] M. Szegedy, *On the Quantum Query Complexity of Detecting Triangles in Graphs*, Tech. report, 2003; available online from http://www.arxiv.org/abs/quant-ph/0310107.

[28] E. Szemerédi, *Regular partitions of graphs*, in Problèmes combinatoires et théorie des graphes (1976), Colloq. Internat. CNRS 260, CNRS, Paris, 1978, pp. 399–401.

[29] J. Watrous, *Quantum simulations of classical random walks and undirected graph connectivity*, J. Comput. System Sci., 62 (2001), pp. 376–391.

[30] A. Yao, *Lower bounds to randomized algorithms for graph properties*, J. Comput. System Sci., 42 (1991), pp. 267–287.

[31] A. Yao, *private communication*, 2003.

[32] S. Zhang, *On the power of Ambainis's lower bounds*, in Automata, Languages and Programming, Lecture Notes in Comput. Sci. 3142, Springer-Verlag, Berlin, 2004, pp. 1238–1250.

SIAM J. COMPUT.
Vol. 37, No. 2, pp. 425–459

MEMBERSHIP PROBLEM FOR THE MODULAR GROUP[*]

YURI GUREVICH[†] AND PAUL SCHUPP[‡]

Abstract. The modular group plays an important role in many branches of mathematics. We show that the membership problem for the modular group is decidable in polynomial time. To this end, we develop a new syllable-based version of the known subgroup-graph approach. The new approach can be used to prove additional results. We demonstrate this by using it to prove that the membership problem for a free group remains decidable in polynomial time when elements are written in a normal form with exponents.

Key words. $\mathrm{PSL}_2(\mathbb{Z})$, combinatorial group theory, folding, free group, geometric group theory, membership problem, modular group, polynomial time, syllabic presentation, unimodular matrices

AMS subject classifications. 03B25, 03D40, 11U05, 14L35, 15A30, 15A36, 20F10, 20G99

DOI. 10.1137/050643295

1. Introduction.

In this paper, a *unimodular matrix* is a 2×2 integer matrix with determinant 1. The multiplicative group of unimodular matrices is known as $\mathrm{SL}_2(\mathbb{Z})$, the special linear group of 2×2 matrices over the ring of integers. The modular group $\mathrm{PSL}_2(\mathbb{Z})$, the projective special linear group of 2×2 matrices over integers, is the quotient of the group $\mathrm{SL}_2(\mathbb{Z})$ modulo the congruence relation that equates a matrix $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ with its negative $\left(\begin{smallmatrix} -a & -b \\ -c & -d \end{smallmatrix}\right)$. The modular group has numerous equivalent characterizations in various parts of mathematics [2, section 1]. In particular, it is the group of complex fractional linear transformations $z \mapsto \frac{az+b}{cz+d}$ with integer coefficients and $ad - bc = 1$.

Recall the membership problem for a group G: given elements $h_1, \ldots, h_n$ and w, determine whether w belongs to the subgroup H generated by $h_1, \ldots, h_n$. This presumes a fixed representation form for group elements. In the case of the modular group, group elements are represented by unimodular matrices. A matrix and its negative represent the same group element. The entries are written in the standard decimal notation. The *size* of an entry is the length of its decimal notation, and the *size* of a unimodular matrix is the sum of the sizes of the four entries.

Remark 1.1 (uniformity). The membership problem above is sometimes called uniform because the subgroup H is not fixed. The problem of deciding whether a given element w of the group G belongs to a fixed subgroup H is called the *membership problem for H in G*. We restrict attention to uniform membership problems and will not use the adjective "uniform."

The membership problem for the modular group, more exactly its bounded version, was raised by Gurevich [4], who was looking for a hard-on-average [6, 3] algebraic NP problem with a natural probability distribution on the instances. In the bounded version of the membership problem for a group G, in addition to a tuple $(h_1, \ldots, h_n, w)$, one is given a positive integer B in the unary notation; the question becomes whether w is a product of at most B of the elements h_i and their inverses.

[*] Received by the editors October 21, 2005; accepted for publication (in revised form) December 12, 2006; published electronically May 16, 2007.
http://www.siam.org/journals/sicomp/37-2/64329.html

[†] Microsoft Research, One Microsoft Way, Redmond, WA 98052 (gurevich@microsoft.com).
[‡] Mathematics, University of Illinois at Urbana-Champaign, Urbana, IL 61801 (schupp@math.uiuc.edu).

After examining the bounded membership problem for the modular group (with an appropriate natural probability distribution), Gurevich conjectured that the problem is not hard on average.

In [2], Cai et al. proved that the bounded membership problem for the modular group is indeed polynomial time on average. They also proved that the unbounded membership problem for the modular group is polynomial time on average. Furthermore, consider the variant of the membership problem definition in which "subgroup" is replaced with "submonoid." The subgroup membership problem can be seen as a special case of the submonoid membership problem where the set $\{h_1, \ldots, h_m\}$ is closed under inverses. Cai et al. proved that both, bounded and unbounded, submonoid membership problems for the modular group are polynomial time on average [2, Theorem 1.1]. All their proofs are constructive: the desired decision procedures are exhibited. In this paper, the proofs are constructive as well.

Proviso 1.2. In this paper, all proofs of the existence of algorithms are constructive. The desired algorithms are exhibited.

As far as the worst-case analysis is concerned, Cai et al. established that the two submonoid membership problems are NP-hard. The bounded membership problem for the modular group was proved NP-hard in [1]. More precisely, it is the group $\mathrm{SL}_2(\mathbb{Z})$ that is called the modular group in [4, 1], and it is the bounded membership problem for $\mathrm{SL}_2(\mathbb{Z})$ that is proved NP-hard in [1]. But the same proof establishes also the NP-hardness of the bounded membership problem for $\mathrm{PSL}_2(\mathbb{Z})$.

THEOREM 1.3 (main). *The membership problem for the modular group is decidable in polynomial time.*

Group membership problems tend to be undecidable [7]. The modular group is atypical from that point of view. It is curious also that, in the case of the modular group, the unbounded membership problem is easier than the bounded one.

We do not try to optimize the decision algorithm and minimize its running time. This gives us freedom to ignore various details, most importantly the details related to various data structures. It is clear though that the algorithm is feasible.

To solve the membership problem for the modular group, we develop a new version of the subgroup-graph approach of combinatorial group theory. The subgroup-graph approach, also known as the folding method, was originated by Stallings [10]. It was employed and developed further in particular by Kapovich and Miasnikov [5] and Schupp [9]. Our version of the approach is combinatorial. The closest version in the literature is that of Kapovich and Miasnikov [5].

The subgroup graph in question is really a finite automaton, in general nondeterministic. We call it a subgroup recognizer or simply a recognizer. We call our approach the syllabic recognizer approach or simply the syllabic approach. It is based on the notion of a syllable. A recognizer reads words one syllable at a time. Another distinctive feature of our approach is that we fold paths rather than edges.

In section 2, we introduce syllabic representations of abstract groups and explain the basics of the syllabic approach. A syllabic presentation of an abstract group G with a fixed finite set of generators is given by means of four items. First, there is a finite alphabet with letters representing the generators and possibly some auxiliary symbols. Second, there is a set of strings in the given alphabet. These strings are called syllables. Finite concatenations of syllables are called words. The words with the concatenation operation form a semigroup. Third, there is an involution on the syllables called the inverse operation. It extends to words in the obvious way. Finally, there is a congruence relation on the word semigroup with the inverse operation such that the quotient algebra is isomorphic to G. Notice that words are strings in the given alphabet. Accordingly the size of a word is the length of the string. Often it suffices to describe the syllables, and the rest of the syllabic representation becomes obvious.

Example 1.4 (standard and succinct free groups). Consider the case in which G is a free group and the fixed set of generators consists of the free generators of G. The standard representation of G is obtained when the syllables have the form a or a^{-1}, where a is a free generator. Another, exponentially more succinct representation of G is obtained when the syllables have the form a^i, where a is a free generator and i is a nonzero integer in decimal notation. For brevity we will speak about *standard free groups* and *succinct free groups* meaning free groups in the standard representation and free groups in the succinct representation, respectively.

Proviso 1.5 (the free group). We will study finitely generated free groups with at least two free generators. The number of free generators will play an insignificant role. To simplify terminology, we fix some integer ≥ 2 and restrict attention to that particular free group.

The membership problem for the standard free group is decidable in polynomial time [8, 7]. We construct a new decision algorithm for the problem in section 2. The purpose of this is twofold: to illustrate the syllabic approach on a simple example and to produce a proof template for sections 3–5.

In section 3 we prove that the membership problem for the free group remains feasible when we go from the standard representation to the succinct.

Theorem 1.6 (succinct free group). *The membership problem for the succinct free group is decidable in polynomial time.*

But the proof of polynomial time decidability is much harder in the case of the succinct representation. One reason for this is that the number of syllables is infinite. Another reason is that the classical edge-folding technique used in the standard case is utterly inadequate in the succinct case. Instead we have to fold paths.

What has all this to do with the membership problem for the modular group? The bridge is the following well-known fact [8, section 1.4, Exercises 18–24]. Recall that, in the notation of combinatorial group theory, $\langle g \mid g^n \rangle$ is a cyclic group of order n with generator g.

Proposition 1.7 (modular group as a free product). *The modular group is isomorphic to the free product* $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$.

Example 1.8 (standard $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$ and succinct $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$). We give two syllabic representations of the group $G = \langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$ with the set s, t of generators. (The symbols s, t allude to "second" and "third," respectively.) The standard representation of G is obtained when the syllables are s, t, t^{-1}. Another, exponentially more succinct representation of G is obtained when the syllables are s, $(ts)^n t$, and $(t^{-1}s)^n t^{-1}$, where n is a natural number in decimal notation. For brevity we will speak about the *standard* $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$ and the *succinct* $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$ meaning $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$ in the standard representation and $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$ in the succinct representation, respectively.

In section 4, we prove that the membership problem for the standard $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$ is decidable in polynomial time. The proof sheds some light on the membership problem for the modular group but is not too useful all by itself.

Ideas and the terminology of sections 2–4 are used in the crucial section 5, where we study the succinct $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$.

Theorem 1.9 (succinct $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$). *The membership problem for the succinct* $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$ *is decidable in polynomial time.*

The proof closely follows that of Theorem 1.6, but new difficulties arise. The main new difficulty is related to the fact that the role of free generators is played by the string ts. This fake free generator is not atomic: it is the concatenation of t and s. The atomicity of free generators was implicitly used in the proof of Theorem 1.6.

The main theorem is finally proved in section 6 where we give a polynomial time reduction of the membership problem for the modular group to that for the succinct $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$. That reduction is relatively simple and is essentially independent from the preceding sections.

The syllabic approach can be used to prove additional results. But this is a topic for separate papers. Here we focus on the modular group.

The intended audience for this paper is computer scientists rather than group theorists. Accordingly we do not presume the knowledge of group theory.

2. Syllabic recognizer approach: Basics and illustration.

We explain the basics of the syllabic recognizer approach and illustrate the approach by constructing a new decision algorithm for the membership problem for the free group in standard presentation. The construction is used as a template for generalization in sections 3–5.

2.1. Syllabic presentations of groups.

The peculiarity of combinatorial/geometric group theory, in comparison to abstract group theory, is that one has to deal not only with group elements but also with their representations. The standard group presentation form employs generators and relators [8, 7]. We introduce a new group presentation form.

Definition 2.1 (syllabic group presentation). *Let G be a group with a fixed finite set of generators. A syllabic presentation of G with fixed generators is given by means of four items: a finite alphabet, a set of syllables, an involution on syllables called the inverse operation, and an equality relation on the semigroup of words built from the syllables. We describe the four items in greater detail.*

- *Alphabet: The symbols of the alphabet split into two categories. First, there are letters that represent the fixed generators, one letter per generator. We call them original letters. We call the remaining symbols auxiliary. The symbols of the alphabet are linearly ordered.*
- *Syllables: Syllables are strings in the alphabet described above. We reserve the Greek letter σ to denote syllables. The number of syllables may be finite or infinite. Finite concatenations of syllables are called words. The empty string 1 is a word. The words with the concatenation operation form a semigroup called the word semigroup. It is in fact a monoid.*
- *Inverse operation: The inverse operation takes a syllable σ to another syllable σ^{-1}. It is an involution, so that $(\sigma^{-1})^{-1} = \sigma$. We extend the inverse operation to words in the obvious way: $(\sigma_1\sigma_2\ldots\sigma_k)^{-1} = \sigma_k^{-1}\ldots\sigma_2^{-1}\sigma_1^{-1}$.*
- *Equality: The equality relation is a congruence of the word semigroup with the inverse operation. In other words, it respects the concatenation operation as well as the inverse operation:*

$$(x_1 = y_1) \ \wedge\ (x_2 = y_2) \ \longrightarrow\ x_1y_1 = x_2y_2,$$
$$x_1 = y_1 \ \longrightarrow\ x_1^{-1} = y_1^{-1}$$

  for all words x_1, x_2, y_1, y_2. It is required that the quotient algebra is isomorphic to G. So two words are equal if they denote the same element of G.

Finally, notice that words are strings in the given alphabet. Accordingly the size, *or* length, *of a word is the number of (the occurrences of) alphabet symbols in the word.*

Remark 2.2 (equality vs. identity). Thus we have two competing relations on words. One is the identity of words as strings in the given alphabet, and the other is the equality of words (as names for the group elements). Notice that we saw the distinction between identity and equality already, in the very beginning of the Introduction, when we recalled the definition of $\mathrm{PSL}_2(\mathbb{Z})$. Elements of $\mathrm{PSL}_2(\mathbb{Z})$ are modular matrices, but every matrix is equated with its negative. For our purposes, it is convenient to use an equality relation that is different from the identity relation. We use a word as a name for a group element, and we don't have to mention the corresponding equivalence class of words all the time. But there is an obvious awkwardness in calling distinct words equal, and so we will have to be careful to avoid a confusion.

In the rest of this subsection we consider the free group in the standard presentation.

Definition 2.3 (standard free group). *The* standard free group *is the free group in the following syllabic presentation.*

- *Alphabet: The* alphabet *consists of original letters denoting the free generators and one auxiliary symbol* $^{-1}$. *The alphabet is linearly ordered in some way; it will play no role what order it is exactly.*
- *Syllables:* Syllables *are strings of the form* a *or* a^{-1}, *where* a *is an original letter and* $^{-1}$ *is the auxiliary symbol.*
- *Inverses: Syllables* a *and* a^{-1} *are the inverses of each other.*
- *Equality: It is the least congruence for the word semigroup with the inverse operation such that* $xx^{-1} \sim 1$ *for every word* x.

It is easy to see that the quotient of the word semigroup over the equality relation is indeed isomorphic to the free group.

A combinatorial characterization of the equality relation. If a word $x = x_1\sigma\sigma^{-1}x_2$, we say that the word x_1x_2 is obtained from x by a single cancellation. (Recall that σ ranges over syllables.) It is easy to see that words w_1 and w_2 are equal if and only if there is a sequence of words $x_0, \dots, x_\ell$ such that x_0 is w_1, x_ℓ is w_2, and for every pair x_i, x_{i+1} of successive words, one is obtained from the other by a single cancellation.

Definition 2.4 (reduced words). *A word* w *is* reduced *if no successive syllables of* w *form an inverse pair.*

Lemma 2.5 (word reduction).

1. *Every word* w *is equal to a unique reduced word. That reduced word is called the* reduct *of* w.
2. *There is a polynomial-time* word reduction algorithm *that transforms any word* w *to its reduct.*

These facts are well known [8, 7] and relatively easy to verify.

2.2. Recognizers and the construction algorithm.

2.2.1. Recognizers: Definitions.

Definition 2.6 (recognizer). *A* recognizer R *is a nondeterministic finite state automaton with the input alphabet* Σ *subject to the following conditions:*

- *Numerical states: The states are positive integers.*
- *Finiteness:* R *has only finitely many transitions.*
- *Reversibility: For every transition* (u, σ, v), *from state* u *on input symbol* σ *to state* v, *there is an inverse transition* (v, σ^{-1}, u).
- *Cyclicity: The initial state is the only final (or accepting) state.*
- *Connectivity: For every state* u, *there is a string* s *of input symbols such that the computation of* R *on* s *ends at* u.

Remark 2.7 (numerical states). What objects can serve as states of a recognizer? Algebraically speaking, the nature of the states is of no importance. Algorithmically speaking, it is important to have a reasonable representation of the states. We will use in particular the fact that the states are linearly ordered.

Definition 2.8 (recognizer's size). *The* size *of a positive integer n is the size of the standard decimal representation of n. The* size *of a transition (u, σ, v) is the sum of the sizes of the three components. The* size *of the recognizer is the sum of the sizes of its transitions.*

Intuitively the size of the recognizer is the number of characters in a reasonable representation of recognizers. The fact that we care only that our algorithms are polynomial time gives us a large freedom in defining the sizes. We could have used the unary notation for the states.

Definition 2.9 (recognizer's subgroup). *The initial state is also called the* origin *and is denoted o. The language recognized by R, that is, the set of words accepted by R, is denoted $L(R)$. It is easy to see that $L(R)$ is closed under concatenation and under the inverse operation. Define $\Gamma(R)$ to be the set of group elements w such that the word w is in $L(R)$. It is easy to see that $\Gamma(R)$ is a subgroup of G. We will say that R* recognizes *$\Gamma(R)$.*

Definition 2.10 (equivalence of recognizers). *Recognizers R_1 and R_2 are* equivalent *if $\Gamma(R_1) = \Gamma(R_2)$.*

2.2.2. A graph-theoretic view of recognizers.

A *recognizer* R over Σ can be viewed as a directed graph with a distinguished vertex o, the *origin*, where every edge is labeled with an element of Σ and parallel edges are allowed. If e is an edge with start-vertex u, end-vertex v, and label σ, then the triple (u, σ, v) is the *profile* of e. It is required that

- R has finitely many vertices and edges;
- R is strongly connected so that there is a path from any vertex u to any other vertex v;
- different edges have different profiles so that the profile of an edge identifies it uniquely; and
- for every edge $e = (u, \sigma, v)$, there is an inverse edge $e^{-1} = (v, \sigma^{-1}, u)$.

We will use both the automata-theoretic terminology as well as the graph-theoretic terminology. Some graph-theoretic terms are used in different ways by different authors. To fix some graph-theoretic terminology, we give the following definitions.

Definition 2.11 (paths). *A* path *is a sequence $\langle e_1, e_2, \ldots, e_\ell \rangle$ of edges such that the start-vertex of e_{i+1} is the end-vertex of e_i. We do not distinguish between an edge e and the single-edge path formed by e. Consider a path*

$$\pi = \langle (u_0, \sigma_1, u_1), (u_1, \sigma_2, u_2), \ldots, (u_{\ell-1}, \sigma_\ell, u_\ell) \rangle.$$

The vertex sequence *of π is $\langle u_0, u_1, \ldots, u_\ell \rangle$. The vertices u_i with $0 < i < \ell$ are* internal*. The string $\sigma_1 \sigma_2 \ldots \sigma_\ell$ is the* label *of π, and the triple $(u_0, \sigma_1\sigma_2 \ldots \sigma_\ell, u_\ell)$ is the* profile *of π. We use the $+$ sign to indicate the concatenation of paths; if a path π_1 has 7 edges and a path π_2 has 11 edges, then the path $\pi_1 + \pi_2$ has 18 edges.*

While every edge is uniquely determined by its profile, this is not necessarily true for paths. For example, you may have different paths $\langle (u_0, aa, u_1), (u_1, a, u_2) \rangle$ and $\langle (u_0, a, v), (v, aa, u_2) \rangle$ with the profile (u_0, aaa, u_2).

Definition 2.12 (branches, cycles, and nooses). *Consider a path $\pi = \langle e_1, \ldots, e_\ell \rangle$ with vertex sequence $\langle u_0, u_1, \ldots, u_\ell \rangle$, and assume that the nonfinal vertices $u_0, u_1, \ldots, u_{\ell-1}$ of π are all distinct.*

- π *is a* branch *if the final vertex* u_ℓ *differs from all nonfinal ones.*
- π *is a* cycle at u_0 *if the final vertex coincides with the start-vertex* u_0.
- π *is a* noose *if the final vertex coincides with an internal vertex* u_i.

If π *is a noose and* $u_\ell = u_i$, *then* π *splits into the* loop $\langle e_{i+1}, \dots, e_\ell \rangle$ *and the* tail $\langle e_1, \dots, e_i \rangle$ *of the noose.*

DEFINITION 2.13 (path segments). *If* u, v *are vertices on a path* π *and if there is a contiguous segment of* π *with initial vertex* u *and final vertex* v, *then* $\pi[u, v]$ *is the shortest of such segments.*

DEFINITION 2.14 (disjoint paths). *Paths* π_0 *and* π_1 *are* internally disjoint *if no internal vertex of* π_i *occurs on* π_{1-i}. *Further, paths* π_0 *and* π_1 *are* disjoint off a vertex set U *if all vertices that occur on both paths belong to* U. *If* π_0 *and* π_1 *are disjoint off* $\{\nu\}$, *we say that they are* disjoint off the vertex ν.

2.2.3. Construction algorithm.

LEMMA 2.15 (construction). *There exists a polynomial-time* construction algorithm *that, given arbitrary words* $h_1, h_2, \dots, h_m$, *constructs a recognizer* R *such that* $\Gamma(R)$ *is the subgroup generated by the group elements* $h_1, \dots, h_m$.

Proof. The desired recognizer R is a bouquet of m cycles labeled with $h_1, \dots, h_m$ and having only the initial state o in common. The desired algorithm is this. Start with a naked origin vertex o. For each generator h_j, put a cycle C_j with profile (o, h_j, o) around o. If $h_j = \sigma_1 \dots \sigma_n$ and $C_j = \langle (u_0, \sigma, u_1), \dots, (u_{n-1}, \sigma_n, u_n) \rangle$, where $u_0 = u_n = o$ and the internal vertices are new, then for each existing edge (u, σ, v), create an inverse edge (v, σ^{-1}, u). That completes the construction of R.

It is easy to see that $L(R)$ is the least set of words that contains $h_1, h_2, \dots, h_m$ and is closed under concatenation and the inverse operation. □

Example 2.16. Suppose that G is the standard free group, a, b, and c are free generators of G, $m = 2$, $h_1 = ab$, and $h_2 = c^{-1}b$. Then R consists of three vertices and eight edges. There is a unique a-labeled edge from o to a nonorigin vertex u. Call the other nonorigin vertex v. Then the edges are

$$(o, a, u), (u, b, o), (o, c^{-1}, v), (v, b, o)$$

and the inverses of these four edges. R accepts any concatenation $x_1 x_2 \dots x_\ell$, where every $x_i \in \{h_1, h_2, h_1^{-1}, h_2^{-1}\}$. In particular, R accepts $(ab)(c^{-1}b)^{-1} = ac$.

Remark 2.17 (nondeterministic algorithms). When we claim that there is a polynomial-time algorithm, we mean a deterministic algorithm, of course. But the algorithm described in the proof of the construction lemma is nondeterministic, and so the description is incomplete. What is missing is how to determinize the construction described in the proof. This easy task is left to the reader. The inessential nondeterminism of that sort allows us to simplify proofs and will be used over and over again.

2.3. Membership criterion and the reading algorithm for the standard free group.

In this section, we deal only with the standard free group. In particular the notion of fat will be redefined again and again as we deal with other syllabically presented groups.

DEFINITION 2.18 (fat). *Let* R *be a recognizer. For every syllable* σ *and every vertex* v, $\mathrm{Fat}_R(\sigma, v) = \max(0, i-1)$, *where* i *is the number of* σ*-edges from* v. *The subscript is omitted when the context uniquely defines the recognizer.*

A recognizer is deterministic if and only if every $\mathrm{Fat}(\sigma, v) = 0$.

Remark 2.19 (deterministic recognizers). When is a nondeterministic finite state automaton deterministic? There are two common definitions in the literature. The stricter definition requires that, for every label and every state u, there is exactly one transition with that label from that state. The more liberal definition requires only that, for every label and every state, there is at most one transition with that label from that state. We adopt the more liberal definition.

Lemma 2.20 (membership criterion). *Let R be a deterministic recognizer and w a word. The group element w belongs to $\Gamma(R)$ if and only if R accepts the reduct of w.*

Proof. If R accepts the reduct of w, then by the recognizer's subgroup definition in section 2.2, the group element w belongs to $\Gamma(R)$.

Now suppose that the group element w belongs to $\Gamma(R)$ so that R accepts at least one word equal to w. Consider a shortest word w_0 equal to w and accepted by R, and let π be the accepting run. The word w_0 is reduced. Otherwise π has the form

$$\pi_1 + \langle (u_1, \sigma, v), (v, \sigma^{-1}, u_2)\rangle + \pi_2.$$

Since $\text{Fat}(\sigma^{-1}, v) = 0$, we have $u_1 = u_2$. Accordingly $\pi_1 + \pi_2$ is a run that accepts a word equal to w that is shorter than w_0, which contradicts the choice of w_0. □

Lemma 2.21 (reading). *There is a polynomial-time algorithm (called the* reading algorithm *below) that, given a deterministic recognizer R and a word w, determines whether the group element $w \in \Gamma(R)$.*

Proof. Use the word reduction algorithm of section 2.1 to compute the reduct w_0 of w. By the membership criterion in section 2.3, $\Gamma(R)$ contains the group element w if and only if R accepts w_0.

To determine whether R accepts w_0, run R on w_0. Let $w_0 = \sigma_1 \dots \sigma_\ell$ and $v_0 = o$. If, after reading an initial segment $\sigma_1 \dots \sigma_i$ of w_0, R arrives to a state v_i without an outgoing edge labeled with σ_{i+1}, then R rejects w_0. If R reads all of w_0 and winds up at a vertex v_ℓ, then R accepts w_0 if and only v_ℓ is o. □

In the remaining part of this section, we construct a polynomial-time algorithm that transforms any recognizer to an equivalent deterministic recognizer.

2.4. Vertex identification and edge folding. We return to consider the general situation. Recall that if H is a subgroup of G and $g \in G$, then the set $Hg = \{hg \ : \ h \in H\}$ is a *(right) coset* of the subgroup H.

Lemma 2.22. *Let R be a recognizer and $H = \Gamma(R)$. For every two paths (o, x, v) and (o, y, v) from the origin to the vertex v, $Hx = Hy$.*

Proof. The concatenation of (o, x, v) and (v, y^{-1}, o) is an accepting run on xy^{-1}, so $xy^{-1} \in H$ and $Hx = Hy$. □

The lemma and the fact that, for every v, there is a path from o to v lead us to the following definition.

Definition 2.23 (vertex's coset). *Let R be a recognizer R and $H = \Gamma(R)$ and v be a vertex of G. Coset(v) is the set Hw, where w is the label of any path from o to v.*

Lemma 2.24 (coset stability). *For any path (u, w, v),* $\text{Coset}(v) = (\text{Coset}(u))w$.

Proof. Let π be a path (o, x, u) and ρ be the given path (u, w, v). Then $\pi + \rho$ is a path with profile (o, xw, v). We have $\text{Coset}(v) = H(xw) = (Hx)w = (\text{Coset}(u))w$. □

Example 2.16 (continuation). Because of paths (o, a, u) and (u, b, o), we have $\text{Coset}(u) = Ha = Hb^{-1}$. Because of paths (o, c^{-1}, v) and (v, b, o), we have $\text{Coset}(v) = Hc^{-1} = Hb^{-1}$. In particular $\text{Coset}(u) = \text{Coset}(v)$.

Definition 2.25 (morphisms). *Let R, S be recognizers with vertex sets U, V, respectively, operating on the same input alphabet Σ. A* morphism *from R to S is a function $\mu : \ U \to V$ satisfying the following conditions.*

- *$V = \{\mu(u) :\ u \in U\}$, and $\mu(o_R) = o_S$.*
- *S has an edge (v_1, σ, v_2) if and only if R has an edge (u_1, σ, u_2) with $\mu(u_1) = v_1$ and $\mu(u_2) = v_2$.*
- *If $\mu(u_1) = \mu(u_2)$, then* $\mathrm{Coset}(u_1) = \mathrm{Coset}(u_2)$.

LEMMA 2.26 (morphism lemma). *If there is a morphism from a recognizer R to a recognizer S, then $\Gamma(R) = \Gamma(S)$.*

Proof. Let $H = \Gamma(R)$, $I = \Gamma(S)$, and μ be a morphism from R to S. To simplify notation, $\mu(u)$ is denoted u'. Any accepting run (o, w, o) of R gives rise to an accepting run (o', w, o') of S. Thus $L(R) \subseteq L(S)$ and $H \subseteq I$. Furthermore, any run (o, w, u) of R gives rise to an accepting run (o', w, u') of S. Thus,

$$\mathrm{Coset}(u) = Hw \implies \mathrm{Coset}(u') = Iw.$$

To prove the other inclusion, it suffices to establish the opposite implication:

$$\mathrm{Coset}(v') = Iw \implies \mathrm{Coset}(v) = Hw.$$

Indeed suppose that $w \in I$. Then $\mathrm{Coset}(o') = Iw$. Hence $H = \mathrm{Coset}(o) = Hw$ and $w \in H$.

We prove the implication by induction on the number ℓ of edges in a shortest path π from o' to v'. The case $\ell = 0$ is trivial. Suppose that $\ell > 0$ so that the label of π has the form $x\sigma$ and $\mathrm{Coset}(v') = Ix\sigma$. We prove that $\mathrm{Coset}(v) = Hx\sigma$. Since μ is a morphism, the penultimate vertex on π has the form u' for some R-vertex u. We have $\mathrm{Coset}(u') = Ix$ and, by the induction hypothesis, $\mathrm{Coset}(u) = Hx$. Since μ is a morphism, there is an edge (u_0, σ, v_0) with $u'_0 = u'$ and $v'_0 = v'$. We have

$$\begin{aligned}&\mathrm{Coset}(v) = \mathrm{Coset}(v_0) = (\mathrm{Coset}(u_0))\sigma \\ &= (\mathrm{Coset}(u))\sigma = (Hx)\sigma = H(x\sigma).\end{aligned}$$

The first equality is by the morphisms definition, and the second equality is by the coset stability lemma. □

DEFINITION 2.27 (vertex identification). *Let R be a recognizer with vertices U, and let v_1, v_2 be two vertices of R such that* $\mathrm{Coset}(v_1) = \mathrm{Coset}(v_2)$. *To* identify *$v_1$ and v_2 in R means to construct a recognizer S with vertices $(U - \{v_1, v_2\}) \cup \{v\}$, where v is different from any vertex in $(U - \{v_1, v_2\})$, such that the map*

$$\mu(u) = \begin{cases} v & \textit{if } u = v_1 \textit{ or } v = v_2, \\ u & \textit{otherwise} \end{cases}$$

is a morphism from R to S.

It is easy to see that the desired S exists and is unique up to isomorphism. Let σ be any label u, be any vertex in $(U - \{v_1, v_2\})$, and i, j range over $\{1, 2\}$. S has an edge (u, σ, v) if and only if R has an edge of the form (u, σ, v_i). S has an edge (v, σ, u) if and only if R has an edge of the form (v_i, σ, u). And S has a loop v, σ, v if and only if R has an edge of the form (v_i, σ, v_j).

Remark 2.28 (vertex identification). There are several ways to implement S. One can remove v_1, v_2 and replace them with a fresh vertex v. One can remove one of vertices v_1, v_2 and use the remaining one as v. Our preferred way to implement S is this: view v_1 and v_2 as two names of the same vertex in S.

LEMMA 2.29 (vertex identification). *The identification of vertices u, v such that* $\mathrm{Coset}(u) = \mathrm{Coset}(v)$ *produces an equivalent recognizer.*

Proof. Use the vertex identification claim and the morphism lemma. □

Example 2.16 (continuation). Consider a recognizer S with two vertices o, v' and six edges:

$$(o, a, v'), (v', a^{-1}, o), (o, b^{-1}, v'), (v', b, o), (o, c^{-1}, v'), (v', c, o).$$

Since $\text{Coset}(u) = \text{Coset}(v)$, there is a homomorphism μ of R onto D_1 such that $\mu(u) = \mu(v) = v'$. By the previous lemma, S is a recognizer for H. Note that S accepts ac while R does not.

Full vertex identification, when all vertices with the same coset are identified, reduces any recognizer R to a recognizer S where distinct vertices have different cosets. By the morphism lemma, $\Gamma(S) = \Gamma(R)$. Further, S is deterministic. Indeed suppose that we have edges (u, σ, v_1) and (u, σ, v_2) in S. By the coset stability lemma, $\text{Coset}(v_1) = (\text{Coset}(u))\sigma = \text{Coset}(v_2)$, and so $v_1 = v_2$. But notice that the question of whether Hw_1 is equal to Hw_2 is equivalent to the question of whether $w_1 w_2^{-1} \in H$ which is an instance of the membership problem, the problem we want to solve in the first place. We need to get around this difficulty.

If R is a recognizer and $(u, a, v_1), (u, a, v_2)$ are edges in R, then by the coset stability lemma, $\text{Coset}(v_1) = (\text{Coset}(u))a = \text{Coset}(v_2)$ in R. This justifies the following definition.

Definition 2.30 (edge folding). *Let (u, σ, v_1) and (u, σ, v_2) be edges in a recognizer. To* fold edges (u, σ, v_1) and (u, σ, v_2) *is to identify the vertices v_1 and v_2.*

Lemma 2.31 (edge folding). *Folding edges does not change the recognizer subgroup.*

Proof. Use the vertex identification lemma. □

Example 2.16 (continuation). S is obtained from R by folding edges (o, b^{-1}, u) and (o, b^{-1}, v) together.

2.5. Fat reduction algorithm for the standard free group.

In this section, we deal exclusively with the standard free group.

Lemma 2.32 (fat reduction). *There is a polynomial-time* fat reduction algorithm *that transforms any recognizer R into an equivalent deterministic recognizer.*

Proof. The desired algorithm repeatedly folds distinct edges of the current recognizer with the same start-vertex u and the same label a until the recognizer becomes deterministic. By the edge-folding lemma, the recognizer subgroup does not change. It is easy to see that the algorithm works in polynomial time. □

The question arises whether the reducer makes all possible vertex identifications.

Lemma 2.33. *Let R be a deterministic recognizer for H. If (o, x, u) and (o, y, v) are paths with $Hx = Hy$, then $u = v$.*

It follows that distinct vertices of R have distinct cosets.

Proof. Induction on $n = |x| + |y|$. The basis of induction, when $n = 0$, is trivial. Assume that $n > 0$. Without loss of generality, x and y are reduced. Indeed, suppose that one of them, say x, is not reduced. Then x has the form $x_1 \sigma \sigma^{-1} x_2$. By the determinacy of R, there is a path $(o, x_1 x_2, u)$. We have $Hx_1 x_2 = Hx = Hy$. By the induction hypothesis, $u = v$.

If $x = 1$ so that $u = o$, then $Hy = H1 = H$ so that the group element y belongs to H. By the membership criterion, R accepts y, and so $v = o = u$. The case $y = 1$ is similar. Thus we can assume that neither word is empty.

Let $x = x'\sigma, y = y'\tau$, and let u', v' be the penultimate vertices in the paths (o, x, u) and (o, y, v), respectively. If $\sigma = \tau$, then $Hx' = Hx\sigma^{-1} = Hy\sigma^{-1} = Hy\tau^{-1} = Hy'$, and so $u' = v'$. Since R is deterministic, $u = v$. So we may assume that $\sigma \neq \tau$. Then the word xy^{-1} is reduced. Also $xy^{-1} \in H$ because $Hx = Hy$. So there is a cycle at o with label xy^{-1}. Since R is deterministic, there is a path (u, y^{-1}, o), and therefore there is a path (o, y, u). Since R is deterministic, the path (o, y, u) coincides with the path (o, y, v), and so $u = v$. □

2.6. Membership problem for the standard free group.

THEOREM 2.34. *There is a polynomial-time decision algorithm for the membership problem for the free group G. More explicitly, there is an algorithm such that*

(i) *given words $h_1, \ldots, h_m$ and w representing elements of G, the algorithm decides whether the subgroup H generated by $h_1, \ldots, h_m$ contains w, and*

(ii) *the algorithm runs in time polynomial in $|h_1| + \cdots + |h_m| + |w|$.*

Proof. Use the construction algorithm of section 2.2 to construct a recognizer R_1 for H. Use the fat reduction algorithm of section 2.5 to transform R_1 into an equivalent deterministic recognizer R_2 for H. Finally, use the reading algorithm of section 2.3 to check whether $w \in H$. Since the three algorithms are polynomial time, the decision algorithm is polynomial time as well. □

3. Succinct free group. We introduce an exponentially more succinct representation of the elements of the free group and show that the membership problem remains polynomial-time decidable.

3.1. Succinct free group: Definition and word reduction.

DEFINITION 3.1 (succinct free group). *The* succinct free group *is the free group in the following syllabic presentation.*

- *Alphabet: The alphabet consists of original letters denoting the free generators and* 11 *auxiliary symbols* ^{0}, ^{1}, *. . .* , ^{9}, *and* $^-$. *The alphabet is linearly ordered in some way; it will play no role in what order it is exactly.*
- *Syllables: Syllables are strings of the form a^i, where a is an original letter and i is a nonzero integer in decimal notation.*
- *Inverses: Syllables a^i and a^j are the inverses of each other if $i + j = 0$. To distinguish the representation of the free group from the standard representation, the new words will be called* exponent words, *and the old words will be called* unary words. *Any exponent word w expands in the obvious way to a unary word called the* unary expansion *of w. For example, $a_1^{-3}a_2^5$ expands to $a_1^{-1}a_1^{-1}a_1^{-1}a_2a_2a_2a_2a_2$.*
- *Equality: Exponent words are* equal *if their unary expansions are equal in the standard free group.*

Obviously the quotient of the word semigroup over the equality relation is isomorphic to the free group.

DEFINITION 3.2 (reduced exponent words). *An exponent word $w = a_1^{p_1} a_2^{p_2} \cdots a_k^{p_k}$ is* reduced *if every a_{i+1} differs from a_i.*

LEMMA 3.3 (exponent word reduction).

1. *Every exponent word w is equal to a unique reduced exponent word. That reduced exponent word is called the* reduct *of w.*
2. *There is a polynomial-time* exponent word reduction algorithm *that transforms any word w to its reduct.*

Proof. We describe the desired algorithm. If there are neighboring syllables a^i, a^j with the same base a, do the following. If $j = -i$, then remove the substring $a^i a^j$; otherwise, replace the substring $a^i a^j$ with the syllable a^{i+j}. Keep doing this until the exponent word is reduced. □

3.2. Vertex creation and membership criterion.

Definition 3.4 (edges). *An edge e of the form (u, a^i, v) is an a-*edge. *We assign to e the sign of i, so that e is positive (respectively, negative) if i is so. The* length *of e is the number of syllables in the unary expansion of its label a^i, so that the length of an edge is exponentially larger than the size of its label.*

Definition 3.5 (edge splitting). *We define how to* split an edge $e = (u_1, x, u_2)$ *into two edges of lengths n_1 and n_2, respectively. It is presumed that n_1 and n_2 are positive integers such that $n_1 + n_2 = n$. Let x_1 be the prefix of x of length n_1 and let x_2 be the corresponding suffix. Add a new vertex v and edges*

$$(u_1, x_1, v), (v, x_1^{-1}, u_1), (v, x_2, u_2), (u_2, x_2^{-1}, v),$$

and remove edges e and e^{-1}.

Definition 3.6 (vertex creation). *We define how to* create a new vertex *on a path $\pi = \langle e_1, \dots, e_i \rangle$ at distance n from the initial vertex u_0. It is presumed that*

$$\text{Length}(\langle e_1, \dots, e_i \rangle) < n < \text{Length}(\langle e_1, \dots, e_{i+1} \rangle)$$

for some i. Split e_{i+1} into two edges of lengths $n - \text{Length}(\langle e_1, \dots, e_i \rangle)$ and $\text{Length}(\langle e_1, \dots, e_{i+1} \rangle) - n$.

Lemma 3.7 (vertex creation). *The creation of a new vertex preserves the recognizer subgroup and the amount of fat.*

Proof. It suffices to prove that splitting an edge preserves the recognizer subgroup. Suppose that R is the given recognizer and a new recognizer S is obtained from R by splitting an edge e of R into edges e_1 and e_2. Note that $\langle e_1, e_2 \rangle$ is a path with the profile of e. The amount of fat at the new vertex is 0, and the amount of fat at any old vertex does not change, so $\text{Fat}(R) = \text{Fat}(S)$.

If $w \in L(R)$ and π is an accepting run of R on w, replace every occurrence of e in π with $\langle e_1, e_2 \rangle$, and replace every occurrence of e^{-1} in π with $\langle e_2^{-1}, e_1^{-1} \rangle$. The result is an accepting run of S on a word equal to w. Suppose that $w \in L(S)$, and let π be an accepting run of S on w. The new vertex can occur only in the context $\langle e_1, e_2 \rangle$, which can be replaced by e, or in the context $\langle e_2^{-1}, e_1^{-1} \rangle$, which can be replaced by e^{-1}. The result of the replacements is an accepting run of R on a word equal to w. □

Definition 3.8 (paths). *The* length *of a path π is the sum of the lengths of its edges. For every original letter a, an a-*path *is a nonempty path composed of a-edges. A* positive a-path *is composed of positive a-edges, and a* negative a-path *is composed of negative a-edges. A* partisan a-path *is a positive or negative a-path.*

In addition to the standard notion of acceptance, will we need a more liberal one.

Definition 3.9 (quasi transitions). *Suppose that π is a path with profile (u, x, v). If x is equal to a syllable or to 1, then π is a* quasi transition *and the syllable or the empty word 1 is the* quasi label *of π. If τ is the quasi label of π, then (u, τ, v) is the* quasi profile *of π.*

Every a-path π is a quasi transition. If $\text{Label}(\pi) = a^{i_1} a^{i_2} \dots a^{i_k}$, then the quasi label of π is a^j, where $j = i_1 + \dots + i_k$.

Example 3.10. If q_1, q_2 are paths

$$\langle (o, a^2, u_1), (u_2, a^{-3}, u_2), (u_3, a^5, u_3) \rangle,$$
$$\langle (u_3, b^{-7}, u_4), (u_4, b^{11}, u_5), (u_5, b^{-13}, o) \rangle,$$

respectively, then q_1 is a quasi transition from o to u_3 with quasi-label a^4, and q_2 is a quasi transition from u_3 to o with quasi-label b^{-9}. According to the paths definition in section 2.2, the labels of q_1 and q_2 are $a^2a^{-3}a^5$ and $b^{-7}b^{11}b^{-13}$, respectively.

DEFINITION 3.11 (quasi runs). *A sequence Q of quasi-transitions $q_1, \dots, q_\ell$ is a* quasi run *if the initial vertex of q_1 is o, every q_{i+1} starts at the final state of q_i, and the final vertex of q_ℓ is o. The* label *of Q is the concatenation of the quasi labels of the constituent quasi transitions. The concatenation $q_1 + \cdots + q_\ell$ of the paths $q_1, \dots, q_\ell$ is the* associate run *of Q.*

Our definition of quasi runs is narrow in the sense that the initial state o is the start and end of any quasi run. We will not need more general quasi-runs.

COROLLARY 3.12 (quasi-run labels). *Let Q be a quasi run and π be the associate run. Then π is an accepting run, and the label of π is equal, as a group element, to the label of Q.*

DEFINITION 3.13 (tolerance). *A recognizer* tolerates *an exponent word w if there is a quasi run with label w.*

Example 2.16 (continuation). The recognizer tolerates a^4b^{-9}. This is witnessed by the sequence $Q = \langle q_1, q_2 \rangle$. The concatenation $q_1 + q_2$ of the paths q_1, q_2 is the run

$$(o, a^2, u_1), (u_2, a^{-3}, u_2), (u_3, a^5, u_3), (u_3, b^{-7}, u_4), (u_4, a^{11}, u_5), (u_5, a^{-13}, o)$$

that accepts the word $a^2a^{-3}a^5b^{-7}b^{11}b^{-13}$ equal to a^4b^{-9} in G.

LEMMA 3.14 (membership criterion). *Consider a recognizer R, and let w be an exponent word. The following are equivalent:*

1. *The group element w belongs to $\Gamma(R)$.*
2. *R tolerates the reduct of w.*

Proof.

2 → 1: Suppose that R tolerates w. Then w is the label of a quasi run of Q. By the quasi-run labels corollary, the associate run of Q is accepting, and its label is equal to w. Therefore $w \in \Gamma(R)$.

1 → 2: Suppose that the group element w belongs to $\Gamma(R)$. Without loss of generality, $w \neq 1$. By the recognizer's subgroup definition in section 2.2, R accepts an exponent word w' equal to w. Since every accepting run is also a quasi run, R tolerates w'. Let $Q = \langle q_1, \dots, q_\ell \rangle$ be a quasi run with the fewest number of quasi transitions that tolerates an exponent word w_0 equal to w. Due to the choice of Q, every quasi label (q_i) is a syllable (rather than 1). Accordingly w_0 has the form $a_1^{p_1} \dots a_\ell^{p_\ell}$. We show that w_0 is reduced.

If w_0 is not reduced, then $a_{i+1} = a_i$ for some i. Let Q' be the quasi run obtained from Q by replacing $\langle q_i, q_{i+1} \rangle$ with a single quasi-transition $q_i + q_{i+1}$. The label of Q' is equal to w_0 but has fewer syllables, which contradicts the choice of Q. □

3.3. Reading algorithm.

DEFINITION 3.15 (fat). *Let R be a recognizer. For every original letter a and every vertex v,*

- $\mathrm{Fat}_R(a^+, v) = \max(0, p-1)$*, where p is the number of positive a-edges from v;*
- $\mathrm{Fat}_R(a^-, v) = \max(0, n-1)$*, where n is the number of negative a-edges from v;*
- $\mathrm{Fat}_R(a, v) = \max(0, p-1) + \max(0, n-1)$*;*
- $\mathrm{Fat}_R(v) = \sum_a \mathrm{Fat}_R(a, v)$ *and* $\mathrm{Fat}(R) = \sum_v \mathrm{Fat}(v)$.

The subscript is omitted if the context uniquely defines the recognizer.

DEFINITION 3.16 (lean recognizers). *A recognizer is* lean *if, for every original letter a and every vertex v, there is at most one positive a-edge and at most one negative a-edge coming from v.*

Obviously, lean recognizers are deterministic. However deterministic recognizers are not necessarily lean. A deterministic recognizer may have two positive a-edges coming from the same vertex if their labels are distinct syllables. Recall that inputs are syllables, not letters.

Lemma 3.17 (partisan quasi transitions).

(A) *If a lean recognizer R tolerates a reduced exponent word w, then there is a quasi-run Q with label w such that every constituent quasi transition of Q is partisan.*

(B) *Let R be a lean recognizer. For any state u and syllable a^j, there is at most one partisan quasi-transition q from u with quasi-label a^j.*

(C) *There is a polynomial-time algorithm that, given a lean recognizer R, a state u of R, and a syllable a^j, determines whether there exists a partisan quasi-transition q from u with quasi-label a^j and, if yes, finds an appropriate q.*

Proof. (A) Suppose that R tolerates a reduced exponent word $w = a_1^{p_1} \cdots a_k^{p_k}$. By the tolerance definition, there is a quasi run with label w. Let Q be a quasi-run $\langle q_1, \ldots, q_k \rangle$ with label w such that the associate run of Q is as short as possible. If some q_i is not partisan, then it has successive a_i-edges e, and f of different signs. The end-vertex of e has outgoing a_i-edges e^{-1} and f. Since R is lean, $e^{-1} = f$, and therefore both edges can be removed from q_i without changing the quasi profile of q. This contradicts the choice of Q.

(B) Assume that $j > 0$; the case $j < 0$ is similar. By contradiction assume that there exist distinct partisan quasi transitions with the same quasi-label a^j from u. Then we have two distinct positive a-paths of the same length from the same vertex u. Since neither path can be a prefix of the other, there is a vertex v where the two paths branch out. Then there are two distinct positive a-edges from v, which contradicts the leanness of R.

(C) We describe the desired algorithm. Assume $j > 0$; the case $j < 0$ is similar. Let $u_0 = u$. Due to the fact that R is lean, for any nonnegative integer k, there is at most one positive a-path π_k of the form

$$\langle (u_0, a^{p_1}, u_1), (u_1, a^{p_2}, u_2), \ldots, (u_{k-1}, a^{p_k}, u_k) \rangle.$$

Let k be the least number such that $\mathrm{Length}(\pi_k) \geq j$ or else π_k does not exist. If $\mathrm{Length}(\pi_k) = j$, then π_k is the desired quasi transition; otherwise, the desired quasi transition does not exist.

From the complexity point of view, one may worry about the case in which the vertices u_i are not distinct. In such a case, consider the very first vertex repetition on π_k. Let u_i be the first π_k vertex on the loop, $\ell = \mathrm{Length}(\pi_k[u_0, u_i])$, and m be the length of the loop. The problem reduces to the case without vertex repetition where u_i plays the role of u_0 and $(j - \ell) \mod m$ plays the role of j. □

Lemma 3.18 (reading). *There is a polynomial-time* reading algorithm *that, given a lean recognizer R and an exponent word w, determines whether the group element w belongs to $\Gamma(R)$.*

Proof. Use the exponent word reduction algorithm of section 3.1 to compute the reduct $w_0 = a_1^{p_1} \ldots a_k^{p_k}$ of w. By the membership criterion of section 3.2, $\Gamma(R)$ contains the group element w if and only if R tolerates w_0. We need to determine whether there is a quasi-run Q with label w_0.

By the partisan quasi-transitions lemma, part (A), we may restrict attention to quasi-runs $Q = \langle q_1, \ldots, q_k \rangle$ such that every q_i is partisan. By part (B) of the same lemma, there is at most one such Q. Let A be the algorithm of part (C) of the same lemma.

The reading algorithm iterates A and has at most k rounds. Assume that the reading algorithm has performed i rounds and in the process has constructed an initial segment $\langle q_1, \ldots, q_i\rangle$ of the desired quasi-run Q. The assumption trivially holds in case $i = 0$. If $i = 0$, let $u_0 = o$; otherwise, let u_i be the end-vertex of q_i.

In case $i < k$, the reading algorithm starts round $i+1$ by applying A to R, u_i, and $a_{i+1}^{p_{i+1}}$. If A determines that there is no partisan quasi transition with a quasi label from u_i with a label of the form $a_{i+1}^{p_{i+1}}$, then the desired quasi-run Q does not exist. Otherwise let q_{i+1} be the partisan quasi transition constructed by A.

In case $i = k$, the desired quasi-run Q exists if and only $u_k = o$. □

In the remaining part of this section, we construct a polynomial-time algorithm that transforms any recognizer to an equivalent lean recognizer.

3.4. Path folding. In section 2 we used edge folding to reduce a recognizer to a lean one. Now the situation is more involved, and edge folding is not going to do the job. Instead we will be folding paths.

Recall the branches, cycles, and nooses definition in section 2.2, and fix an original letter a.

Definition 3.19 (impasse). *A nonempty partisan a-path is an* impasse *if it is a branch and its end-vertex has no outgoing a-edges of the sign of π.*

Definition 3.20 (closed path). *A nonempty partisan a-path is* closed *if it is an impasse, a cycle, or a noose.*

Lemma 3.21 (closed path). *Every a-edge e_1 gives rise to a closed partisan a-path $\pi = \langle e_1, \ldots, e_k\rangle$ that continues e_1. Furthermore, there is an algorithm that constructs such a path in a time polynomial in the recognizer's size.*

Proof. The desired algorithm is iterative; we describe one round of it. Suppose that we have already an a-path $\langle e_1, \ldots, e_i\rangle$ with vertex sequence $\langle u_0, \ldots, u_i\rangle$. If $u_i \in \{u_0, \ldots, u_{i-1}\}$ or if u_i has no outgoing a-edge of the same sign as e_1, then halt. Otherwise let e_{i+1} be the lexicographically first a-edge from u_i of the same sign as e_1. □

As usual g.c.d.(m, n) is the greatest common divisor of positive integers m and n. Define $m = n \mod \infty$ to be equivalent to $m = n$.

Definition 3.22 (two-path divisor). *For $i = 1, 2$, let π_i be a branch or cycle of length ℓ_i. We define the* divisor *as follows:*

$$\mathrm{Div}(\pi_1, \pi_2) = \begin{cases} \infty & \textit{if } \pi_1, \pi_2 \textit{ are branches,} \\ \ell_i & \textit{if } \pi_i \textit{ is a cycle and } \pi_{3-i} \textit{ is a branch,} \\ \text{g.c.d.}(\ell_1, \ell_2) & \textit{if } \pi_1, \pi_2 \textit{ are cycles.} \end{cases}$$

Definition 3.23 (entanglement). *Suppose that π and ρ are nonempty partisan a-paths of the same sign and with the same initial vertex ν, and suppose that either path is a branch or a cycle. The two paths are* entangled *if there exist vertices u and v on π and ρ, respectively, such that $u \neq v$ and*

$$\mathrm{Length}(\pi[\nu, u]) = \mathrm{Length}(\rho[\nu, v]) \mod \mathrm{Div}(\pi, \rho).$$

Otherwise the two paths are disentangled.

Corollary 3.24 (entanglement algorithm). *There is a polynomial-time algorithm that, given a recognizer and two paths π and ρ as in the entanglement definition, determines whether they are entangled. Furthermore, if the paths are entangled, then the algorithm produces vertices u and v witnessing the entanglement and identifies them.*

We omit the pretty obvious proof.

Lemma 3.25 (entanglement). *If π and ρ are entangled and vertices u and v witness the entanglement, then the identification of u and v does not change the recognizer subgroup.*

Proof. Let $d = \mathrm{Div}(\pi, \rho)$, $k = \mathrm{Length}(\pi[\nu, u])$, $\ell = \mathrm{Length}(\rho[\nu, v])$, and $H = \mathrm{Coset}(\nu)$. We assume that π and ρ are positive; the negative case is similar. By the coset stability lemma in section 2.4, $\mathrm{Coset}(u) = Ha^k$, and $\mathrm{Coset}(v) = Ha^\ell$. By the vertex identification lemma in section 2.4, it suffices to prove that $Ha^k = Ha^\ell$. Since u and v witness the entanglement, we have $k = \ell \mod d$.

Case 1: Both paths are branches. Then $k = \ell$, and therefore $Ha^k = Ha^\ell$.

Case 2: One of the paths is a branch, and the other is a cycle. Without loss of generality, π is a circle, and ρ is a branch. Then $Ha^k = H$, and $\ell = p \cdot k$ for some integer p. Then $Ha^\ell = H(a^k)^p = H = Ha^k$.

Case 3: Both paths are cycles. Then $Ha^k = Ha^\ell = H$. Since $d = \text{g.c.d.}(k, \ell)$, there are integers i and j such that $ik + j\ell = d$, and so $Ha^d = H(a^k)^i(a^\ell)^j = H$. Since $k = \ell \mod d$, there is an integer p such that $k = pd + \ell$. Then $Ha^k = H(a^d)^p a^\ell = Ha^\ell$. ∎

Recall the disjoint paths definition in section 2.2.

Definition 3.26 (path folding). *Suppose that π and ρ are nonempty partisan a-paths such that*

- *they have the same sign and the same start-vertex ν,*
- *either path is a branch or a cycle,*
- $\mathrm{Length}(\pi) \geq \mathrm{Length}(\rho)$ *if both paths are branches, and*
- $\mathrm{Length}(\pi) = \mathrm{Div}(\pi, \rho)$ *if both paths are cycles.*

To fold ρ into π, *execute the following* folding algorithm:

1. *Apply the entanglement algorithm to π and ρ. If the number of vertices is reduced, then halt. Otherwise the paths are disentangled.*
2. *For each v on ρ, create a new vertex v' on π such that*

$$\mathrm{Length}(\pi[\nu, v']) = \mathrm{Length}(\rho[\nu, v]) \mod \mathrm{Div}(\pi, \rho)$$

   unless π has a vertex at this position already.
3. *Identify every clone v' with its original v.*
4. *Remove all edges of ρ and their inverses.*

One of our referees suggested "path merging" instead of "path folding."

Remark 3.27 (path folding). Concerning stage 2 of the folding algorithm, consider the case in which π has a vertex u such that $\mathrm{Length}(\pi[\nu, u]) = \mathrm{Length}(\rho[\nu, v]) \mod \mathrm{Div}(\pi, \rho)$. Since π and ρ are disentangled (otherwise we would not arrive at stage 2), we have $u = v$. Assume that π and ρ are internally disjoint. Then vertex v is an extreme vertex on both paths. Consider the scenario in which v differs from the initial vertex ν. Then v is the final vertex of both π and ρ, both π and ρ are branches, and $\mathrm{Length}(\pi) = \mathrm{Length}(\pi[\nu, u]) = \mathrm{Length}(\rho[\nu, v]) = \mathrm{Length}(\rho)$.

Lemma 3.28 (path folding). *Let π and ρ be as in the path folding definition. Folding ρ into π preserves the recognizer subgroup. If the algorithm halts at stage* 1*, then the number of vertices of the recognizer decreases. Otherwise the number of vertices is unchanged, and the (amount of) fat changes as follows.*

(BB) *Suppose that π and ρ are branches of lengths m and n, respectively.*
If $m = n$, then the fat decreases by 2.
If $m > n$ and ρ is an impasse, then the fat decreases by 1.
If $m > n$ but ρ is not an impasse, then the fat does not change.

(CB) *Suppose that π is a cycle and ρ is a branch.*
If ρ is an impasse, then the fat decreases by 1*;*
otherwise, the fat does not change.

(CC) *Suppose that π and ρ are cycles. Then the fat decreases by* 2.

Proof. We consider only the case in which π and ρ are positive; the case in which they are negative is similar.

Let R be the given recognizer, and for $p \leq 4$, let R_p be the recognizer obtained from R by executing p stages of the folding algorithm, so that $R_0 = R$. Let the vertex sequence of π be $\langle u_0, \ldots, u_k \rangle$. Let $\rho = \langle f_1, \ldots, f_\ell \rangle$ and the vertex sequence of ρ be $\langle v_0, \ldots, v_\ell \rangle$. Let $d = \text{Div}(\pi, \rho)$. The case in which π and ρ are entangled is obvious. Assume that π and ρ are disentangled. Then nothing happens at stage 1, and so $R_1 = R_0$.

First we note that the vertices of R_4 are those of R_1. Indeed all vertices v'_j created at stage 2 are identified with the respective vertices v_j at stage 3; thus, the vertices of R_3 are those of R_1. And the vertices do not change at stage 4.

Second we show that $\Gamma(R_4) = \Gamma(R)$. By the vertex creation lemma, $\Gamma(R_2) = \Gamma(R_1)$. By the vertex identification lemma of section 2.4, $\Gamma(R_3) = \Gamma(R_2)$. It remains to show that $\Gamma(R_4) = \Gamma(R_3)$. Let $n_j = \text{Length}(\pi[v_0, v_j])$, $r_j = n_j \mod d$, and $v'_0 = u_0$. It suffices to show that for every edge f_j of ρ, R_4 has a path P_j with the profile of f_j. Indeed, suppose we have the desired paths P_j. Recall that the profile of a path includes not only the label but also the initial and final vertices, so the desired paths P_j match up appropriately to simulate ρ. It is easy to see that, for every vertex v, Coset(v) computed in R_3 is the same as the one computed in R_4.

The profile of f_j is (v_{j-1}, a^p, v_j), where $p = n_j - n_{j-1}$. In scenario (BB), the desired path P_j is $\pi[v'_{j-1}, v'_j]$. In scenarios (CB) and (CC), $p = d \cdot q + (r_{j+1} - r_j)$ for some q. The desired path P_j starts at v'_{j-1} and ends at v'_j. If $r_j \leq r_{j+1}$, then π does q full revolutions around π. If $r_j > r_{j+1}$, then $q > 0$, and π does $q - 1$ full revolutions around π.

Finally we prove the claims about the fat. By the vertex creation lemma, Fat$(R_2) =$ Fat(R_1). Thus we need to examine only the evolution of the fat from R_2 to R_4. Furthermore, it suffices to examine the evolution of the numbers Fat(a, v_j). Indeed, this will account for the vertices v'_j identified with the corresponding vertices v_j at stage 3. If a vertex v of R_2 differs from any v_j, v'_j, then the immediate vicinity of v does not change. If an original letter $b \neq a$, then Fat(b, v_j) does not change. The reason is that, upon the creation, the vertex v'_j has no b-edges adjacent to it. As a result the identification of v_j with v'_j creates no b-fat.

If $0 < j < \ell$, then Fat(a, v_j) does not change from R_2 to R_4. Indeed, as a result of identification with v'_j, the vertex v_j acquires one outgoing positive a-edge and one outgoing negative a-edge at stage 3, but then, at stage 4, it loses one outgoing positive a-edge, namely, f_{j+1}, and one outgoing negative a-edge, namely, f_j^{-1}.

The vertex v_0 does not acquire any outgoing edges at stage 3, and thus neither Fat(a^+, v_0) nor Fat(a^-, v_0) increase at stage 3. It loses one positive outgoing edge, namely, f_0 at stage 4, and so Fat(a^+, v_0) decreases by 1 from R_2 to R_4. It does not lose any negative outgoing edge in scenarios (BB) or (CB), and so Fat(a^-, v_0) does not change in scenarios (BB) and (CB). Since $v_0 = v_\ell$ in scenario (CC), it remains to examine only the evolution of the numbers Fat(a^+, v_ℓ) and Fat(a^-, v_ℓ) in the three scenarios.

(BB) In this scenario, we first suppose that $m = n$. Since π and ρ are disentangled, $u_k = v_\ell$. The vertex v_ℓ does not acquire any outgoing edges at stage 3 and loses only one outgoing edge, namely, the negative edge f_l^{-1}, at stage 4. Thus Fat(a^+, v_ℓ)

does not change, and $\mathrm{Fat}(a^-, v_\ell)$ decreases by 1. To summarize, $\mathrm{Fat}(a^+, v_0)$ and $\mathrm{Fat}(a^-, v_\ell)$ decrease by 1 while $\mathrm{Fat}(a^-, v_0)$ and $\mathrm{Fat}(a^+, v_\ell)$ do not change. Hence $\mathrm{Fat}(R_4) = \mathrm{Fat}(R_2) - 2$.

Second we suppose that $m > n$. At stage 3, v_ℓ acquires one positive and one negative outgoing edge. At stage 4, v_ℓ loses no outgoing positive edges but loses f_ℓ^{-1}. Thus $\mathrm{Fat}(a^-, v_\ell)$ does not change. If ρ is not an impasse, then $\mathrm{Fat}(a^+, v_\ell)$ increases by 1; otherwise, $\mathrm{Fat}(a^+, v_\ell)$ remains zero throughout the process. We summarize. If ρ is an impasse, then $\mathrm{Fat}(a^+, v_0)$ decreases by 1 while $\mathrm{Fat}(a^-, v_0)$, $\mathrm{Fat}(a^+, v_\ell)$, and $\mathrm{Fat}(a^-, v_\ell)$ do not change, so that $\mathrm{Fat}(R_4) = \mathrm{Fat}(R_2) - 1$. If ρ is not an impasse, then $\mathrm{Fat}(a^+, v_0)$ decreases by 1, $\mathrm{Fat}(a^+, v_\ell)$ increases by 1, and $\mathrm{Fat}(a^-, v_0)$ and $\mathrm{Fat}(a^-, v_\ell)$ do not change, so that $\mathrm{Fat}(R_4) = \mathrm{Fat}(R_2)$.

(CB) This scenario is similar to the case $m > n$ of scenario (BB). Let us just point out that the vertex v_ℓ does not occur on π in R. Indeed suppose the opposite. Since π and ρ are internally disjoint and $u_k = u_0$, we have $v_\ell = u_0 = v_0$. But then ρ is a cycle which contradicts scenario (CB).

(CC) In this scenario, $v_0 = v_\ell$, and thus $\mathrm{Fat}(a^+, v_\ell)$ decreases by 1. $\mathrm{Fat}(a^-, v_\ell)$ does not change at stage 3 and decreases by 1 at stage 4 because f_ℓ^{-1} is removed. To summarize, the overall change in the fat is this: both $\mathrm{Fat}(a^+, v_\ell)$ and $\mathrm{Fat}(a^-, v_\ell)$ decrease by 1. Thus $\mathrm{Fat}(R_4) = \mathrm{Fat}(R_2) - 2$. □

3.5. Weight reduction algorithm.

DEFINITION 3.29 (recognizer weight). *The* weight *of a recognizer R is a pair (i, j) of natural numbers where i is the number of vertices of R and $j = \mathrm{Fat}(R)$. The weights are ordered lexicographically with the number of vertices being the more significant component.*

LEMMA 3.30 (weight reduction). *There is a polynomial-time* weight reduction algorithm *that reduces any recognizer to an equivalent lean recognizer.*

Proof. We construct an iterative algorithm that transforms the given recognizer by means of path folding; the algorithm halts when the recognizer is lean. By the folding lemma, the algorithm preserves the recognizer subgroup.

We describe one round of the algorithm and show that the weight decreases at each round. It will be obvious that the algorithm is polynomial time.

If the current recognizer R is lean, halt. Otherwise find a quadruple (a, ξ, e_1, f_1), where a is an original letter, ξ is a vertex with $\mathrm{Fat}(a, \xi) > 0$, and e_1 and f_1 are two a-edges from ξ of the same sign. It could be the lexicographically first such quadruple, for example. We consider only the case in which e_1 and f_1 are positive; the case in which they are negative is similar. Use the algorithm of the closed paths lemma to construct closed a-paths E and F continuing the e_1 and e_2, respectively. We consider first the cases in which E and F are disjoint off ξ and then the other cases.

Part 1: Assume that E and F are disjoint off ξ.

Case 1: E and F are cycles.

If the cycles are of different lengths, let π be the shorter one; otherwise, let π be either of the cycles. Let ρ be the other cycle, $m = \mathrm{Length}(\pi)$, and $n = \mathrm{Length}(\rho)$. If m divides n, fold ρ into π. Otherwise, let d be the greatest common divisor of m and n. Create a positive single-edge a-cycle λ of length d at ξ. Then fold π into λ, and let π' be the resulting cycle of length d. Then fold ρ into π'.

To examine the evolution of weight, we apply scenario (CC) of the folding lemma to the following two subcases.

First suppose that m divides n. If π and ρ are entangled, then the number of vertices drops. Otherwise the number of vertices is unchanged, but the fat decreases by 2.

Second suppose that m does not divide n. If π' and ρ are entangled, then the number of vertices drops. Suppose that π' and ρ are disentangled. With the creation of λ, the vertex ξ acquires one outgoing positive a-edge and one outgoing negative a-edge, so that the fat increases by 2. The folding of π into λ decreases the fat by 2. The folding of ρ into π' decreases the fat by 2. Altogether the fat decreases by 2.

Case 2: One of the paths E and F is a cycle, and the other is an impasse.

Fold the impasse into the cycle. By scenario (CB) of the folding lemma, this decreases the recognizer weight.

Case 3: One of the paths E and F is a cycle, and the other is a noose.

Let π be the cycle, ρ be the noose, and v be the end-vertex of ρ. Fold the tail of ρ into π, and let π' be the resulting cycle. Note that v occurs on π'. By scenario (CB) of the folding lemma, this does not increase the recognizer weight. If the weight dropped, then finish the round. Otherwise let π'' be the cycle at v obtained from π' by redefining the initial vertex as v. The loop of ρ is another cycle at v. Clearly the two cycles at v are disjoint off ξ. Proceed as in case 1.

Case 4: E and F are impasses.

If the impasses are of different length, let π be the longer one; otherwise, let π be either of the impasses. Let ρ be the other impasse. Fold ρ into π. By scenario (BB) of the folding lemma, this decreases the recognizer weight.

Case 5: One of the paths E and F is an impasse, and the other is a noose.

Let π be the impasse, ρ the noose, τ the tail of ρ, λ the loop of ρ, $m = \text{Length}(\pi)$, and $n = \text{Length}(\tau)$. If $m \leq n$, then fold π into τ; by scenario (BB) of the folding lemma, this decreases the weight. Suppose that $m > n$. Then fold τ into π, and let π' be the resulting impasse. By the folding and weight corollary, this does not increase the weight. If the weight decreases, finish the round. Otherwise let v be the final vertex of τ and π'' be the suffix π' with initial vertex v. Obviously, λ is a cycle at v, π'' is an impasse with initial vertex v, and the two paths are disjoint off v. Proceed as in case 2.

Case 6: E and F are nooses.

Let π be the noose with a shorter tail. If the two tails are of the same length, let π be either noose. Let ρ be the other noose and u be the final vertex of π. Fold the tail of π into the tail of ρ, and let R' be the resulting recognizer. By the folding and weight corollary, $\text{Weight}(R') \leq \text{Weight}(R)$. If $\text{Weight}(R') < \text{Weight}(R)$, finish the round. Otherwise let π' be the loop of π and ρ' be the suffix of ρ with initial vertex u. π' is a cycle at u, ρ' is either a cycle at u or a noose with initial vertex u, and the two paths are disjoint. If ρ' is a cycle, proceed as in case 1. If ρ' is a noose, proceed as in case 3.

Part 2: Assume that E and F are not disjoint off ξ.

Case 7: At least one of the paths E and F is a cycle.

Let π be the cycle or one of the two cycles, and let ρ be the maximal initial segment of the other path that is internally disjoint from π. The final vertex v of ρ splits π into two segments $\pi_1 = \pi[\xi, v]$ and $\pi_2 = \pi[v, \xi]$. Without loss of generality, we may assume $\text{Length}(\pi_1) \geq \text{Length}(\rho)$; otherwise, swap π_1 and ρ in the remainder of this case. Let $m = \text{Length}(\pi_1)$ and $n = \text{Length}(\rho)$.

Fold ρ into π_1, and let R' be the resulting recognizer. If the weight decreases, then end the current round of the algorithm. Suppose that the weight does not decrease. Then, by the folding lemma, $m > n$. We have two internally disjoint cycles at v in R'. One is formed by the suffix of length $m - n$ of π_1. The other is formed by the concatenation of π_2 and the prefix of length n of π_1. Proceed as in case case 1.

Case 8: At least one of the paths E and F is a noose.

Let τ be a noose or one of the two nooses, τ_1 and τ_2 be the tail and loop of τ, respectively, and v be the final vertex of τ_1. Recall that every path P gives rise to a reverse path P^{-1}. Let π be the cycle τ_2^{-1} at v. By an argument similar to the proof of the closed path lemma, there is a closed path ρ that continues the path τ_1^{-1}. Thus we have two closed paths, π and ρ, sharing the same initial vertex v and having the same sign (that is both positive or both negative). If π and ρ are internally disjoint, proceed according to the appropriate case of part 1. Otherwise proceed as in case 7.

Case 9: Both E and F are impasses.

Let F_1 be the maximal initial segment of F disjoint from E. The final vertex v of F_1 splits E into the prefix $E_1 = E[\xi, v]$ and the corresponding suffix E_2. If $\text{Length}(E_1) \geq \text{Length}(F_1)$, let $\pi = E_1$ and $\rho = F_1$; otherwise, let $\pi = F_1$ and $\rho = E_1$. Let $m = \text{Length}(\pi)$ and $n = \text{Length}(\rho)$. Fold ρ into π. If the recognizer weight decreases, then finish the round of the algorithm. Suppose that the weight does not decrease. Then, by the folding lemma, $m > n$. We have two internally disjoint closed paths of the same sign with initial vertex v. One is the cycle formed by a suffix of length $m - n$ of π. The other is the impasse E_2. Proceed as in case 2. This completes the proof of the lemma. □

3.6. The theorem. We restate the free groups with elements in exponent normal form theorem formulated in section 1.

THEOREM 3.31. *There is a polynomial-time decision algorithm for the membership problem for the succinct free group. More explicitly, there is an algorithm such that*

(i) *given exponent words $h_1, \dots, h_m$ and w, the algorithm decides whether the subgroup H generated by $h_1, \dots, h_m$ contains w, and*

(ii) *the algorithm runs in time polynomial in $|h_1| + \cdots + |h_m| + |w|$.*

Proof. Use the construction algorithm of section 2.2 to produce a recognizer R_1 for H. Then use the weight reduction algorithm of section 3.5 to transform R_1 into a lean recognizer R_2. Finally use the reading algorithm of section 3.3 to check whether the group element w belongs to H. Since all the constituent algorithms are polynomial time, the decision algorithm is polynomial time. □

4. The free product of $(\mathbb{Z}/2\mathbb{Z})$ and $(\mathbb{Z}/3\mathbb{Z})$. This auxiliary section aims to clarify certain aspects of the membership problem for the modular group unrelated to the matrix representation of the modular group.

In the notation of combinatorial group theory, $\langle g \mid g^n \rangle$ is a cyclic group of order n with generator g. It is isomorphic to the additive group $\mathbb{Z}/n\mathbb{Z}$ of integers modulo n. The modular group is isomorphic to the free product $(\mathbb{Z}/2\mathbb{Z})*(\mathbb{Z}/3\mathbb{Z})$; see [8] for a clear explanation of this fact. We use the syllabic recognizer approach to give a polynomial time decision procedure for the membership problem for the group $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$ in the following presentation.

DEFINITION 4.1 (standard $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$). Standard $\langle s \mid s^2 \rangle * \langle t \mid t^3 \rangle$ *is the free group in the following syllabic presentation.*

- *Alphabet: The alphabet consists of original letters s, t and one auxiliary symbol $^{-1}$. The alphabet is linearly ordered in some way; it will play no role in what order it is exactly.*
- *Syllables:* Syllables *are s, t, and t^{-1}. The Greek letter ε is reserved to range over $\{1, -1\}$, so that t^ε is always either t or t^{-1}.*
- *Inverses: Syllables t and t^{-1} are the inverses of each other, and s is its own inverse.*

- *Equality: It is the least congruence for the word semigroup with the inverse operation such that $ss = 1, tt^{-1} = 1, t^{-1}t = 1, tt = t^{-1}$, and $t^{-1}t^{-1} = t$.*

It is easy to see that the quotient of the word semigroup is indeed isomorphic to $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$. Consider the five equalities that appear in the definition of the equality relation. Reading each of the five equalities from left to right gives us five different reduction steps. It is easy to see that any two words w_1, w_2 are equal if and only if there is a *witness sequence* of words $x_0, x_1, \ldots, x_\ell$ such that $x_0 = w_1$, $x_\ell = w_2$ and, for every two successive words x_i, x_{i+1}, one is obtained from the other by a single reduction step.

A word w is *reduced* if it does not contain any of the "forbidden" contiguous substrings $ss, tt^{-1}, t^{-1}t, tt$, or $t^{-1}t^{-1}$. The group G in consideration is the quotient of W with respect to the equality relation.

LEMMA 4.2 (word reduction).

1. *For every word w, there is a unique reduced word equal to w. The reduced word is the* reduct *of w.*
2. *There is a polynomial-time* word reduction algorithm *that, given a word w, produces the reduct of w.*

We omit the proof of this well-known fact.

DEFINITION 4.3 (irregular paths). *There are two kinds of irregular paths.*

- *An* s-irregular path *has the form $\langle(u_0, s, u_1), (u_1, s, u_2)\rangle$, where $u_0 \neq u_2$.*
- *An* t-irregular path *has the form $\langle(u_0, t, u_1), (u_1, t, u_2), (u_2, t, u_3)\rangle$, where $u_0 \neq u_3$.*

DEFINITION 4.4 (regular recognizers). *A recognizer without irregular paths is* regular.

LEMMA 4.5 (irregular paths). *The identification of the end-vertices of an irregular path produces an equivalent recognizer.*

Proof. Due to the vertex identification lemma of section 2.4, it suffices to show that the end-vertices of the given irregular path π have the same associated coset. To this end we use the coset stability lemma of section 2.4. If π has the form $\langle(u, s, u_1), (u_1, s, v)\rangle$, then $\mathrm{Coset}(v) = \mathrm{Coset}(u)ss = \mathrm{Coset}(u)$. And if π has the form $\langle(u, t, u_1), (u_1, t, u_2), (u_2, t, v)\rangle$, then $\mathrm{Coset}(v) = \mathrm{Coset}(u)ttt = \mathrm{Coset}(u)$. □

DEFINITION 4.6 (fat). *Let R be a regular recognizer and u be a state of R. For every syllable σ, $\mathrm{Fat}_R(\sigma, u) = \max(0, n - 1)$, where n is the number of σ-labeled transitions from u. The subscript may be omitted when the recognizer is uniquely determined by the context.*

A recognizer R is deterministic if every $\mathrm{Fat}_R(\sigma, u) = 0$. Note that a deterministic recognizer does not have s-irregular paths.

DEFINITION 4.7 (triangles). *A* triangle *in a recognizer is a cycle of the form $\langle(u_1, t, u_2), (u_2, t, u_3), (u_3, t, u_1)\rangle$. We allow the possible "degenerate" cases where some of the edges coincide. In particular, if $e = (v, t, v)$, then $\langle e, e, e\rangle$ is a triangle. An* incomplete triangle *is a two-edge path $\langle(u_1, t, u_2), (u_2, t, u_3)\rangle$ such that there is no edge (u_3, t, u_1).*

DEFINITION 4.8 (triangle complete). *A recognizer is* triangle complete *if it does not have any incomplete triangles.*

Note that if a recognizer is triangle complete, then for any edges $e_1 = (u_1, t^{-1}, u_2)$ and $e_2 = (u_2, t^{-1}, u_3)$, there is an edge $e_3 = (u_3, t^{-1}, u_1)$. Indeed, by the triangle completeness requirement, applied to $\langle e_2^{-1}, e_1^{-1}\rangle$, there is an edge (u_1, t, u_3). But its inverse is the desired (u_3, t^{-1}, u_1).

LEMMA 4.9 (completing one triangle). *Suppose that a deterministic regular recognizer R has an incomplete triangle $\langle(u_1,t,u_2),(u_2,t,u_3)\rangle$. Add a new edge (u_3,t,u_1) and its inverse, and let S be the resulting recognizer. Then*

1. *S is equivalent to R;*
2. *S is deterministic;*
3. *S is regular.*

Proof. Let $e_1=(u_1,t,u_2)$, $e_2=(u_2,t,u_3)$, and $e_3=(u_3,t,u_1)$.

1. Obviously $\Gamma(R)\subseteq\Gamma(S)$. We show that $\Gamma(S)\subseteq\Gamma(R)$. If π is a run of S that accepts a word w, replace every occurrence of e_3 (respectively, e_3^{-1}) in π with $\langle e_2^{-1},e_1^{-1}\rangle$ (respectively, $\langle e_1,e_2\rangle$). The result is a run of R that accepts a word equal to w in G.

2. By contradiction assume that S is not deterministic. Taking into account that R is deterministic and that e_3 and e_3^{-1} are the only new edges of S, we have $\mathrm{Fat}_S(t^{-1},u_1)>0$ or $\mathrm{Fat}_S(t,u_3)>0$. Either case leads to a contradiction. We consider only the first case; the second case is similar.

Suppose that $\mathrm{Fat}_S(t^{-1},u_1)>0$. Then vertex u_1 has an outgoing edge of the form (u_1,t^{-1},u_0) in R. Let $e_0=(u_0,t,u_1)$. Since R is regular, the end-vertices of the path $\langle e_0,e_1,e_2\rangle$ coincide, so that $u_0=u_3$ and $e_0=(u_3,t,u_1)$, which is impossible because $\langle e_1,e_2\rangle$ is an incomplete triangle in R.

3. If S is not regular, then it has an irregular path π. We have proved already that S is deterministic. It follows that π cannot be s-irregular. So π is t-irregular. Let $\pi=\langle f_1,f_2,f_3\rangle$. Since R is regular, π contains the new edge e_3. We have three different scenarios $e_3=f_i$. Each of the scenarios leads to a contradiction.

We consider here only the scenario $e_3=f_2$; the other scenarios are similar. Taking into account that S is deterministic, we have that $f_1=e_2$ and $f_3=e_1$. But then u_2 is the initial and final vertex of π, which contradicts the irregularity of π. □

COROLLARY 4.10 (triangle completion). *There is a polynomial-time* triangle completion algorithm *that transforms an arbitrary deterministic regular recognizer into an equivalent recognizer that is regular, deterministic, and triangle complete.*

Proof. If there is an incomplete triangle $\langle(u_1,t,u_2),(u_2,t,u_3)\rangle$, then add a new edge (u_3,t,u_1) and its inverse. Keep doing that until the recognizer is triangle complete. □

LEMMA 4.11 (membership criterion). *Let R be a triangle complete, deterministic, regular recognizer, and let w be an arbitrary word. The subgroup $\Gamma(R)$ contains the group element w if and only if R accepts the reduct of w.*

Proof. If R accepts the reduct of w, then $w\in\Gamma(R)$ by the definition of recognizers. Suppose that $w\in\Gamma(R)$, and let w_0 be a word with the fewest number of syllables accepted by R. We show that w_0 is reduced.

Let π be a run that accepts w_0. If π has the form

$$\pi_1+\langle(v_1,\sigma,v_2)(v_2,\sigma^{-1},v_3)\rangle+\pi_2,$$

then the middle segment can be removed from π. And if π has the form

$$\pi_1+\langle(v_1,t^{\varepsilon},v_2)(v_2,t^{\varepsilon},v_3)\rangle+\pi_2,$$

then the middle segment can be replaced by the edge $(v_1,t^{-\varepsilon},v_3)$. In either case, the resulting run accepts a word that is equal to w and that has fewer syllables than w_0, which contradicts the choice of w_0. □

LEMMA 4.12 (reading). *There is a polynomial-time* reading algorithm *that, given a deterministic regular recognizer R and a word w, determines whether $w\in\Gamma(R)$.*

Proof. Let R_1 be the given recognizer. Use the triangle completion algorithm to transform R_1 into an equivalent recognizer R_2 that is regular, deterministic, and triangle complete. Use the word reduction algorithm to transform w to an equal reduced word w_0. By the membership criterion lemma, $w \in \Gamma(R_2)$ if and only if R_2 accepts w_0. To determine whether R_2 accepts w_0, run R_2 on w_0. □

LEMMA 4.13 (fat reduction). *There is a polynomial-time* fat reduction algorithm *that transforms any recognizer into an equivalent deterministic regular recognizer.*

Proof. The desired algorithm is iterative. At every round it decreases the number of vertices. We describe one round of the algorithm.

1. If there an irregular path, then identify the end-vertices of the path, and start a new round.
2. If there are distinct edges with the same initial vertex and the same label, then identify their final vertices, and start a new round.
3. Halt.

Obviously the algorithm halts, and when it does the recognizer is deterministic and regular. By the vertex identification and edge folding lemmas in section 2.4, the algorithm does not change the recognizer subgroup. □

THEOREM 4.14. *There is a polynomial-time decision algorithm for the membership problem for* $(\mathbb{Z}/2\mathbb{Z}) * (\mathbb{Z}/3\mathbb{Z})$.

Proof. Use the construction algorithm of section 2.2 to produce a recognizer R_1 for H. Use the weight reduction algorithm to transform R_1 to an equivalent recognizer that is regular and deterministic. Finally use the reading algorithm to check whether $w \in \Gamma(S)$. □

5. Succinct $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$. In the previous section, we proved that the membership problem for standard $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$ is polynomial-time decidable. Here we introduce an exponentially more succinct syllabic representation of the elements of $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$ and show that the membership problem for G remains polynomial-time decidable.

5.1. Succinct $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$: Definition and word reduction.

DEFINITION 5.1 (succinct $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$). Succinct $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$ *is the group* $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$ *in the following syllabic representation.*

- *Alphabet: The alphabet consists of original letters s and t and the following auxiliary symbols: $^{-1}$, left and right parentheses, and ^{0}, ^{1}, ..., ^{9}. The alphabet is linearly ordered in some way; it will play no role in what order it is exactly.*
- *Syllables: The syllables split into three categories.*
  –Positive: strings $(ts)^k t$, where k is a natural number in decimal notation,
  –Negative: strings $(t^{-1}s)^k t^{-1}$, where k is a natural number in decimal notation,
  –Neutral: the string s.

  Positive and negative syllables are partisan. *The Greek letter ε is reserved to range over $\{1, -1\}$, so that t^ε is always either t or t^{-1}. To distinguish this representation of $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$ from the standard representation, new words will be called* exponent words, *and the old words will be called* unary words. *Any exponent word w expands in the obvious way to a unary word called the* unary expansion *of w. For example, $(t,s)^4 s(ts)^4$ expands to $tststststststststst$.*
- *Inverses: Syllables $(ts)^k t$ and $(t^{-1}s)^k t^{-1}$ are the inverses of each other, and s is its own inverse.*

- *Equality: Exponent words are* equal *if their unary expansions are equal in the sense of the standard $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$.*

Obviously the quotient of the word semigroup over the equality relation is isomorphic to $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$.

As usual, the sign of an integer i is $1, 0$, or -1 if $i > 0$, $i = 0$, or $i < 0$, respectively, and $|i|$ is the absolute value of i.

Definition 5.2 (T notation).

$$T_i = \begin{cases} (ts)^{i-1}t & \textit{if } i > 0, \\ (t^{-1}s)^{|i|-1}t^{-1} & \textit{if } i < 0, \\ 1 & \textit{if } i = 0. \end{cases}$$

Lemma 5.3 (syllables). *Let i, j be nonzero integers.*

1. $T_i s T_j = T_{i+j}$ *if* $\mathrm{sign}(i) = \mathrm{sign}(j)$.
2. $$T_iT_j = \begin{cases} T_{i-1}s^\alpha T_{-1}s^\beta T_{j-1} & \textit{if } \mathrm{sign}(i) = \mathrm{sign}(j) = 1, \\ T_{i+1}s^\alpha T_1 s^\beta T_{j+1} & \textit{if } \mathrm{sign}(i) = \mathrm{sign}(j) = -1, \end{cases}$$
$$\textit{where } \alpha = \begin{cases} 1 & \textit{if } \ |i| \neq 1, \\ 0 & \textit{if } \ |i| = 1, \end{cases} \quad \textit{and} \quad \beta = \begin{cases} 1 & \textit{if } \ |j| \neq 1, \\ 0 & \textit{if } \ |j| = 1. \end{cases}$$
3. $$T_iT_j = \begin{cases} sT_{i+j} & \textit{if } \mathrm{sign}(i) = -\mathrm{sign}(j) \textit{ and } |i| < |j|, \\ 1 & \textit{if } \mathrm{sign}(i) = -\mathrm{sign}(j) \textit{ and } |i| = |j|, \\ T_{i+j}s & \textit{if } \mathrm{sign}(i) = -\mathrm{sign}(j) \textit{ and } |i| > |j|. \end{cases}$$

The proof is straightforward. We give here only a few examples.

$$\begin{aligned}
&T_2sT_3 = (tst)s(tstst) = (ts)^4t = T_5,\\
&T_{-2}sT_{-3} = (t^{-1}st^{-1})s(t^{-1}st^{-1}st^{-1}) = (t^{-1}s)^4t^{-1} = T_{-5},\\
&T_2T_3 = (tst)(tstst) = (t)s(tt)s(tst) = T_1sT_{-1}sT_2,\\
&T_1T_3 = (t)(tstst) = (tt)s(tst) = T_{-1}sT_2,\\
&T_2T_1 = (tst)(t) = (t)s(tt) = T_1sT_{-1},\\
&T_{-2}T_{-2} = (t^{-1}st^{-1})(t^{-1}st^{-1}) = (t^{-1})s(t^{-1}t^{-1})s(t^{-1}) = T_{-1}sT_1sT_{-1},\\
&T_2T_{-3} = (tst)(t^{-1}st^{-1}st^{-1}) = st^{-1} = sT_{-1},\\
&T_3T_{-2} = (tstst)(t^{-1}st^{-1}) = ts = T_1s,\\
&T_2t_{-2} = (tst)(t^{-1}st^{-1}) = 1.
\end{aligned}$$

Definition 5.4 (reduced exponent words). *An exponent word is* reduced *if*

(R1) *every nonfinal neutral syllable is followed by a partisan syllable,*
(R2) *every nonfinal partisan syllable is followed by a neutral syllable,*
(R3) *any two partisan syllables separated only by a neutral syllable have different signs.*

For example, the exponent word $T_3sT_{-7}sT_2$ is reduced.

Lemma 5.5 (exponent word reduction).

1. *Every exponent word w is equal in G to a unique reduced exponent word. The reduced exponent word is the* reduct *of w.*

2. *There is a polynomial-time* exponent word reduction algorithm *that, given any exponent word w, computes the reduct of w.*

Proof. It is easy to see that (i) if an exponent word is reduced, then its unary expansion is reduced in the sense of section 4 and (ii) if two reduced exponent words are equal, then the reduced unary expansions are equal. By the word reduction lemma of section 4, equal reduced unary words are identical. It follows that equal reduced exponent words have identical unary expansions. It is easy to see that equal reduced exponent words are also identical as exponent words; that is, they have the same syllables in the same order. This establishes the uniqueness part of the first claim.

In the remainder of the proof, we construct polynomial-time algorithms A_1, A_2, and A_3 transforming any exponent word w to an equal exponent word in such a way that $A_1(w)$ satisfies requirement (R1), $A_2(A_1(w))$ satisfies requirements (R1) and (R2), and $A_3(A_2(A_1(w)))$ satisfies requirements (R1), (R2), and (R3) and thus is reduced.

Taking into account that $s^2 = 1$, A_1 is obvious. Taking into account part 1 of the syllables lemma, A_3 is obvious. It remains to construct A_2.

We say that two successive syllables of an exponent word w *collide* if both syllables are partisan. Define $\mathrm{rank}(w) = (c, b)$, where c is the number of collisions in w and b is the number of partisan syllables in w. Order ranks lexicographically. A_2 is an iterative rank-reducing algorithm. It is presumed that the input exponent word satisfies A_1. In the next paragraph, we describe one round of A_2.

If w satisfies (R2) then halt. Otherwise w has the form xT_mT_ny. If $\mathrm{sign}(m) = \mathrm{sign}(n)$, use part 2 of the syllables lemma to reduce the number of collisions. If $\mathrm{sign}(m) = -\mathrm{sign}(n)$, use part 3 of the syllables lemma to reduce the rank. If (R1) is violated in the process, then apply A_1. ∎

5.2. Deficit reduction.

Recognizers were introduced in section 2.2.

DEFINITION 5.6 (edges). *An edge of a recognizer is* partisan *or* neutral *if its label is so. A partisan edge is* positive *if or* negative *if its label is so. The* length *of an edge e with label σ is the number of symbols in the unary expansion of σ. Thus* $\mathrm{Length}(e) = 2|i| - 1$ *if $\sigma = T_i$, and* $\mathrm{Length}(e) = 1$ *if $\sigma = s$.*

DEFINITION 5.7 (paths). *The* length *of a path π is the sum of the lengths of its edges. A path π is* positive *if*

- *it has at least one positive edge and no negative edges, and*
- *the positive and neutral edges of π alternate.*

Negative paths are defined similarly. Positive and negative paths are partisan.

DEFINITION 5.8 (fat). *Let R be a recognizer and u range over the vertices of R.*

- $\mathrm{Fat}_R(s, u) = \max(0, n-1)$, *where n is the number of neutral edges from u.*
- $\mathrm{Fat}_R(t, u) = \max(0, n-1)$, *where n is the number of positive edges from u.*
- $\mathrm{Fat}_R(t^{-1}, u) = \max(0, n-1)$, *where n is the number of negative edges from u.*

Further, $\mathrm{Fat}_R(u) = \mathrm{Fat}_R(s, u) + \mathrm{Fat}_R(t, u) + \mathrm{Fat}_R(t^{-1}, u)$, *and* $\mathrm{Fat}(R) = \sum_u \mathrm{Fat}_R(u)$. *The subscript may be omitted if the context uniquely defines the recognizer.*

DEFINITION 5.9 (lean recognizers). *It is* lean *if* $\mathrm{Fat}(R) = 0$.

As in section 3.3, lean recognizers are deterministic, but deterministic recognizers are not necessarily lean. Recall the regular recognizers and the triangles definitions in section 4.

DEFINITION 5.10 (deficit). *Let R be a recognizer. A vertex u_2 of R is* deficient *if there are positive edges (u_1, T_m, u_2) and (u_2, T_n, u_3) such that either $m + n > 2$ or* *else* $m = n = 1$ *and* $\langle(u_1, T_m, u_2), (u_2, T_n, u_3)\rangle$ *is an incomplete triangle. The number of deficient vertices is the* deficit $\Delta(R)$ *of* R.

Lemma 5.11 (deficit decrement). *There is a polynomial-time* deficit decrement algorithm *that transforms any lean regular recognizer* R *with positive deficit to an equivalent lean regular recognizer* S *with lesser deficit.*

Proof. Find a deficient vertex u_2 together with edges $e_1 = (u_1, T_m, u_2)$ and $e_2 = (u_2, T_n, u_3)$ witnessing the deficiency of u_2. Without loss of generality $m \geq n$; the case $n \geq m$ is similar. We consider various scenarios that arise and advise the reader to draw diagrams.

Case 1: $m = n = 1$.

Case 1-00: u_1 has no incoming positive edges, and u_3 has no outgoing positive edges. Add edges (u_3, t, u_1) and (u_1, t^{-1}, u_3). By the vertex creation lemma in section 3.2, the resulting recognizer S is equivalent to R. Obviously S is regular and lean. $\Delta(S) = \Delta(R) - 1$ as one deficiency has been repaired without creating any other deficiencies.

Case 1-11: u_1 has an incoming positive edge $e_0 = (u_0, T_\ell, u_1)$, and u_3 has an outgoing positive edge $e_3 = (u_3, T_p, u_4)$. Vertices u_1, u_2, and u_3 are all deficient. Since R is regular, we have $\ell, p > 1$. Split e_0 and e_3 into paths of the form

$$\langle(u_0, T_{\ell-1}, u), (u, s, u'), (u', t, u_1)\rangle,\ \langle(u_3, t, v), (v, s, v'), (v', T_{p-1}, u_4)\rangle.$$

The resulting recognizer R' is lean and equivalent to R. $\Delta(R') = \Delta(R)$. Note t-irregular paths from u' to u_3 and from u_1 to v. Identify u' with u_3 and u_1 with v, so that the edges (u', t, u_1) and (u_3, t, v) become one edge (u_3, t, u_1), and the edges (u_1, t^{-1}, u') and (v, t^{-1}, u_3) become one edge (u_1, t^{-1}, u_3). The resulting recognizer S is regular, lean, and equivalent to R'. $\Delta(S) = \Delta(R) - 3$.

Case 1-01: u_1 has no incoming positive edges, and u_3 has an outgoing positive edge $e_3 = (u_3, T_p, u_4)$. Since R is regular, we have $p > 1$. Vertices u_2 and u_3 are deficient. Split e_3 as in case 1-11. The resulting recognizer R' is lean and equivalent to R. $\Delta(R') = \Delta(R)$. Note a t-irregular path from u_1 to v. Identify u_1 with v, so that the edge (u_3, t, v) becomes (u_3, t, u_1) and the edge (v, t^{-1}, u_3) becomes (u_1, t^{-1}, u_3). The resulting recognizer S is regular, lean, and equivalent to R'. $\Delta(S) = \Delta(R) - 2$.

Case 1-10: u_1 has an incoming positive edge $e_0 = (u_0, T_\ell, u_1)$, and u_3 has no outgoing positive edge. This case is dual (and similar) to case 1-01.

Case 2: $m, n > 1$. Split e_1 and e_2 into paths of the form

$$\langle(u_1, T_{m-1}, u), (u, s, u'), (u', t, u_2)\rangle,\ \langle(u_2, t, v), (v, s, v'), (v', T_{n-1}, u_3)\rangle.$$

The resulting recognizer R' is regular, lean, and equivalent to R. $\Delta(R') = \Delta(R)$. Continue as in case 1-00, with t-edges (u', t, u_2) and (u_2, t, v) witnessing the deficiency of u_2.

Case 3: $m > 1$ and $n = 1$.

Case 3-0: u_3 has no outgoing positive edges. Split e_1 as in case 2. The resulting recognizer R' is lean and equivalent to R. $\Delta(R') = \Delta(R)$. Continue as in case 1-00 with edges (u', t, u_2) and (u_2, t, u_3) witnessing the deficiency of u_2.

Case 3-1: u_3 has an outgoing positive edge $e_3 = (u_3, T_p, u_4)$.

Case 3-11: $p = 1$. If u_4 has no outgoing positive edges, then we have case 1-10 with the current u_2, u_3, and u_4 playing the role of u_1, u_2, and u_3 of case 1-10. If u_4 has an outgoing positive edge, then we have case 1-11 with the current u_2, u_3, and u_4 playing the role of u_1, u_2, and u_3 of case 1-11.

Case 3-12: $p > 1$. Vertices u_2 and u_3 are deficient. Split e_1 as in case 2, and split e_3 as in case 1-11. The resulting recognizer R' is lean and equivalent to R. $\Delta(R') = \Delta(R)$. Note a t-irregular path from u' to v. Identify u' with v. The resulting recognizer S is regular, lean, and equivalent to R'. $\Delta(S) = \Delta(R') - 2$. □

COROLLARY 5.12 (deficit reduction). *There is a polynomial-time* deficit reduction algorithm *that transforms any lean regular recognizer into an equivalent zero-deficit lean regular recognizer*

5.3. Membership criterion and the reading algorithm. The quasi-runs definition, the quasi-run labels corollary, and the tolerance definition of section 3.2 remain valid. By Part 1 of the syllables lemma in section 5.1, every partisan path that starts and ends with partisan edges is a quasi transition.

DEFINITION 5.13 (standard quasi transitions). *A quasi-transition q is* partisan *if it is a partisan path that starts and ends with partisan edges. A partisan quasi transition is* positive *(respectively,* negative*) if it is so as a path. A* neutral quasi transition *consists of one neutral edge. A quasi transition is* standard *if it is partisan or neutral.*

The label of every standard quasi transition is a syllable (rather than 1).

DEFINITION 5.14 (standard quasi runs). *A quasi-run $\langle q_1, \dots, q_\ell\rangle$ is* standard *if every quasi-transition q_i is standard.*

LEMMA 5.15 (membership criterion). *Suppose that R is a zero-deficit lean regular recognizer, and let w be a reduced exponent word. The following are equivalent.*

1. *There is a standard quasi run with label w.*
2. *The group element w belongs to $\Gamma(R)$.*

Proof. $2 \to 1$: By the quasi-run labels corollary, the associate run of Q accepts a word equal to w, and so $w \in \Gamma(R)$.

$1 \to 2$: Suppose that the group element w belongs to $\Gamma(R)$. Without loss of generality, $w \neq 1$. By the recognizer's subgroup definition in section 2.2, R accepts an exponent word w' equal to w. Let π be a run that accepts w'.

We claim that the partisan and neutral edges alternate in π. To prove that, we assume that π has the form $\pi_1 + e + f + \pi_2$, where the edges e and f are both partisan or both neutral, and we prove that there is a shorter run accepting a word equal to w. Let v be the final vertex of e. If e and f are neutral, then by the regularity of R, $f = e^{-1}$, and so $\pi_1 + \pi_2$ is a shorter run accepting a word equal to w. So e and f are partisan. Let $T_m = \text{Label}(e)$ and $T_n = \text{Label}(f)$. Without loss of generality, $m > 0$; the case $m < 0$ is similar. If $n < 0$, then $f = e^{-1}$ because $\text{Fat}(t^{-1}, v) = 0$. Again, $\pi_1 + \pi_2$ is a shorter run accepting a word equal to w. Thus $n > 0$. Since $\Delta(R) = 0$, the vertex v is not deficient. It follows that $m = n = 1$, and there is an edge g such that $\langle e, f, g\rangle$ is a triangle. Then $\pi_1 + g^{-1} + \pi_2$ is a shorter run accepting an exponent word equal to w.

Notice that π is a standard quasi run. Let $Q = \langle q_1, \dots, q_\ell\rangle$ be a standard quasi run with the fewest number of quasi transitions that tolerates an exponent word w_0 equal to w. Accordingly w_0 has the form $a_1^{p_1} \dots a_\ell^{p_\ell}$. We show that w_0 is reduced. Recall the definition of reduced exponent words in section 5.1. Obviously w_0 satisfies the conditions (R1) and (R2). It remains to prove that it satisfies the condition (R3). It suffices to prove that the positive and negative quasi transitions alternate in Q.

By contradiction suppose that two partisan syllables σ_i and σ_{i+2} of the same sign are separated by a neutral syllable σ_{i+1}. Using the syllables lemma in section 5.1, replace the segment $\langle q_i, q_{i+1}, q_{i+2}\rangle$ with a single quasi transition whose quasi label equals $\sigma_i\sigma_{i+1}\sigma_{i+2}$ in G. This gives a quasi-run Q' with fewer quasi transitions that accepts a word equal to w, which contradicts the choice of Q. □

LEMMA 5.16 (quasi transitions).

1. *Let R be a zero-deficit lean regular recognizer and u be a vertex of R. For every syllable σ, there is at most one standard quasi-transition q from u with quasi-label σ.*
2. *There is a polynomial-time algorithm that, given a zero-deficit lean regular recognizer R, a state u of R, and a syllable σ, determines whether there exists a standard quasi transition with profile of the form (u, σ, v) and, if yes, constructs the desired quasi transition.*

Proof. 1. The case $\sigma = s$ is obvious because R is regular. By contradiction assume that there exist distinct standard quasi transitions with the same quasi-label T_n from the same vertex u. We assume that $n > 0$; the case $n < 0$ is similar. Thus we have two distinct positive paths of the same length $2n - 1$ from u. Since neither path can be a prefix of the other, there is a vertex v where the two paths branch out which contradicts the leanness of R.

2. The case $\sigma = s$ is obvious. Thus we may assume that $\sigma = T_n$ for some $n \neq 0$. We assume $n > 0$; the case $n < 0$ is similar. Using the leanness of R, construct a unique empty or positive path π such that $\text{Length}(\pi) < 2n - 1$ but π cannot be extended to a longer positive path or such that $\text{Length}(\pi) \geq 2n - 1$ and π is the shortest such path. In the first case, the desired quasi transition does not exist. Consider the second case. If $\text{Length}(\pi) > 2n - 1$, then the desired quasi transition does not exists. But if $\text{Length}(\pi) = 2n - 1$, then we have the desired quasi transition. Notice that π may cycle from some point on. □

LEMMA 5.17 (reading). *There is a polynomial-time* reading algorithm *that, given a lean regular recognizer R and a reduced exponent word w, determines whether the group element w belongs to $\Gamma(R)$.*

Proof. Taking into account the deficit corollary in section 5.2, we may assume without loss of generality that R is zero-deficit.

The reduced exponent word w is a concatenation $\sigma_1 \dots \sigma_\ell$ of syllables where the partisan syllables alternate with the neutral syllables and where, among the partisan syllables, positive syllables alternate with negative syllables. By the membership criterion lemma, it suffices to determine whether there exists a standard quasi-run $Q = \langle q_1, \dots, q_\ell \rangle$ with label w. By the quasi-transition lemma, there is at most one such quasi run.

Intuitively speaking, we use R to "read" w. Let $u_0 = o$. Suppose that $j \leq \ell$ and S has read the initial segment $\sigma_1 \dots \sigma_j$ of w and arrived at state u_j. In the process an initial segment $q_1 \dots q_j$ of the desired Q has been constructed.

If $j < \ell$, then apply the algorithm of the quasi-transition lemma to (R, u_j, σ_{j+1}). If the algorithm determines that there is no appropriate q_{j+1}, then $w \notin \Gamma(R)$. Otherwise the algorithm produces the appropriate q_{i+1}. Set u_{j+1} to be the final vertex of q_{j+1}, and proceed to read σ_{j+2}.

If $j = \ell$, then $w \in \Gamma(R)$ if and only if $u_\ell = o$. □

5.4. Path folding. This section is similar to section 3.4, but we need to do a little work to make the similarity apparent. The role of a fixed original letter a of section 3.4 is played by the word ts in this section. The fact that the word ts is not a single letter creates some difficulties.

Consider a regular recognizer R. Since R is regular, every vertex of R has at most one outgoing s-edge.

Definition 5.18 (dual vertices). *If a vertex u has an outgoing s-edge e, then the end of e is the* dual *of the vertex u; otherwise, u is an s-*orphan.

It will be convenient to pretend that orphans have dual vertices as well.

Definition 5.19 (virtual duals). *With every orphan u, we associate an object v outside of R in such a way that distinct objects are associated with distinct vertices. These objects v are called* virtual elements *of R. The nature of virtual elements is of no importance. If an orphan u is associated with a virtual element v, we say that u and v are the* duals *of each other. Further, the triples (u, s, v) and (v, s, u) are called* virtual edges *of R.*

Recall that a partisan path has the following properties: partisan and neutral edges alternate and all partisan edges have the same sign.

Definition 5.20 (regular paths). *A* regular path *π is a nonempty partisan path of the form $\langle e_1, f_1, \ldots, e_k, f_k\rangle$ subject to the following restrictions.*

- *Edges e_i are partisan.*
- *Edges f_i are neutral, and the final edge f_k may be virtual.*
- *The initial vertices of edges e_i are all distinct.*

If f_k is virtual, then π is odd, *and if f_k is real, then π is* even. *Notice that the sequence $\langle e_1, \ldots, e_k\rangle$ completely determines the regular path π which will be denoted* $\mathrm{RP}(e_1, \ldots, e_k)$.

Definition 5.21 (even vertices). *Let π be a regular path $\langle e_1, f_1, \ldots, e_k, f_k\rangle$, where $e_i = (u_{i-1}, \sigma_i, v_i)$ and $f_i = (v_i, s, u_i)$. Notice that every* $\mathrm{Length}(\pi[u_0, u_i])$ *is even and every* $\mathrm{Length}(\pi[u_0, v_i])$ *is odd. Call vertices u_i* even *and vertices v_i* odd. *So the even vertices are the initial vertices of the partisan edges and the final vertices of the neutral edges. The* even vertex sequence *of π is the sequence $\langle u_0, \ldots, u_{k-1}, u_k\rangle$. By the definition of regular paths, even vertices $u_0, \ldots, u_{k-1}$ are distinct.*

In the following definition, we redefine the notion of branch and cycle and some related notions to the case of regular paths. We will not apply the old versions of these notions to regular paths. In fact, the old notions will not be used in the rest of this paper, except that here and there we refer the reader to previous sections where the old notions are in use.

Definition 5.22 (branches, cycles, etc.). *Consider a regular path $\pi = \langle e_1, \ldots, e_k\rangle$ with even vertex sequence $\langle u_0, u_1, \ldots, u_k\rangle$.*

- *π is a* branch *if $u_k \notin \{u_0, \ldots, u_{k-1}\}$.*
- *π is an* impasse *if it is a branch and u_k does not have an outgoing partisan edge of the sign of π.*
- *π is a* cycle at u_0 *if $u_k = u_0$.*
- *π is a* noose *if $u_k = u_i$ for some positive $i < k$.*
- *If π is a noose and $u_k = u_i$, where $i < k$, then π splits into the* loop $\mathrm{RP}(e_{i+1}, \ldots, e_k)$ *and the* tail *$\langle e_1, \ldots, e_i\rangle$ of the noose.*
- *π is* closed *if it is an impasse, a cycle, or a noose.*

Lemma 5.23 (closed paths). *Every partisan edge e gives rise to a closed regular path $\pi = \mathrm{RP}(e_1, \ldots, e_k)$ with $e_1 = e$. Furthermore, there is a polynomial-time algorithm that, given (a regular recognizer and) a partisan edge e, constructs such a path π.*

Proof. The desired algorithm is iterative. At the first round, it constructs the regular path $\mathrm{RP}(e_1)$ with $e_1 = e$. Suppose that we did i rounds and constructed a regular path $\mathrm{RP}(e_1, \ldots e_i)$. If it is closed, then halt. Otherwise $\mathrm{RP}(e_1, \ldots e_i)$ is even. Let e_{i+1} be the lexicographically first partisan edge from u_i of the sign of e, and construct $\mathrm{RP}(e_1, \ldots e_{i+1})$. The process converges because the number of vertices is finite. □

Definition 5.24 (edge splitting). *Suppose that p, q, and r are positive integers such that $r = p + q$. By the syllables lemma in section* 5.1, $T_r = T_p s T_q$, *and* $T_{-r} = T_{-p} s T_{-q}$. *To* split an edge $e = (u, T_r, v)$ according to (p, q), *create two new vertices u' and v' and replace edges e and e^{-1} with six new edges: (u, T_p, u'), (u', s, v'), and (v', T_q, v) and their inverses. Splitting an edge $e = (u, T_{-r}, v)$ according to (p, q) is defined similarly; just substitute T_p, T_q, and T_r with T_{-p}, T_{-q}, and T_{-r}, respectively.*

Lemma 5.25 (edge splitting). *Edge splitting preserves the regularity of the recognizer, the recognizer subgroup, and the amount of fat of the recognizer.*

Proof. The proof is obvious. □

Definition 5.26 (vertex creation). *Let $\pi = \mathrm{RP}(e_1, \ldots, e_k)$, ν be the initial vertex of π, and m be a positive even number such that $m <$ Length(π) and there is no even vertex of π with* Length$(\pi[\nu, u]) = m$. *We explain how to create, on π, a new even vertex v' at distance m from ν as well as its dual u' at distance m from ν. Let $L(i) =$ Length$(\mathrm{RP}\langle e_1, \ldots, e_i\rangle)$ for $i = 0, \ldots, k$. Find the index i with $L(i-1) < m < L(i)$, and split the edge e_i according to (p, q), where $p = (m-L(i-1))/2$ and $q = (L(i) - m)/2$. This creates, on π, the desired even vertex v' as well as its dual u'.*

Corollary 5.27 (vertex creation). *Vertex creation preserves the regularity of the recognizer, the recognizer subgroup, and the amount of fat of the recognizer.*

We adjust the entanglement definition of section 3.4 to fit our needs in this section. The two path divisor definition of section 3.4 remains valid.

Definition 5.28 (entanglement). *Suppose that π and ρ are regular paths of the same sign and with the same initial vertex ν. Suppose further that either path is a branch or a cycle. The two paths are* entangled *if there exist even vertices u and v on π and ρ, respectively such that $u \neq v$ and*

$$\mathrm{Length}(\pi[\nu, u]) = \mathrm{Length}(\rho[\nu, v]) \mod \mathrm{Div}(\pi, \rho).$$

Otherwise the two paths are disentangled.

The entanglement algorithm corollary of section 3.4 remains valid.

Lemma 5.29 (entanglement). *If π and ρ are entangled and even vertices u and v witness the entanglement, then the identification of u and v does not change the recognizer subgroup.*

Proof. Let $d = \mathrm{Div}(\pi, \rho)$, $2k = \mathrm{Length}(\pi[\nu, u])$, $2\ell = \mathrm{Length}(\rho[\nu, v])$, and $H = \mathrm{Coset}(\nu)$. We assume that π are ρ are positive; the negative case is similar. By the syllables lemma in section 5.1, $\mathrm{Label}(\pi[\nu, u]) = (ts)^k$, and $\mathrm{Label}(\rho[\nu, v]) = (ts)^\ell$. By the coset stability lemma in section 2.4, $\mathrm{Coset}(u) = H(ts)^k$ and $\mathrm{Coset}(v) = H(ts)^\ell$. By the vertex identification lemma in section 2.4, it suffices to prove that $H(ts)^k = H(ts)^\ell$. Since u and v witness the entanglement, we have $2k = 2\ell \mod d$.

Case 1: Both paths are branches. Then $k = \ell$, and therefore $H(ts)^k = H(ts)^\ell$.

Case 2: One of the paths is a branch and the other is a cycle. Without loss of generality, π is a cycle, and ρ is a branch. Then $d = 2k$, $H(ts)^k = H$, and $2\ell = p \cdot 2k$ for some integer p. Then $\ell = kp$, and $H(ts)^\ell = H((ts)^k)^p = H = H(ts)^k$.

Case 3: Both paths are cycles. Then $d = \mathrm{g.c.d.}(2k, 2\ell)$, and $H(ts)^k = H(ts)^\ell = H$. Clearly d is even; let $\delta = d/2$ so that $\delta = \mathrm{g.c.d.}(k, \ell)$. Since $\delta = \mathrm{g.c.d.}(k, \ell)$, there are integers i and j such that $ik + j\ell = \delta$, and so $H(ts)^\delta = H((ts)^k)^i((ts)^\ell)^j = H$. Since $2k = 2\ell \mod d$, there is an integer p such that $2k = pd + 2\ell$, and therefore $k = p\delta + \ell$. Then $H(ts)^k = H((ts)^\delta)^p(ts)^\ell = H(ts)^\ell$. □

Definition 5.30 (even vertex disjoint regular paths). *Consider two regular paths sharing the same initial vertex ν. The two paths are* even vertex disjoint off ν *if ν is the only even vertex on both paths.*

If two regular paths are even vertex-disjoint off their common initial vertex ν, then the dual of ν is the only possible odd vertex on both paths. But it is possible that an even vertex of one path appears as an odd vertex on the other path.

Definition 5.31 (path folding). *Suppose that π and ρ are regular paths such that*

- *they have the same sign and the same initial vertex ν,*
- *either path is a branch or a cycle,*
- $\mathrm{Length}(\pi) \geq \mathrm{Length}(\rho)$ *if both paths are branches, and*
- $\mathrm{Length}(\pi) = \mathrm{Div}(\pi, \rho)$ *if both paths are cycles.*

To fold ρ into π, *execute the following* folding algorithm:

1. *Apply the entanglement algorithm to π and ρ. If the number of vertices is reduced, then halt. Otherwise the paths are disentangled.*
2. *For each even vertex v on ρ, create, on π, a new even vertex v' and its dual such that*
$$\mathrm{Length}(\pi[\nu, v']) = \mathrm{Length}(\rho[\nu, v]) \mod \mathrm{Div}(\pi, \rho)$$
unless π has an even vertex at this position already.
3. *Identify every clone v' with its original v, and identify the dual of v' with the dual of v.*
4. *Remove all partisan edges of ρ and their inverses.*

The path folding lemma of section 3.4 remains valid. Its formulation does not change at all, and its proof requires only small adjustments. For the reader's convenience we give here all details.

Lemma 5.32 (path folding). *Let π and ρ be as in the path folding definition. Folding ρ into π preserves the recognizer subgroup. If the algorithm halts at stage* 1, *then the number of vertices of the recognizer decreases. Otherwise the number of vertices is unchanged, and the (amount of) fat changes as follows.*

(BB) *Suppose that π and ρ are branches of lengths m and n, respectively.*
If $m = n$, then the fat decreases by 2.
If $m > n$ and ρ is an impasse, then the fat decreases by 1.
If $m > n$ but ρ is not an impasse, then the fat does not change.

(CB) *Suppose that π is a cycle and ρ is a branch.*
If ρ is an impasse, then the fat decreases by 1*;*
otherwise, the fat does not change.

(CC) *Suppose that π and ρ are cycles. Then the fat decreases by* 2.

Proof. We consider only the case in which π and ρ are positive; the case in which they are negative is similar.

Let R be the given recognizer, and let R_p be the recognizer obtained from R by executing p stages of the folding algorithm, so that $R_0 = R$. Let the even vertex sequence of π be $\langle u_0, \ldots, u_k \rangle$. Let $\rho = \mathrm{RP}(f_1, \ldots, f_\ell)$ and the even vertex sequence of ρ be $\langle v_0, \ldots, v_\ell \rangle$. Let $d = \mathrm{Div}(\pi, \rho)/2$. The case in which π and ρ are entangled is obvious. Assume that π and ρ are disentangled. Then nothing happens at stage 1, and so $R_1 = R_0$.

First we note that the vertices of R_4 are those of R_1. Indeed all vertices v_j' and their duals created at stage 2 are identified with the respective vertices v_j and their duals at stage 3; thus, the vertices of R_3 are those of R_1. And the vertices do not change at stage 4.

Second we show that $\Gamma(R_4) = \Gamma(R)$. By the vertex creation lemma in section 2.4, $\Gamma(R_2) = \Gamma(R_1)$. By the vertex identification lemma in section 2.4, $\Gamma(R_3) = \Gamma(R_2)$. It remains to show that $\Gamma(R_4) = \Gamma(R_3)$. Let $n_j = \frac{1}{2}\mathrm{Length}(\pi[v_0, v_j])$, $r_j = n_j \mod d$, and $v'_0 = u_0$. It suffices to show that, for every partisan edge f_j of ρ, R_4 has a path P with the profile of f_j. The profile of f_j is (v_{j-1}, T_p, v_j), where $p = n_j - n_{j-1}$. In scenario (BB), the desired path P is $\pi[v'_{j-1}, \mathrm{Dual}(v'_j)]$. In scenarios (CB) and (CC), $p = d \cdot q + (r_{j+1} - r_j)$ for some q. The desired path P starts at v'_{j-1} and ends at $\mathrm{Dual}(v'_j)$. If $r_i \leq r_{i+1}$, then π does q full revolutions around π. If $r_i > r_{i+1}$, then $q > 0$, and π does $q - 1$ full revolutions around π.

Finally we prove the claims about the fat. By the vertex creation lemma, $\mathrm{Fat}(R_2) = \mathrm{Fat}(R_1)$. Thus we need only to examine the evolution of the fat from R_2 to R_4. Furthermore, it suffices to examine the evolution of the numbers $\mathrm{Fat}(t, v_j)$ and $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_j))$. Indeed, v'_j and its dual merge with v_j and its dual, respectively, at stage 3. $\mathrm{Fat}(t^{-1}, v_j)$ and $\mathrm{Fat}(t, \mathrm{Dual}(v_j))$ do not change. And if a vertex v of R_2 differs from any v_j, from any v'_j, and from their duals, then the immediate vicinity of v does not change.

If $0 < j < \ell$, then $\mathrm{Fat}(t, v_j)$ and $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_j))$ do not change from R_2 to R_4. Indeed, as a result of identification with v'_j, at stage 3, the vertex v_j acquires one outgoing positive edge, and $\mathrm{Dual}(v_j)$ acquires one outgoing negative edge, but then, at stage 4, v_j loses one outgoing positive edge, namely, f_{j+1}, and $\mathrm{Dual}(v_j)$ loses one outgoing negative edge, namely, f_j^{-1}.

The vertices v_0 and its dual do not acquire any outgoing edges at stage 3, and thus neither $\mathrm{Fat}(t, v_0)$ nor $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_0))$ increase on stage 3. v_0 loses one positive outgoing edge, namely, f_0, at stage 4, and so $\mathrm{Fat}(t, v_0)$ decreases by 1 from R_2 to R_4. $\mathrm{Dual}(v_0)$ does not lose any negative outgoing edge in scenarios (BB) or (CB), and so $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_0))$ does not change in scenarios (BB) and (CB). Since $v_0 = v_\ell$ in scenario (CC), it remains only to examine the evolution of the numbers $\mathrm{Fat}(t, v_\ell)$ and $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_\ell))$ in the three scenarios.

(BB) In this scenario, we first suppose that $m = n$. Since π and ρ are disentangled, $u_k = v_\ell$. The vertices v_ℓ and its dual do not acquire any outgoing edges at stage 3 and lose only one outgoing edge, namely, the negative edge f_l^{-1}, at stage 4. Thus $\mathrm{Fat}(t, v_\ell)$ does not change, and $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_\ell))$ decreases by 1. To summarize, $\mathrm{Fat}(t, v_0)$ and $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_\ell))$ decrease by 1 while $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_0))$ and $\mathrm{Fat}(t, v_\ell)$ do not change. Hence $\mathrm{Fat}(R_4) = \mathrm{Fat}(R_2) - 2$.

Second we suppose that $m > n$. At stage 3, v_ℓ acquires one positive outgoing edge, and $\mathrm{Dual}(v_\ell)$ acquires one negative outgoing edge. At stage 4, v_ℓ loses no outgoing edges while $\mathrm{Dual}(v_\ell)$ loses f_ℓ^{-1}. Thus $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_\ell))$ do not change. If ρ is not an impasse, then $\mathrm{Fat}(t, v_\ell)$ increases by 1; otherwise, $\mathrm{Fat}(t, v_\ell)$ remains zero throughout the process. We summarize. If ρ is an impasse, then $\mathrm{Fat}(t, v_0)$ decreases by 1 while $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_0))$, $\mathrm{Fat}(t, v_\ell)$, and $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_\ell))$ do not change, so that $\mathrm{Fat}(R_4) = \mathrm{Fat}(R_2) - 1$. If ρ is not an impasse, then $\mathrm{Fat}(t, v_0)$ decreases by 1, $\mathrm{Fat}(t, v_\ell)$ increases by 1, and $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_0))$ and $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_\ell))$ do not change, so that $\mathrm{Fat}(R_4) = \mathrm{Fat}(R_2)$.

(CB) This scenario is similar to the case $m > n$ of scenario (BB). Let us just point out that the vertex v_ℓ does not occur on π in R. Indeed suppose the opposite. Since π and ρ are even vertex disjoint off their common initial vertex and $u_k = u_0$, we have $v_\ell = u_0 = v_0$. But then ρ is a cycle which contradicts scenario (CB).

(CC) In this scenario, $v_0 = v_\ell$, and thus $\mathrm{Fat}(t, v_\ell)$ decreases by 1. $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_\ell))$ does not change at stage 3 and decreases by 1 at stage 4 because f_ℓ^{-1} is removed. To summarize, the overall change in the fat is this: both $\mathrm{Fat}(t, v_\ell)$ and $\mathrm{Fat}(t^{-1}, \mathrm{Dual}(v_\ell))$ decrease by 1. Thus $\mathrm{Fat}(R_4) = \mathrm{Fat}(R_2) - 2$. □

5.5. Weight reduction algorithm. The recognizer weight definition of section 3.5 remains in force.

Definition 5.33 (recognizer weight). *The* weight *of a recognizer R is a pair (i, j) of natural numbers, where i is the number of vertices of R and $j = Fat(R)$. The weights are ordered lexicographically with the number of vertices being the more significant component.*

Lemma 5.34 (weight reduction). *There is a polynomial-time* weight reduction algorithm *that reduces any recognizer to an equivalent lean regular recognizer.*

Proof. The proof is very close to the proof of the weight reduction lemma of section 3.5. There is a slight difference in the beginning, and so we give here that new beginning.

We construct an iterative algorithm that transforms the given recognizer by means of path folding; the algorithm halts when the recognizer is lean. By the folding lemma of section 5.4, the algorithm preserves the recognizer subgroup.

We describe one round of the algorithm and show that the weight decreases at each round. It will be obvious that the algorithm is polynomial time.

If the current recognizer R is regular and lean, halt. If there exists an irregular path, then identify the two ends of the path. This decreases the recognizer weight. Irregular paths were defined in section 4.

If there is a vertex ν with $\mathrm{Fat}(\nu) > 0$, find a witness $(t^\varepsilon, \nu, e_1, f_1)$ for this fact, where $\varepsilon \in \{1, -1\}$, ν is a vertex with $\mathrm{Fat}(t^\varepsilon, \nu) > 0$, and e_1 and f_1 are two t^ε-edges from ν of the sign of ε. We consider only the case $\varepsilon = 1$; the case $\varepsilon = -1$ is similar. Use the algorithm of the closed paths lemma to construct closed regular paths E and F continuing the e_1 and e_2, respectively.

The rest of the proof mimics the corresponding part of the proof of the weight reduction lemma of section 3.5. □

5.6. The theorem.

Theorem 5.35. *There is a polynomial-time decision algorithm for the membership problem for the succinct $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$. More explicitly, there is an algorithm such that*

(i) *given exponent words $h_1, \dots, h_m$ and w, the algorithm decides whether the subgroup H generated by $h_1, \dots, h_m$ contains w, and*

(ii) *the algorithm runs in time polynomial in $|h_1| + \cdots + |h_m| + |w|$.*

Proof. We describe the desired decision algorithm. Use the construction algorithm of section 2.2 to construct a recognizer R_1 for H. Use the weight reduction algorithm of section 5.5 to reduce R_1 into a lean regular recognizer R_2. Use the reading algorithm of section 5.3 to check whether the group element w belongs to H. Since all constituent algorithms are polynomial time, the decision algorithm is polynomial time. □

6. Main theorem.

We reiterate the main theorem formulated in section 1.

Theorem 6.1 (main). *The membership problem for the modular group* $\mathrm{PSL}_2(\mathbb{Z})$*, with integer entries in the standard decimal notation, is polynomial-time decidable.*

Proof. In the previous section, we proved the polynomial-time decidability of the membership problem for the succinct $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$. Thus it suffices to prove that the membership problem for the modular group is polynomial-time reducible to the membership problem for $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$. We construct the desired reduction.

By the modular group as a free product proposition in section 1, the modular group is isomorphic to $\langle s \mid s^2\rangle * \langle t \mid t^3\rangle$, where

$$s = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \qquad \text{and} \qquad t = \begin{pmatrix} 0 & 1 \\ -1 & 1 \end{pmatrix}.$$

Recall that a matrix is identified with its negative in the modular group.

Consider the four basic elementary transformations over the columns of unimodular matrices:

- subtract the first column from the second,
- subtract the second column from the first,
- add the second column to the first, and
- add the first column to the second.

Applying these transformations to the identity matrix, we get the basic elementary matrices

$$\begin{pmatrix}1 & -1\\ 0 & 1\end{pmatrix} \qquad \begin{pmatrix}1 & 0\\ -1 & 1\end{pmatrix} \qquad \begin{pmatrix}1 & 0\\ 1 & 1\end{pmatrix} \qquad \begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix}.$$

Multiplying a unimodular matrix M on the right by a basic elementary matrix E performs the corresponding column operation on M, and multiplying M by a E^k performs the column operation k times. Check that

$$st = \begin{pmatrix}1 & -1\\ 0 & 1\end{pmatrix}, \quad st^{-1} = \begin{pmatrix}1 & 0\\ -1 & 1\end{pmatrix}, \quad ts = \begin{pmatrix}1 & 0\\ 1 & 1\end{pmatrix}, \quad t^{-1}s = \begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix}.$$

We need to transform an arbitrary unimodular matrix M into an exponent word that represents M. Note that every exponent word w represents a unimodular matrix and thus gives rise to an operation $X \mapsto X \times w$ over unimodular matrices. It suffices to construct a polynomial-time procedure that reduces any unimodular matrix

$$M = \begin{pmatrix}a & b\\ c & d\end{pmatrix}$$

to the identity matrix by means of such operations.

Without loss of generality $a > 0$, because we can work either with M or with $-M$. To simplify exposition (though not the algorithm), we can assume that $b > 0$ as well. Indeed, if $b < 0$, then replace M with $M \times (t^{-1}s)^k$, where k is the least integer such that $ka > |b|$.

If $a \geq b$, then compute the number k such that $0 \leq a - kb < b$, and replace M with $M \times (st^{-1})^k$, so that we have $a \mod b$ in the left upper corner of the resulting matrix. Note that $a \mod b < a/2$. Indeed, if $b \leq a/2$, then $a \mod b < b \leq a/2$; otherwise, $b > a/2$, and $a \mod b \leq a - b < a/2$. Similarly, if $a < b$, then compute the number k such that $0 \leq b - ka < a$, and replace M with $M \times (st)^k$, so that we have $b \mod a$ in the right upper corner of the resulting matrix. We have $b \mod a \leq b/2$.

Keep doing that until you have zero in one of the upper corners. In every two steps, the entries in both upper corners are more than halved. It follows that this iteration works in linear time.

Thus, without loss of generality, we may assume that the left upper entry a of the given matrix M is zero; the case when the right upper entry b is zero is similar. So

$$M = \begin{pmatrix}0 & b\\ c & d\end{pmatrix}.$$

Further, we may assume without loss of generality that $b = 1$ and $c = -1$ because the determinant of M is 1, and we can work with either M or its negative. Replace M with $(M \times t^{-1}s)^d$; the result is the matrix

$$s = \begin{pmatrix}0 & 1\\ -1 & 0\end{pmatrix}.$$

Finally replace the resulting matrix s by $s \times s$, and get (the negative of) the identity matrix. □

REFERENCES

[1] A. Blass and Y. Gurevich, *Matrix transformation is complete for the average case*, SIAM J. Comput., 22 (1995), pp. 3–29.

[2] J.-Y. Cai, W. H. J. Fuchs, D. Kozen, and Z. Liu, *Efficient Average-case algorithms for the modular group*, in Proceedings of the 35th Annual Symposium on Foundations of Computer Science, IEEE Press, Piscataway, NJ, 1994, pp. 143–152.

[3] Y. Gurevich, *Average case completeness*, J. Comput. System Sci., 42 (1991), pp. 346–398.

[4] Y. Gurevich, *Matrix decomposition problem is complete for the average case*, in Proceedings of the 31st Annual Symposium on Foundations of Computer Science, IEEE Press, Piscataway, NJ, 1994, pp. 802–811.

[5] I. Kapovich and A. Miasnikov, *Stallings foldings and subgroups of free groups*, J. Algebra, 248 (2002), pp. 608–668.

[6] L. Levin, *Average case complete problems*, SIAM J. Comput., 15 (1986), pp. 285–286.

[7] R. Lyndon and P. E. Schupp, *Combinatorial Group Theory*, Springer-Verlag, Berlin, 1977.

[8] W. Magnus, A. Karrass, and D. Solitar, *Combinatorial Group Theory: Presentations of Groups in Terms of Generators and Relations*, Dover, New York, 1976.

[9] P. E. Schupp, *Coxeter groups, 2-completion, perimeter reduction and subgroup separability*, Geom. Dedicata, 96 (2003), pp. 179–198.

[10] J. R. Stallings, *Topology of finite graphs*, Invent. Math., 71 (1983), pp. 551–565.

SIAM J. COMPUT.
Vol. 37, No. 2, pp. 460–481

APPROXIMATION ALGORITHMS FOR CONSTRAINED NODE WEIGHTED STEINER TREE PROBLEMS[*]

A. MOSS[†] AND Y. RABANI[†]

Abstract. We consider a class of optimization problems where the input is an undirected graph with two weight functions defined for each node, namely the node's *profit* and its *cost*. The goal is to find a connected set of nodes of low cost and high profit. We present approximation algorithms for three natural optimization criteria that arise in this context, all of which are NP-hard. The *budget* problem asks for maximizing the profit of the set subject to a budget constraint on its cost. The *quota* problem requires minimizing the cost of the set subject to a quota constraint on its profit. Finally, the *prize collecting* problem calls for minimizing the cost of the set plus the profit (here interpreted as a penalty) of the complement set. For all three problems, our algorithms give an approximation guarantee of $O(\log n)$, where n is the number of nodes. To the best of our knowledge, these are the first approximation results for the quota problem and for the prize collecting problem, both of which are at least as hard to approximate as the set cover. For the budget problem, our results improve on a previous $O(\log^2 n)$ result of Guha et al. Our methods involve new theorems relating tree packings to (node) cut conditions. We also show similar theorems (with better bounds) using edge cut conditions. These imply bounds for the analogous budget and quota problems with edge costs which are comparable to known (constant factor) bounds.

Key words. combinatorial approximation algorithms, network design, node-weighted problems

AMS subject classifications. 68W25, 90C27

DOI. 10.1137/S0097539702420474

1. Introduction.

We consider optimization problems on graphs with node costs and profits. Let $G = (V, E)$ be an undirected graph. Let every node $v \in V$ have a nonnegative cost $c(v)$ and a nonnegative profit $\pi(v)$. Our goal is to find a connected set of nodes $S \subset V$ that has low cost and high profit. There are several ways to cast this goal as an optimization criterion. In this paper we consider three such problems. In the *quota* problem, we are given a profit quota Q, and we have to find a set S of the minimum total cost among those with a total profit of at least Q. In the *budget* problem, the total cost is constrained to be at most a budget B. Subject to this constraint, we have to find a set S of the maximum total profit. Finally, in the *prize collecting* problem, the objective is to minimize the total cost of S plus the total profit of $V \setminus S$ (the profit loss). All three problems also have a *rooted* version, where S must contain a specified node $r \in V$. Clearly, an algorithm for the rooted version can be used to solve the unrooted version, by enumerating over all possible roots.

We give $O(\log |V|)$ approximation guarantees for the rooted (and thus the unrooted) quota problem, for the (rooted) budget problem, and for the (rooted) prize collecting problem. Our result for the budget problem is a bicriteria approximation in the sense that the approximation algorithm is allowed to exceed the budget by a factor of 2. The quota problem as well as the prize collecting problem are at least as hard to approximate as the set cover, so our guarantees are asymptotically the best possible under reasonable assumptions on the complexity of NP-hard problems (see, for example, the survey by Arora and Lund (1997)). By the same token, the

[*] Received by the editors December 30, 2002; accepted for publication (in revised form) October 19, 2006; published electronically May 22, 2007.

http://www.siam.org/journals/sicomp/37-2/42047.html

[†] Computer Science Department, Technion City, Haifa, Israel 32000 (annaru@cs.technion.ac.il, rabani@cs.technion.ac.il).

budget problem is at least as hard to approximate as the maximum coverage, so there is a constant lower bound on its approximability under the same assumptions (see Khuller, Moss, and Naor (1998)). In addition to these new results, our methods can be used to derive previously discovered approximation guarantees for some related problems. Our solutions are more uniform, and perhaps more elegant, than previous results. See the discussion below for more details.

Most previous related research was done on closely related problems with edge costs instead of node costs. (The cost of a set of nodes is the total cost of its minimum spanning tree.) These problems are NP-hard, and in recent years constant factor approximation algorithms were found for most of them. Goemans and Williamson (1995) give a primal-dual $2 - \frac{1}{n-1}$ approximation algorithm for the prize collecting Steiner tree problem, which is the version with edge costs of our prize collecting problem. Following the groundbreaking work of Blum, Ravi, and Vempala (1999), Garg (1996) gives a 3-approximation (as well as a simpler 5-approximation) for k-minimum spanning tree (MST), which is a special case of the version with edge costs of our quota problem, with unit profits and a quota of k. Arora and Karakostas (2006) improve Garg's approximation guarantee to $2 + \epsilon$ for all $\epsilon > 0$. The constant factor approximations of the latter three works are derived by using the Goemans–Williamson prize collecting Steiner tree algorithm on the graph with uniform profits. The profit is determined (using binary search) to yield a tree (more precisely, a convex combination of two trees) of size k. As noted by Chudak, Roughgarden, and Williamson (2004), Garg's k-MST 5-approximation algorithm can be recast in terms of a Lagrangian relaxation for the problem, similar to the treatment by Jain and Vazirani (2001) of the k-median problem. Recently, Garg (2005) gave a factor 2 approximation for k-MST.

Awerbuch et al. (1998) (in a previous k-MST paper giving polylogarithmic approximation guarantees) note that a polynomial time k-MST approximation algorithm can be used to get a pseudopolynomial time approximation algorithm with the same guarantee for the (version with edge costs) quota problem. Johnson, Minkoff, and Phillips (2000) note that, for the above-mentioned constant approximation k-MST algorithms, the pseudopolynomial time algorithm can be converted into a truly polynomial time one. Moreover, they show that such an algorithm can be used to solve the (version with edge costs) unrooted budget problem, losing a constant factor in the approximation guarantee.

The budget problem (with node costs) was introduced by Guha et al. (1999), who motivate it by applications to problems in the maintenance of electric power networks. They give an $O(\log^2 |V|)$ approximation guarantee for the unrooted version. They also show how to obtain a similar approximation guarantee for the rooted version using a set that has a total cost at most twice the allowed budget B. Their algorithm is based on an $O(\log |V|)$ approximation algorithm for the node weighted Steiner tree problem by Klein and Ravi (1995). This problem is similar to the well-known Steiner tree problem, except that nodes and not edges have costs. The problem is at least as hard to approximate as the set cover. As pointed out by Guha et al., the algorithm of Klein and Ravi is implicitly a primal-dual algorithm.

The starting point for all of our algorithms is a standard linear programming relaxation for the rooted version, where nodes are chosen fractionally to be in the output set S, and the connectivity requirement is expressed as constraints on (node) cuts. For the quota and budget problems we show that any feasible solution to the linear programming relaxation can be approximated by a convex combination of connected sets of nodes containing the root. Moreover, such a convex combination is computable in polynomial time. It follows from an averaging argument that a good set can be found. Our algorithm for the prize collecting problem is used to construct the convex combination. It combines ideas from both the Klein–Ravi node weighted Steiner tree algorithm (with its primal-dual interpretation by Guha et al.) and the Goemans–Williamson prize collecting Steiner tree algorithm. Both the region growing process and the final delete step are more complicated in our algorithm than in either of those algorithms.

We then use the solution to the prize collecting problem to prove the following theorem.

THEOREM 1. *Let $G = (V, E)$ be an undirected graph with nonnegative node weights $d : V \to \mathbb{Q}^+$. Let $r \in V$, and assume that, for every $v \in V$, $d(v) \leq 1$ and $d(r) = 1$. Furthermore assume that, for every node $v \in V \setminus \{r\}$, the minimum weight node cut separating r and v has a weight of at least $d(v)$. Then there exist polynomial time algorithms that compute*

1. *a packing in G of connected sets containing r such that for every node $v \in V$ the total weight of sets containing v is between $d(v)/c \log |V|$ and $d(v)$ and*
2. *a packing in G of connected sets containing r such that for every node $v \in V$ the total weight of sets containing v is between $d(v)$ and $\min\{1, d(v)c \log |V|\}$,*

where, in both cases, c is an absolute constant. Notice that the second claim implies the first. We state both claims for ease of application.

For optimization problems that can be formulated as positive integer programs, Carr and Vempala (2002) prove that an approximation algorithm for the problem can be used to generate a packing of integer solutions that approximates a feasible solution to the linear programming (LP) relaxation of the integer program with the same guarantee. Our results extend their results and use an approximation algorithm for a different problem to generate the packing. Our approximation guarantees for the budget and quota problems follow from applying the packing algorithms to the solutions for the linear programming relaxations we use and then using averaging arguments to find good integer solutions.

Using techniques similar to those used to prove Theorem 1, one can show the following theorem. We omit the proof.

THEOREM 2. *Let $G = (V, E)$ be an undirected graph with nonnegative node weights $d : V \to \mathbb{Q}^+$ and nonnegative edge capacities $c : E \to \mathbb{Q}^+$. Let $r \in V$, with $d(r) = 1$. Assume that, for every node $v \in V \setminus \{r\}$, the minimum capacity edge cut separating r and v has a capacity of at least $d(v)$. Then there exist polynomial time algorithms that compute*

1. *a packing in G of trees rooted at r such that for every node $v \in V$ the total weight of trees containing v is between $d(v)/2$ and $d(v)$ and for every edge $e \in E$ the total weight of trees containing e is at most $c(e)$ and*
2. *a packing in G of trees rooted at r such that for every node $v \in V$ the total weight of trees containing v is between $d(v)$ and 1 and for every edge $e \in E$ the total weight of trees containing e is at most $2c(e)$.*

The celebrated theorem of Nash-Williams (see Diestel (2000)) implies such packings in the case of $d(v) = 1$ for all $v \in V$. One of the consequences of Theorem 2 is a modification of Garg's 5-approximation for k-MST avoiding the Lagrangian relaxation. A similar modification (using another packing theorem) can be applied to the 6-approximation for the metric k-median of Jain and Vazirani. It is not clear if the better algorithms that use continuity properties of the Lagrangian relaxation (for example, Garg's 3-approximation for k-MST or the 4-approximation for the metric k-median of Charikar and Guha (2005)) can be derived from our packings.

We observe that the prize collecting problem can be viewed as the Lagrangian relaxation of the quota problem. Thus, the Lagrangian relaxation method used in Garg's 5-approximation for k-MST and 6-approximation for the metric k-median of Jain and Vazirani can be applied to the quota problem and together with our algorithm for the prize collecting problem yields the same approximation guarantee for the quota problem as our tree packings method. Furthermore, the algorithm for the quota problem can be used to obtain an approximation algorithm for the budget problem by performing a binary search on the quota space. Again, the latter method yields the same approximation guarantee for the budget problem as the tree packings method. The detailed description of these solutions for the quota and the budget problems has been presented by Moss (2001). Though our results for the quota and the budget problems can be obtained using the Lagrangian relaxation method, we wish to introduce the tree packings method as an interesting alternative technique.

The rest of the paper is organized as follows. In section 2, we present the primal-dual algorithm for the prize collecting problem. In section 3, we present the proof of Theorem 1. In section 4, we give the algorithm for the quota problem. Finally, in section 5, we give the algorithm for the budget problem.

2. The prize collecting problem. In the prize collecting problem, one is given an undirected graph $G = (V, E)$ with a cost function $c : V \to \mathbb{Q}^+$, a profit function $\pi : V \to \mathbb{Q}^+$, and a specified root r. The objective is to find a subtree T of G containing the root r such that $\sum_{v \in T} c(v) + \sum_{v \notin T} \pi(v)$ is minimized. In this section we present an approximation algorithm for the prize collecting problem and analyze its approximation ratio. This algorithm is used to obtain the results in the following sections.

2.1. The algorithm. Let $V' = V \setminus \{r\}$. Observe that the prize collecting problem can be formulated as the following integer program:

$$\text{minimize} \sum_{v \in V'} c(v) x_v + \sum_{S \subseteq V'} \pi(S) z_S \text{ subject to (s.t.)}$$

$$\sum_{v \in \Gamma(S)} x_v + \sum_{T | S \subseteq T} z_T \geq 1 \ \forall S \subseteq V', \tag{1}$$

$$x_v + \sum_{T | v \in T} z_T \geq 1 \qquad \forall v \in V', \tag{2}$$

$$x_v \in \{0,1\} \qquad \forall v \in V',$$

$$z_S \in \{0,1\} \qquad \forall S \subseteq V',$$

where $x_r = 1$ and $\Gamma(S) = \{v \mid \exists uv \in E \text{ s.t. } u \in S, v \notin S\}$. A linear programming relaxation (denoted PC-LP) is obtained by replacing the last two sets of constraints with $x, z \geq 0$.

Consider the dual program for PC-LP, denoted by PC-D:

$$\text{maximize} \quad \sum_{S \subseteq V'} y_S + \sum_{v \in V'} p_v \quad \text{s.t.}$$

$$\sum_{S | v \in \Gamma(S)} y_S + p_v \leq c(v) \ \ \forall v \in V',$$

$$\sum_{S | S \subseteq T} y_S + \sum_{v \in T} p_v \leq \pi(T) \forall T \subseteq V',$$

$$y, p \geq 0.$$

The algorithm we propose for the prize collecting problem is a primal-dual algorithm which greedily constructs an implicit solution to the dual problem PC-D.

Partition the vertices of V' into *cheap* vertices, for which $c(v) \leq \pi(v)$, and *expensive* vertices, which are the rest of V'. Implicitly set $p_v = c(v)$ for each cheap v and $p_v = \pi(v)$ for each expensive v. Consider the connected components induced by cheap vertices v and the root r. Let $\mathcal{P}$ denote the set of these components.

For each vertex $v \in V'$, define the *residual cost* $c_r(v) = c(v) - p_v$ and the *residual penalty* $\pi_r(v) = \pi(v) - p_v$. The algorithm greedily constructs a packing $\{y_S\}_{S \subseteq V'}$ which together with the values of p_v, $v \in V'$, determined above, is a feasible solution to PC-D; i.e., it satisfies

$$\sum_{S | v \in \Gamma(S)} y_S \leq 0 \qquad \forall \text{cheap } v \in V',$$

$$\sum_{S | v \in \Gamma(S)} y_S \leq c_r(v) \quad \forall \text{expensive } v \in V', \tag{3}$$

$$\sum_{S \subseteq T} y_S \leq \sum_{\text{cheap} v \in T} \pi_r(v) \forall T \subseteq V', \tag{4}$$

$$y \geq 0.$$

The algorithm maintains a set $\mathcal{C}$ of *active* and *inactive* components. We say that C is *deactivated* iff it changes its status from active to inactive. Initially, $\mathcal{C} \leftarrow \mathcal{P}$, and all of the components $C \in \mathcal{C}$ are active, except for the component containing the root. We raise uniformly the dual variables y_S corresponding to each active component, until either constraint (3) becomes tight for some expensive vertex v or constraint (4) becomes tight for some active component $\tilde{C}$. In the former case, the vertex v joins a component C such that $v \in \Gamma(C)$, and, in case there are several such components, all of these components are merged. In the latter case, the component $\tilde{C}$ gets deactivated. The component containing the root remains inactive throughout the algorithm, and the process terminates when no active components remain. The algorithm also builds a *connection tree* T, which contains edges by which expensive vertices join components. At the end of the algorithm, we perform a deletion step on the tree T. During this step, a set of vertices $S \subseteq T$ is *expendable* iff its removal does not disconnect any remaining cheap vertex from r. The code in Figure 2.1 defines the algorithm more rigorously. The variables $w(C)$ maintain the values $\sum_{S \subseteq C} y_S$, and the variables $d(v)$ maintain the values $\sum_{S | v \in S} y_S$. The algorithm does not need to maintain explicitly the values y_S. Their computation is included in the code for the purpose of analysis.

2.2. Analysis. Next we prove Theorem 3, which establishes an $O(\log n)$ approximation guarantee for the primal-dual algorithm.

THEOREM 3.

$$\sum_{v \in T} c(v) + \beta \sum_{v \notin T} \pi(v) \leq \beta \left(\sum_{S \subseteq V'} y_S + \sum_{v \in V'} p_v \right),$$

where $\beta = O(\log n)$.

Proof. Let A be the set of cheap vertices not spanned by the final solution T, and let F denote the set of cheap vertices spanned by T. Also, let N be the set of expensive vertices spanned by T. Recall that, for each cheap vertex v, $p_v = c(v)$

```
**PrizeCollecting**

**Initialize:**
$\mathcal{C} \leftarrow \mathcal{P}$
$y_C \leftarrow 0$ for all $C \in \mathcal{C}$
$w(C) \leftarrow 0$ for all $C \in \mathcal{C}$
$d(v) \leftarrow 0$ for all $v \in V$
All $C \in \mathcal{C}$ are active, except for $C \ni r$
$T \leftarrow$ the union of spanning trees for $C \in \mathcal{C}$

**Main loop:**
**while** $\exists$active $C \in \mathcal{C}$ **do**
    Let $d_v(C) = \max_{u \in C | \exists uv \in E} d(u)$
    Let $\epsilon_1(v) = \frac{c_r(v) - \sum_{C | v \in \Gamma(C)} d_v(C)}{|\{\text{active } C | v \in \Gamma(C)\}|}$
    Find $\tilde{v} \in V \setminus \bigcup C \in \mathcal{C}$ that minimizes $\epsilon_1 = \epsilon_1(\tilde{v})$
    Let $\epsilon_2(C) = \sum_{\text{cheap } i \in C} \pi_r(i) - w(C)$
    Find an active $\tilde{C} \in \mathcal{C}$ that minimizes $\epsilon_2 = \epsilon_2(\tilde{C})$
    $\epsilon \leftarrow \min\{\epsilon_1, \epsilon_2\}$
    $y_C \leftarrow y_C + \epsilon$ for all active $C \in \mathcal{C}$
    $w(C) \leftarrow w(C) + \epsilon$, for all active $C \in \mathcal{C}$
    $d(v) \leftarrow d(v) + \epsilon$, for all active $C \in \mathcal{C}$, $v \in C$
    **if** $\epsilon = \epsilon_2$ **then**
        Deactivate $\tilde{C}$
        Label all $v \in \tilde{C}$ by label $\tilde{C}$
     **else**
        Let $\overline{\mathcal{C}} = \{C \in \mathcal{C} \mid \tilde{v} \in \Gamma(C)\}$
        Let $\overline{C} = (\bigcup(C \in \overline{\mathcal{C}})) \bigcup \{\tilde{v}\}$
        Let $u_C = \text{argmax}\{d(w) \mid w \in C,\ w\tilde{v} \in E\}$, $\forall C \in \overline{\mathcal{C}}$
        $w(\overline{C}) \leftarrow \sum_{C | \tilde{v} \in \Gamma(C)} w(C)$
        $\mathcal{C} \leftarrow (\mathcal{C} \bigcup \{\overline{C}\}) \setminus \{C \mid \tilde{v} \in \Gamma(C)\}$
        **if** $r \in \overline{C}$ **then**
            $\overline{C}$ is inactive
         **else**
            $\overline{C}$ is active
        **endif**
        Add $\tilde{v}$ to the node set of $T$
        Add $u_C\tilde{v}$ to the edge set of $T$, for every $C \in \overline{\mathcal{C}}$
    **endif**
**endwhile**

**Deletion step:**
Remove from $T$ nodes that are disconnected from $r$.
**while** $\exists$cheap $v$ labeled $C$ s.t. $C \bigcup \Gamma(C)$ is expendable **do**
    Remove $C \bigcup \Gamma(C)$ from $T$
**endwhile**
**while** $\exists$expensive $v \in T$ that is expendable **do**
    Remove $v$ from $T$
**endwhile**

Output $T$.
```

FIG. 2.1. *Algorithm* **PrizeCollecting** *for the prize collecting problem.*

and, for each expensive vertex v, $p_v = \pi(v)$. Therefore, to establish the claim of the theorem, it suffices to prove that the following inequality holds:

$$\sum_{v \in N} c_r(v) + \beta \sum_{v \in A} \pi_r(v) \leq \beta \sum_{S \subseteq V'} y_S,$$

where $\beta = O(\log n)$. We establish the inequality above by proving that the following two inequalities hold:

$$\sum_{v \in N} c_r(v) \leq \beta \sum_{S | S \bigcap F \neq \emptyset} y_S, \tag{5}$$

$$\sum_{v \in A} \pi_r(v) \leq \sum_{S | S \bigcap F = \emptyset} y_S. \tag{6}$$

First, observe that the inequality (6) follows from the greedy construction of the dual solution y by the algorithm. Indeed, consider the set $\mathcal{M}$ of maximal components deactivated by the algorithm due to the residual penalty constraints (4) such that no vertex of the component is included in the final solution T. Clearly, these components are disjoint, and as each S with a nonzero value of y_S contains some cheap vertex, we get

$$\sum_{S | S \bigcap F = \emptyset} y_S = \sum_{C \in \mathcal{M}} \sum_{S \subseteq C} y_S = \sum_{C \in \mathcal{M}} \sum_{v \in A \bigcap C} \pi_r(v) = \sum_{v \in A} \pi_r(v).$$

To see the last equality, notice that by the elimination step of the algorithm every vertex in A is included in some deactivated component not containing a vertex from F.

In the rest of the proof we show that the inequality (5) holds. In what follows we refer to a set of vertices S as *tree-bound* if it contains a cheap vertex spanned by the final tree T constructed by the algorithm (i.e., $S \bigcap F \neq \emptyset$). Similarly, a component maintained by the algorithm will be called *tree-bound* if it contains a cheap vertex spanned by T.

Before proceeding with the proof, we wish to outline its main ideas. We observe again that the ideas of the proof are similar to those of the Klein–Ravi node weighted Steiner tree algorithm and its primal-dual interpretation by Guha et al. But since the component growing process in our algorithm is more complicated and, unlike that of the Steiner tree algorithm, includes deactivation of components, our proof incorporates special treatment of deactivated components.

We need to show that the residual cost of expensive vertices spanned by T is bounded, within a logarithmic factor, by the sum of dual values of tree-bound sets. We divide the algorithm into phases so that at the end of each phase the number of tree-bound active components decreases. Clearly, a phase can end either by a merge of two or more tree-bound active components or the root component or by deactivation of a tree-bound active component. Thus a phase can span one or more iterations of the algorithm.

During each phase, we pay the residual cost of some expensive vertices spanned by T, that have not yet been paid. The exact subset of expensive vertices paid for at each phase will be defined later. In the end of the process all of the expensive vertices in T get paid for. We show that, if the number of tree-bound active components is m in the beginning of a phase and $m-k$ at the end of the phase, then the residual cost paid during the phase is at most $\frac{k+1}{m}$ times the sum of the dual values of the tree-bound sets. Notice that the initial number of tree-bound active components is bounded by the number of cheap vertices and therefore is at most n. It is not hard to see that the maximum cost will be payed if at each phase the number of tree-bound components is decreased by one, in which case the total cost paid until the beginning of the last iteration is at most the sum of the dual values multiplied by $\sum_{i=2}^{n} \frac{2}{i} = O(\log n)$.

Next we continue with the proof of the theorem. We refer to expensive vertices that caused two or more components (active or inactive) to be merged as *merge vertices*. We proceed by establishing the following properties of the dual solution constructed by the algorithm. Note that it can be easily shown by induction that for all v, $d(v) = \sum_{S|v\in S} y_S$ at each step of the algorithm.

LEMMA 4. *Let $v_0, v_1, \ldots, v_k$ be a simple path in the connection tree T such that for each i, $1 \leq i \leq k$, v_i is not a merge vertex, and v_i gets connected to T by an edge (v_{i-1}, v_i). Then at the moment v_k gets connected to T the following holds:*

$$\sum_{S|v_0\in S} y_S = \sum_{l=1}^{k} c_r(v_l).$$

Proof. We prove the claim of the lemma by induction on k. If $k = 1$, let $d(v_0) = \sum_{S|v_0\in S} y_S = p$ before the increase of dual variables caused by v_1. Then $\epsilon_1 = c_r(v_1) - p$, and, after the increase of dual variables by ϵ_1,

$$\sum_{S|v_0\in S} y_S = c_r(v_1),$$

proving the basis of the induction. For the induction step, suppose that the claim of the lemma holds for vertices $v_1, \ldots, v_{k-1}$, and consider the moment when v_k gets connected to T via the edge (v_{k-1}, v_k). By the induction hypothesis, at the moment v_{k-1} entered T, it held that $\sum_{S|v_0\in S} y_S = \sum_{l=1}^{k-1} c_r(v_l)$. Let $d(v_{k-1}) = q$ at the current iteration. As v_{k-1} participates only in those sets that got nonzero values after v_{k-1} entered T, it holds that

$$\begin{aligned}\sum_{S|v_0\in S} y_S &= \sum_{S|v_0\in S, v_{k-1}\notin S} y_S + \sum_{S|v_0\in S, v_{k-1}\in S} y_S \\ &= \sum_{l=1}^{k-1} c_r(v_l) + q.\end{aligned}$$

The increase ϵ_1 in the dual variables, caused by v_k, is $c_r(v_k) - q$, and therefore, when v_k joins T, we have

$$\begin{aligned}\sum_{S|v_0\in S} y_S &= \sum_{l=1}^{k-1} c_r(v_l) + q + c_r(v_k) - q \\ &= \sum_{l=1}^{k} c_r(v_l). \qquad \square\end{aligned}$$

LEMMA 5. *Let $C_1, C_2, \ldots, C_k$ be components merged via a merge vertex v so that $1, \ldots, C_{k'}$ are active at the time of the merge and $C_{k'+1}, \ldots, C_k$ are inactive. Let $u_i = \text{argmax}\{d(w) \mid (w, v) \in E,\ w \in C_i\}$, $i = 1, \ldots, k$; that is, (u_i, v) is an edge in T*

by which v gets connected to C_i. Then immediately after the merge

$$\sum_{i=1}^{k} \sum_{S|u_i \in S} y_S = c_r(v).$$

Proof. Put

$$\sum_{S|u_i \in S} y_S = q_i$$

before the merge for each $i = 1, \dots, k$. Then the increase of dual variables induced by v is

$$\epsilon_1 = \frac{c_r(v) - \sum_{i=1}^{k} q_i}{k'}.$$

Since the dual variable of each of the components $C_1, C_2, \dots, C_{k'}$ gets increased by this amount, after the merge it holds that

$$\sum_{i=1}^{k} \sum_{S|u_i \in S} y_S = \sum_{i=1}^{k} q_i + \sum_{i=1}^{k'} \frac{c_r(v) - \sum_{j=1}^{k} q_j}{k'} = c_r(v). \qquad \square$$

Following the notation of Lemma 5, define the *partial cost* of v with respect to the component C_i as $cost_\delta^{C_i}(v) = \sum_{S|u_i \in S} y_S$. Clearly, if C_i is inactive at the time of the merge, then $cost_\delta^{C_i}(v) = q_i$; otherwise, $cost_\delta^{C_i}(v) = q_i + \epsilon_1$. By Lemma 5 it follows that

$$\sum_{i=1}^{k} cost_\delta^{C_i}(v) = c_r(v). \tag{7}$$

Recall that we partition the algorithm into phases such that in the end of each phase the number of tree-bound active components decreases. Let ϕ_i denote the number of tree-bound active components at the end of phase i. If the phase i ends by a merge of two or more tree-bound active components, let h_i denote the number of tree-bound active components merged at phase i. We refer to such a phase as a *merge phase*. Otherwise, if the phase i ends by deactivation of a tree-bound active component, let h_i=2. We refer to such a phase as a *deactivation phase*.

Next we wish to assign expensive vertices of T that are not merge vertices to phases so that the residual cost of each expensive vertex will be paid at some phase. Let k be the total number of merge phases. Let T_j, $j = 1, \dots, k$, be a collection of subtrees of T spanning cheap vertices in tree-bound components in the end of the jth merge phase. Because of the deletion step of the algorithm it holds that $T_k = T$. Now consider a merge phase i. We wish to pay for expensive vertices added to connect the components merged at phase i. Formally, let $\tilde{i}$ be the last merge phase of the algorithm preceding the merge phase i. Then the collection D_i of expensive vertices assigned to phase i is defined as vertices of $T_i \setminus T_{\tilde{i}}$, not including merge vertices and the vertices that belong to components inactive at the beginning of phase i. The latter vertices are paid for at deactivation phases of the corresponding inactive components. Thus, to define D_i for a deactivation phase i, ending by deactivation of component C, consider the earliest merge phase j following phase i such that, at the end of phase

j, C becomes part of some active component or the root component. Let $\tilde{j}$ be the last merge phase preceding the phase j. Then D_i is defined as the set of expensive vertices in $(T_j \setminus T_{\tilde{j}}) \bigcap C$.

Finally, we wish to pay the residual cost of merge vertices. The cost of such vertices may be divided between several phases. Formally, if v is a merge vertex between active components $C_1, \ldots, C_{k'}$ and inactive components $C_{k'+1}, \ldots, C_k$, then $\{cost_\delta^{C_i}(v)\}_{i=1}^{k'}$ will be assigned to the merge phase during which the merge takes place, and for each $i = k' + 1, \ldots, k$, $cost_\delta^{C_i}(v)$ will be assigned to the deactivation phase at which the component C_i was deactivated. Observe that by (7) the partial costs of v sum up to its residual cost.

For each phase i, let $cost(phase_i)$ denote the sum of residual costs of vertices in D_i and partial costs of merge vertices assigned to phase i. Observe that $\sum_i cost(phase_i) = \sum_{v \in N} c_r(v)$; that is, by the end of the last phase all the expensive vertices get paid for.

The following lemma proves that the cost paid at each phase of the algorithm is bounded in terms of the dual solution.

LEMMA 6. *For each phase i,*

$$cost(phase_i) \leq \frac{h_i}{\phi_{i-1}} \sum_{S | S \bigcap F \neq \emptyset} y_S.$$

Proof. Let Δ denote the sum of the values of ϵ for each iteration of the algorithm until the end of phase i.

First, consider a merge phase i ending in a merge of tree-bound active components $C_1, C_2, \ldots, C_{h_i}$ via a merge vertex v. Let ϵ_m denote the value of ϵ in the last iteration of the phase (i.e., the iteration ending by the merge via v). By definition of D_i it follows that D_i is a collection of h_i paths from v to some cheap vertices $t_j \in C_j$, $j = 1, \ldots, h_i$, excluding the vertices of the components inactive at the beginning of phase i. Let P_j, $j = 1, \ldots, h_i$, denote these paths. Let $P_j = \{t_j = v_0^j, v_1^j, \ldots, v_{p-1}^j, v_p^j = v\}$. Observe that the partial costs of v assigned to phase i are $\sum_{l=1}^{h_i} \sum_{S | v_{p-1}^l \in S, v \notin S} y_S + \epsilon_m \cdot h_i$. We will attribute the part $\sum_{S | v_{p-1}^j \in S, v \notin S} y_S + \epsilon_m$ to each path P_j. It can be seen that in this way we have divided all of the partial costs of v assigned to phase i between the h_i paths.

Now we will bound the contribution of each path P_j to $cost(phase_i)$. First, consider the simple case when P_j does not contain merge vertices. Then by Lemma 4,

$$\sum_{S | t_j \in S, v_{p-1}^j \notin S} y_S = \sum_{l=1}^{p-1} c_r(v_l^j).$$

Therefore, the contribution of the path P_j to $cost(phase_i)$ is

$$\sum_{S | t_j \in S, v_{p-1}^j \notin S} y_S + \sum_{S | v_{p-1}^j \in S, v \notin S} y_S + \epsilon_m \leq (\Delta - \epsilon_m) + \epsilon_m = \Delta.$$

The last inequality follows since only one component containing t_j is active at each time the dual variables are increased by a value of ϵ, and the sum of all values of ϵ before the final merge is $\Delta - \epsilon_m$.

Since there might occur merges involving one active and one or more inactive tree-bound components during the phase i, consider the second case when the path

P_j passes through merge vertices and through vertices of inactive components. For the sake of simplicity of presentation, we consider the case when P_j contains one merge vertex and vertices of one inactive component. The case of multiple merge vertices can be treated similarly by inductive argument. Recall that $P_j = \{t_j = v_0^j, v_1^j, \ldots, v_{p-1}^j, v_p^j = v\}$, and let v_m^j be a merge vertex, causing the merge of the active component C_j and inactive tree-bound components $C^1, C^2, \ldots, C^l$, $l \geq 1$. P_j may pass through one of the components $C^t, t = 1, \ldots, l$, which is without loss of generality C^1. Let $q > m$ be the smallest index such that v_q is not contained in C^1. Observe that if P_j does not pass through C^1, then $q = m + 1$. By Lemma 4,

$$\sum_{S | t_j \in S, v_{m-1}^j \notin S} y_S = \sum_{l=1}^{m-1} c_r(v_l^j),$$

and also

$$\sum_{S | t_j \in S, v_{m-1}^j \in S, v_{p-1}^j \notin S} y_S = \sum_{l=q}^{p-1} c_r(v_l^j).$$

Furthermore, the partial cost of v_m assigned to phase i is $\sum_{S | v_{m-1}^j \in S, v_m^j \notin S}$. Therefore, in this case the contribution of P_j to $\mathit{cost}(\mathit{phase}_i)$ is

$$\sum_{S | t_j \in S, v_{m-1}^j \notin S} y_S + \sum_{S | t_j \in S, v_{m-1}^j \in S, v_{p-1}^j \notin S} y_S + \sum_{S | v_{m-1}^j \in S, v_m^j \notin S} y_S + \sum_{S | v_{p-1}^j \in S, v \notin S} y_S \leq \Delta,$$

where the last inequality follows by the same argument as in the previous case.

Since there are h_i paths to v contributing to $\mathit{cost}(\mathit{phase}_i)$, we get that

$$\mathit{cost}(\mathit{phase}_i) \leq h_i \Delta.$$

Now consider a phase i ending in deactivation of a component C^i. Using the same notation as in the previous case, D_i is a set of at most two paths from vertices v_{m+1}^j and v_{q-1}^j to some cheap vertices in C^i. By the same reasoning as in the case of a merge phase, each of these paths contributes at most Δ to $\mathit{cost}(\mathit{phase}_i)$; therefore, we get $\mathit{cost}(\mathit{phase}_i) \leq 2\Delta = h_i \Delta$.

Observe that each component active at the beginning of phase i must contain a vertex that has always been in some active component since the beginning of the algorithm. Clearly, for such a vertex t, $\sum_{S | t \in S} y_S = \Delta$. Therefore,

$$\sum_{S | S \bigcap F \neq \emptyset} y_S \geq \phi_{i-1} \Delta.$$

Therefore, we get

$$\begin{aligned} \mathit{cost}(\mathit{phase}_i) &\leq h_i \Delta \\ &\leq \frac{h_i \Delta}{\phi_{i-1} \Delta} \sum_{S | S \bigcap F \neq \emptyset} y_S \\ &= \frac{h_i}{\phi_{i-1}} \sum_{S | S \bigcap F \neq \emptyset} y_S. \qquad \square \end{aligned}$$

Let m denote the total number of phases. By Lemma 6 and the argument mentioned in the beginning of the proof, it follows that

$$\sum_{i=1}^{m} cost(phase_i) \leq O(\log n) \sum_{S|S\bigcap F\neq\emptyset} y_S.$$

By Lemma 6, the cost added at the last phase of the algorithm is at most

$$\begin{aligned}\frac{h_m}{\phi_{m-1}} \sum_{S|S\bigcap F\neq\emptyset} y_S &\leq \frac{h_m}{h_m-1} \sum_{S|S\bigcap F\neq\emptyset} y_S \\ &\leq 2 \sum_{S|S\bigcap F\neq\emptyset} y_S.\end{aligned}$$

Therefore, the inequality (5) follows. This completes the proof of Theorem 3. □

3. Tree packings. In this section we prove Theorem 1. We prove part 1 of the theorem first, as the proof is somewhat simpler to follow. Notice that part 2 of the theorem implies part 1.

Let $V' = V \setminus \{r\}$. We wish to show that for $d : V \to \mathbb{Q}^+$ such that $d_i \leq 1$, for every $i \in V'$, $d_r = 1$, and

$$d_i \leq \sum_{j\in\Gamma(S)} d_j \quad \forall S \subseteq V',\ \forall i \in S, \tag{8}$$

there exists a packing $\mathcal{T}$ of connected node sets containing r such that $T \in \mathcal{T}$ has weight λ_T in the packing, $\sum_{T\in\mathcal{T}} \lambda_T \leq 1$, and such that for every node $v \in V$, the following property holds:

$$\alpha d_v \leq \sum_{v\in V} \sum_{T\ni v} \lambda_T \leq d(v), \tag{9}$$

where $\alpha = 1/c\log|V|$. The best packing satisfying the above property is given by the solution to the following linear program, which we denote CLP:

$$\begin{gathered}\text{maximize} \qquad \alpha \qquad \text{s.t.} \\ d_i\alpha - \sum_{T\in\mathcal{T}|i\in T} \lambda_T \leq 0 \forall i \in V, \\ \sum_{T\in\mathcal{T}|i\in T} \lambda_T \leq d_i \quad \forall i \in V, \\ \lambda \geq 0,\end{gathered} \tag{10}$$

where $\alpha \in \mathbb{R}$ and $\lambda \in \mathbb{R}^{\mathcal{T}}$.

The dual program, which we denote by CD, is

$$\begin{gathered}\text{minimize} \quad \sum_{i\in V} d_i x_i \quad \text{s.t.} \\ \sum_{i\in V} d_i y_i = 1, \\ \sum_{i\in T} (x_i - y_i) \geq 0 \forall T \in \mathcal{T}, \\ x, y \geq 0,\end{gathered} \tag{11}$$

where $x, y \in \mathbb{R}^V$.

Let β be the approximation guarantee for the prize collecting problem. We have $\beta = \Theta(\log n)$. Consider now a modified program, which we denote by MCD, where the constraints (11) are replaced by the constraints

$$\sum_{i \in T} x_i - \beta \sum_{i \in T} y_i \geq 0 \forall T \in \mathcal{T}. \tag{12}$$

LEMMA 7. *Let Z^*_{CD} and Z^*_{MCD} be the optimal values of CD and MCD, respectively (with the same coefficients d). Then $Z^*_{\mathrm{CD}} \geq Z^*_{\mathrm{MCD}}/\beta$.*

Proof. If (x, y) is a feasible solution to CD, then $(\beta x, y)$ is a feasible solution to MCD. □

Let $\{\lambda_T\}_{T \in \mathcal{T}}, \alpha$ be a feasible solution to CLP. Note that $\sum_{T \in \mathcal{T}} \lambda_T \leq 1$. This follows from the fact that the root r is contained in every tree, and therefore, the inequality above is implied by constraint (10) for r.

Observe that if we could find in polynomial time a solution to CLP of value $\frac{1}{\beta}$, it would induce a packing of sets satisfying the property (9). Indeed, such packing could be obtained by picking sets T, with $\lambda_T > 0$. Unfortunately, the linear program CLP has an exponential number of variables and therefore cannot be solved in polynomial time by a linear programming algorithm. We overcome this problem by solving the modified dual program MCD. This program has an exponential number of constraints, but it can be solved in polynomial time using the ellipsoid algorithm, given a separation oracle. (Each constraint of MCD has a short description: see Grötschel, Lovász, and Schrijver (1993) for the conditions required for applying the ellipsoid.) Next we describe a separation oracle for MCD, which can be applied under the assumption that $\sum_{i \in V} d_i x_i < 1$. We show that such an oracle suffices for computing a solution that suits our purpose.

Formally, under the assumption $\sum_{i \in V} d_i x_i < 1$, we wish to find a set T violating the constraint (12), i.e., a set T such that

$$\sum_{i \in T} x_i - \beta \sum_{i \in T} y_i < 0. \tag{13}$$

The following lemma provides a method for finding such a set.

LEMMA 8. *Let $x, y \in \mathbb{R}^V$ so that $\sum_{i \in V} d_i y_i = 1$ and $\sum_{i \in V} d_i x_i < 1$, where d is a vector satisfying the conditions of Theorem* 1. *Then the set T produced by* **PrizeCollecting** *for the instance of the prize collecting problem with $c(v) = x_v$ and $\pi(v) = y_v$, for all $v \in V'$, satisfies the inequality* (13).

Proof. By Theorem 3, **PrizeCollecting** produces a set T which satisfies:

$$\sum_{i \in T} x_i + \beta \sum_{i \notin T} y_i \leq \beta \textsc{pc-opt}(x, y), \tag{14}$$

where $\textsc{pc-opt}(x, y)$ is an optimal value of PC-LP for the problem instance described above.

Define a feasible solution to PC-LP as follows. Number the vertices of $V \setminus \{r\}$ so that $d_1 \leq d_2 \leq \cdots \leq d_k$, $k = |V| - 1$. Define $S_i = \{1, 2, \ldots, i\}$ for $1 \leq i \leq k$. Set

$$\begin{aligned} x_i &= d_i \qquad \forall i \in V, \\ z_{S_k} &= 1 - d_k, \\ z_{S_i} &= d_{i+1} - d_i, 1 \leq i \leq k, \\ z_S &= 0, \qquad S \neq S_i \forall i \in \{1, \ldots, k\}. \end{aligned}$$

First, observe that this is, indeed, a feasible solution to PC-LP. To see that the constraints (1) are satisfied, consider any set $S \subseteq V \setminus \{r\}$. Let j be a vertex with maximal d_j in S. Then sets $T \subseteq V'$ with nonzero z_T containing S are exactly the sets $S_j, S_{j+1}, \ldots, S_k$. Therefore, we get $\sum_{T \supseteq S} z_T = 1 - d_j$. Then constraint (1) is satisfied for S as d satisfies constraint (8) for S and $j \in S$. It can be easily verified that constraints (2) are also satisfied. Now consider the value of the solution to PC-LP described above. Observe that $\sum_{S \ni i} z_S = 1 - d_i$, $1 \leq i \leq k$. Therefore, each vertex i, $1 \leq i \leq k$, contributes its penalty multiplied by $(1 - d_i)$ to the objective function value. Thus, we get that the value of this solution is $\sum_{i \in V} d_i x_i + \sum_{i \in V} (1 - d_i) y_i$. As this value is not smaller than the optimal value for PC-LP, by (14) we get

$$\begin{aligned}\sum_{i \in T} x_i + \beta \sum_{i \notin T} y_i &\leq \beta \left(\sum_{i \in V} d_i x_i + \sum_{i \in V} (1 - d_i) y_i \right) \\ &= \beta \left(\sum_{i \in V} d_i x_i + \sum_{i \in V} y_i - 1 \right) \\ &< \beta \sum_{i \in V} y_i.\end{aligned}$$

The latter inequality follows by the assumption that $\sum_{i \in V} d_i x_i < 1$. Therefore, it follows that

$$\sum_{i \in T} x_i - \beta \sum_{i \in T} y_i < 0,$$

which implies the lemma. □

We use the separation oracle described in the lemma above on the set of constraints of MCD together with the constraint $\sum_{i \in V} d_i x_i < 1$ until no feasible solution satisfying the latter constraint and the constraints already used by the algorithm can be found. At this point, the optimal value of MCD with the subset of constraints used by the ellipsoid algorithm is at least 1, and therefore the optimal value of CLP with just the variables λ_T corresponding to the used constraints is at least $\frac{1}{\beta}$. Therefore, we can solve CLP with just those variables to obtain the packing of sets satisfying (9).

We now proceed with the proof of part 2 of Theorem 1. The proof structure is essentially the same as the proof of part 1. However, some of the details are more complicated.

Let $V' = V \setminus \{r\}$. We show that for $d : V \to \mathbb{Q}^+$ such that $d_i \leq 1$ for every $i \in V'$, $d_r = 1$, and such that constraints (8) are satisfied, there exists a packing $\mathcal{T}$ of connected node sets containing r such that $T \in \mathcal{T}$ has weight λ_T in the packing, $\sum_{T \in \mathcal{T}} \lambda_T \leq 1$, and such that for every node $v \in V$, the following property holds:

$$d(v) \leq \sum_{v \in V} \sum_{T \ni v} \lambda_T \leq \min\{1, d(v)\alpha\}, \tag{15}$$

where $\alpha = c \log |V|$. The best packing satisfying the above property is given by the solution to the following linear program which we denote CLP':

$$\text{minimize} \qquad \alpha \qquad \text{s.t.}$$

$$\begin{aligned} d_i\alpha - \sum_{T\in\mathcal{T}|i\in T} \lambda_T &\geq 0 \forall i \in V, \\ \sum_{T\in\mathcal{T}|i\in T} \lambda_T &\geq d_i \quad \forall i \in V, \\ -\sum_{T\in\mathcal{T}|i\in T} \lambda_T &\geq -1 \ \forall i \in V. \\ \lambda &\geq 0, \end{aligned} \tag{16}$$

where $\alpha \in \mathbb{R}$ and $\lambda \in \mathbb{R}^{\mathcal{T}}$.

Consider the dual program for CLP', denoted CD':

$$\text{maximize} \qquad \sum_{i\in V}(d_i y_i - z_i) \qquad \text{s.t.}$$

$$\begin{aligned} \sum_{i\in V} d_i x_i &= 1, \\ \sum_{i\in T}(y_i - z_i - x_i) &\geq 0 \forall T \in \mathcal{T}, \\ x, y, z &\geq 0, \end{aligned} \tag{17}$$

where $x, y, z \in \mathbb{R}^V$.

We modify CD' by replacing constraints (17) by:

$$\sum_{i\in T}(\beta(y_i - z_i) - x_i) \leq 0 \quad \forall T \in \mathcal{T}. \tag{18}$$

We denote the modified dual program by MCD'. By the argument, similar to that of the proof of Lemma 7,

$$Z^*_{\mathrm{CD}'} \leq \beta Z^*_{\mathrm{MCD}'}.$$

Next we show a separation oracle for MCD', which can be applied under the assumption $\sum_{i\in V}(d_i y_i - z_i) > 1$. Formally, we wish to find a set T that violates constraint (18). Similarly to the proof of Lemma 8, we apply **PrizeCollecting** to an instance with costs x_i and penalties y_i. Let T be the set produced by the algorithm. By the same reasoning as in the proof of Lemma 8, we get

$$\sum_{i\in T} x_i + \beta \sum_{i\notin T} y_i \leq \beta \left(\sum_{i\in V} d_i x_i + \sum_{i\in V}(1-d_i) y_i \right).$$

Since by our assumption

$$\sum_{i\in V} d_i y_i > 1 + \sum_{i\in V} z_i,$$

we get

$$\sum_{i\in T} x_i + \beta \sum_{i\notin T} y_i < \beta + \beta \sum_{i\in V} y_i - \beta - \beta \sum_{i\in V} z_i,$$

or, equivalently,

$$\sum_{i \in T} x_i - \beta \sum_{i \in T} y_i + \beta \sum_{i \in V} z_i < 0.$$

Since $z_i \geq 0$, for all $i \in V$, it follows that

$$\sum_{i \in T} (x_i + \beta(z_i - y_i)) < 0,$$

implying that the set T violates constraint (18).

We use the separation oracle described above on the set of constraints of MCD' together with the constraint $\sum_{i \in V}(d_i y_i - z_i) > 1$ until no feasible solution satisfying the latter constraint, and the constraints already used by the algorithm, can be found. At this point, the optimal value of MCD' with the subset of constraints used by the ellipsoid algorithm is at most 1, and therefore the optimal value of CLP' with just the variables λ_T corresponding to the used constraints is at most β. Therefore, we can solve CLP' with just those variables to obtain the packing of sets satisfying (15).

4. The quota problem. In the quota problem, given an undirected graph $G = (V, E)$ with a cost function $c : V \to \mathbb{Q}^+$, a profit function $\pi : V \to \mathbb{Q}^+$, a specified root r, and a quota Q, the objective is to find a connected subset T of V containing r such that the total profit of T is at least Q and the total cost of T is minimized.

For every node i, we denote by $L(i)$ the minimum cost of a path connecting r and i. Notice that in dealing with the quota problem we may eliminate nodes i with $\pi(i) \geq Q$. If the optimal solution contains such a node i, then the optimal solution is a least-cost path connecting r and i. We can compute such a path for every node i with $\pi(i) \geq Q$ and compare its cost to the solution produced by the algorithm described below. Furthermore, let Q-OPT denote the cost of the optimal solution for the quota problem. Clearly, if $L(i) >$ Q-OPT, then i is not contained in any optimal solution. We can eliminate nodes i with $L(i) >$ Q-OPT by enumerating over possible values for Q-OPT. The only interesting values for this purpose are the values $L(i)$ for all nodes i (i.e., for $L(i)$ we eliminate all nodes j with $L(j) \geq L(i)$). In the following discussion we assume that V does not contain nodes i with $\pi(i) \geq Q$ or with $L(i) >$ Q-OPT.

Consider the following linear programming relaxation for the quota problem (V' denotes $V \setminus \{r\}$), which we denote by Q-LP:

$$\begin{aligned} \text{minimize} \quad & \sum_{i \in V} c(i) d_i \quad \text{s.t.} \\ & \sum_{i \in V} \pi(i) d_i \geq Q, \\ & d_i \leq \sum_{j \in \Gamma(S)} d_j \ \ \forall S \subseteq V',\ \forall i \in S, \qquad (19) \\ & d_r = 1, \\ & d \geq 0. \end{aligned}$$

Note that constraints (19) imply $d_i \leq 1$, for all $i \in V'$. Further note that Q-LP can be solved using the ellipsoid algorithm (a separation oracle requires computing minimum node cuts).

Let d be a solution to Q-LP. Let Q-LP(d) denote the value of d. Consider the packing of connected sets containing r from part 2 of Theorem 1. Let $\mathcal{T}$ denote the support of the packing, and for every $T \in \mathcal{T}$ let λ_T denote the (positive) weight of T in the packing. The following lemma is a trivial consequence of Theorem 1. We omit the proof.

LEMMA 9. *The packing from part* 2 *of Theorem* 1 *satisfies the following conditions:*

$$\sum_{T \in \mathcal{T}} \lambda_T = 1, \tag{20}$$

$$\sum_{T \in \mathcal{T}} c(T)\lambda_T \leq \beta\text{Q-LP}(d), \tag{21}$$

$$\sum_{T \in \mathcal{T}} \pi(T)\lambda_T \geq Q. \tag{22}$$

Let $\mathcal{L} = \{T \in \mathcal{T} \mid c(T) \leq \beta\text{Q-LP}(d)\}$, $\mathcal{H} = \{T \in \mathcal{T} \mid c(T) > \beta\text{Q-LP}(d)\}$, $\mathcal{C} = \{T \in \mathcal{T} \mid \pi(T) < Q\}$, and $\mathcal{X} = \{T \in \mathcal{T} \mid \pi(T) \geq Q\}$.

LEMMA 10. *If* $\mathcal{L} \bigcap \mathcal{X} = \emptyset$, *then there exist* $T_1 \in \mathcal{L}$ *and* $T_2 \in \mathcal{H}$ *such that* $\pi(T_2 \setminus T_1) \geq Q - \pi(T_1)$ *and*

$$\frac{c(T_2 \setminus T_1)}{\pi(T_2 \setminus T_1)} \leq \frac{\beta\text{Q-LP}(d)}{Q - \pi(T_1)}.$$

Proof. For $\mathcal{A} \subset \mathcal{T}$, let $\lambda_{\mathcal{A}}$ denote $\sum_{T \in \mathcal{A}} \lambda_T$. By Lemma 9, the packing is good, so by property (21), $\mathcal{L}$ is nonempty. Let T_1 be the most profitable set in $\mathcal{L}$. Assume that $\pi(T_1) < Q$. By property (22), $\mathcal{H}$ is nonempty. We may assume that $\mathcal{H} \bigcap \mathcal{C} = \emptyset$, otherwise, remove from $\mathcal{T}$ the sets in $\mathcal{H} \bigcap \mathcal{C}$ and scale the weights of the remaining sets by $1 - \lambda_{\mathcal{H} \bigcap \mathcal{C}}$. As we eliminated sets with above-average cost and below-average profit, the modified packing is still good. Notice that $\pi(T_1)\lambda_{\mathcal{L}} \geq \sum_{T \in \mathcal{L}} \pi(T)\lambda_T$. Therefore, $\pi(T_1)\lambda_{\mathcal{L}} + \sum_{T \in \mathcal{H}} \pi(T)\lambda_T \geq Q$, or, as $\lambda_{\mathcal{L}} = 1 - \lambda_{\mathcal{H}}$,

$$\sum_{T \in \mathcal{H}} \pi(T \setminus T_1)\lambda_T \geq Q - \pi(T_1). \tag{23}$$

Notice that for all $T \in \mathcal{H}$, $\pi(T) \geq Q$ (as $\mathcal{H} \bigcap \mathcal{C} = \emptyset$), so $\pi(T \setminus T_1) \geq Q - \pi(T_1)$. Also, because the packing is good,

$$\sum_{T \in \mathcal{H}} c(T \setminus T_1)\lambda_T \leq \sum_{T \in \mathcal{H}} c(T)\lambda_T \leq \beta\text{Q-LP}(d). \tag{24}$$

It follows from (23) and (24) that there exists $T_2 \in \mathcal{H}$ which, together with T_1, satisfies the claims stipulated by the lemma. □

Our algorithm for the quota problem proceeds as follows. Given an optimal solution d to Q-LP, we use part 2 of Theorem 1 to compute a packing of connected sets containing r. If this packing has a set $T \in \mathcal{L} \bigcap \mathcal{X}$, we output T. Otherwise, we take $T_1 \in \mathcal{L}$ and $T_2 \in \mathcal{H}$ as exist by Lemma 10 and proceed as follows. Consider the graph induced by $T_2 \setminus T_1$. This graph is a collection of components, each of which is connected to some vertex in T_1. We take a spanning tree in each component, rooted at a vertex adjacent to a vertex of T_1. Denote the set of these spanning trees by $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$.

In what follows we describe a trimming procedure that we apply to the set $\mathcal{S}$. We observe that our trimming procedure bears some similarity to that used in Garg's 5-approximation for k-MST. Our procedure outputs a trimmed set of trees $\mathcal{R} =$

$\{R_1, R_2, \ldots, R_l\}$ whose roots are taken from the set of roots of the trees in the original set. The trimmed set has the property that its total profit is at least $Q-\pi(T_1)$, and its total cost is at most $(2\beta+1)$Q-OPT. We then connect $R_1, R_2, \ldots, R_l$ to T_1. (Notice that the roots of the trimmed trees are adjacent to vertices of T_1.) Let T be the resulting tree. By the previous discussion,

$$c(T) \leq \Theta(\log n)\text{Q-OPT},$$
$$\pi(T) \geq Q.$$

Our algorithm outputs T.

We proceed to describe the trimming procedure. Let $p = Q - \pi(T_1)$. Notice that the total profit of the trees in $\mathcal{S}$ is at least p (because $\pi(T_2) \geq Q$). We repeatedly remove a maximal rooted subtree of any tree in $\mathcal{S}$ (including an entire tree), whose removal leaves the cost-to-profit ratio at most $\rho = \frac{\beta\text{Q-OPT}}{p}$ and the profit at least p until no such subtree can be found. Let $\overline{\mathcal{S}}$ denote the set of remaining trees. If the profit of $\overline{\mathcal{S}}$ is at most $2p$, then its cost must be at most 2βQ-OPT. In this case we output $\mathcal{R} = \overline{\mathcal{S}}$.

Otherwise, the profit of $\overline{\mathcal{S}}$ is more than $2p$. We consider two cases.

Case 1. All rooted subtrees of trees in $\overline{\mathcal{S}}$ have a cost-to-profit ratio of at most ρ. We find a subtree T' rooted at some vertex r' such that the profit of T' is at least p, but the profit of each subtree rooted at a child of r' is less than p. As the total profit of trees in $\overline{\mathcal{S}}$ is greater than $2p$, T' exists and can be found by a simple scanning procedure. Consider the subtrees rooted at the children of r'. We repeatedly remove such a subtree until the total profit of the remaining tree T'' (including r') is between p and $2p$. Notice that, by the assumption in this case, each remaining subtree has a cost-to-profit ratio at most ρ. The total profit of all remaining subtrees is at most $2p$, so their total cost (excluding r') is at most $2p\frac{\beta\text{Q-OPT}}{p} = 2\beta$Q-OPT. We connect T'' to the root of the tree in $\overline{\mathcal{S}}$ containing T'' using a least-cost path in G between the two vertices. As the cost of the path is at most Q-OPT (this includes $c(r')$), the total cost of the resulting tree is at most $(2\beta+1)$Q-OPT. We output the singleton set containing this tree.

Case 2. There exists a rooted subtree T' of a tree in $\overline{\mathcal{S}}$ with a cost-to-profit ratio larger than ρ. Consider a minimal T', inclusionwise. Notice that the profit of the rest of $\overline{\mathcal{S}}$ is less than p (otherwise we would delete T'), and the cost-to-profit ratio is less than ρ. Therefore, the cost of the rest of $\overline{\mathcal{S}}$ is less than βQ-OPT. Moreover, as the total profit of $\overline{\mathcal{S}}$ is more than $2p$, the profit of T' is more than p. If T' is a single vertex r' (a leaf), then, as $c(r') \leq$ Q-OPT, $\overline{\mathcal{S}}$ has a profit of more than $2p$ and a cost of less than $(\beta+1)$Q-OPT, so we output $\mathcal{R} = \overline{\mathcal{S}}$. Otherwise, let r' be the root of T'. By the minimality assumption, every rooted subtree of T' has a cost-to-profit ratio of at most ρ. If there exists a subtree rooted at a child of r' with a profit of at least p, we can apply the argument of Case 1 to this tree. Otherwise, the profit of each such subtree is less than p. We remove subtrees until the total profit (including r') is between p and $2p$. The remaining subtrees have a total cost of at most βQ-OPT. We add the least-cost path in G from the root of the tree in $\overline{\mathcal{S}}$ containing T' to r'. The added cost is at most Q-OPT. We output the singleton set containing the resulting tree.

5. The budget problem. In the budget problem, given an undirected graph $G = (V, E)$ with a cost function $c : V \to \mathbb{Q}^+$, a profit function $\pi : V \to \mathbb{Q}^+$, a specified root r, and a budget B, the objective is to find a subtree T of G containing r such that the total cost of T does not exceed B and the total profit of T is maximized. We may assume that, for every vertex $i \in V$, $L(i) \leq B$; otherwise, no feasible solution can include i, so we can discard it. Let B-OPT denote the value of the optimal solution to the budget problem.

Consider the following linear programming relaxation to the budget problem (V' denotes $V \setminus \{r\}$):

$$\begin{aligned} \text{maximize} \quad & \sum_{i \in V} \pi(i) d_i \quad \text{s.t.} \\ & \sum_{i \in V} c(i) d_i \leq B, \\ & d_i \leq \sum_{j \in \Gamma S} d_j \quad \forall S \subseteq V',\ \forall i \in S, \\ & d_r = 1, \\ & d \geq 0, \end{aligned} \tag{25}$$

where $d \in \mathbb{R}^V$. We denote this linear program by B-LP. Note that B-LP can be solved in polynomial time using the ellipsoid algorithm.

Let d be a feasible solution to B-LP. Let B-LP(d) denote the value of d. Consider the packing of connected sets containing r from part 1 of Theorem 1. Let $\mathcal{T}$ denote the support of the packing, and for every $T \in \mathcal{T}$ let λ_T denote the weight of T in the packing. It is easy to verify that this packing satisfies the following properties:

$$\sum_{T \in \mathcal{T}} \lambda_T \leq 1, \tag{26}$$

$$\sum_{v \in V} c(v) \sum_{T \ni v} \lambda_T \leq B, \tag{27}$$

$$\sum_{v \in V} \pi(v) \sum_{T \ni v} \lambda_T \geq \Theta\left(\frac{1}{\log n}\right) \text{B-OPT}. \tag{28}$$

Next we show how the packing $\mathcal{T}$ can be used to derive an $O(\log n)$-approximation for the budget problem. Let $\mathcal{L} = \{T \in \mathcal{T} \mid c(T) \leq B\}$, and $\mathcal{H} = \{T \in \mathcal{T} \mid c(T) > B\}$.

LEMMA 11. *At least one of the following conditions holds:*

1. $\exists T \in \mathcal{L}$ *such that* $\pi(T) \geq \Theta(\frac{1}{\log n})\text{B-OPT}$*;*
2. $\exists T \in \mathcal{H}$ *such that* $\frac{\pi(T)}{c(T)} \geq \Theta(\frac{1}{\log n}) \frac{\text{B-OPT}}{B}$.

Proof. Denote

$$\pi(\mathcal{T}) = \sum_{v \in V} \pi(v) \sum_{T \ni v} \lambda_T.$$

First, consider the case when the profit from $\mathcal{L}$ is at least half of the total profit achieved by the packing. Formally, assume

$$\sum_{v \in V} \pi(v) \sum_{T \ni v | T \in \mathcal{L}} \lambda_T \geq \frac{1}{2} \pi(\mathcal{T}).$$

In this case there exists a set $T' \in \mathcal{L}$ such that

$$\sum_{v \in T'} \pi(v) \geq \frac{1}{2} \pi(\mathcal{T}).$$

Indeed, assume this is not the case. Then

$$\begin{aligned}\sum_{v\in V}\pi(v)\sum_{T\ni v|T\in\mathcal{L}}\lambda_T &= \sum_{T\in\mathcal{L}}\lambda_T\sum_{v\in T}\pi(v)\\ &< \frac{1}{2}\pi(\mathcal{T})\sum_{T\in\mathcal{L}}\lambda_T\\ &\le \frac{1}{2}\pi(\mathcal{T}),\end{aligned}$$

contradicting our assumption. As $\pi(\mathcal{T}) = \Theta(\frac{1}{\log n})$B-OPT, the first condition of the lemma holds.

Now consider the other case when

$$\sum_{v\in V}\pi(v)\sum_{T\ni v|T\in\mathcal{L}}\lambda_T < \frac{1}{2}\pi(\mathcal{T}).$$

Then

$$\sum_{v\in V}\pi(v)\sum_{T\ni v|T\in\mathcal{H}}\lambda_T \ge \frac{1}{2}\pi(\mathcal{T}).$$

Moreover, by property (27),

$$\sum_{v\in V}c(v)\sum_{T\ni v}\lambda_T \le B,$$

and, in particular,

$$\sum_{v\in V}c(v)\sum_{T\ni v|T\in\mathcal{H}}\lambda_T \le B.$$

Therefore, we get

$$\frac{\sum_{v\in V}\pi(v)\sum_{T\ni v|T\in\mathcal{H}}\lambda_T}{\sum_{v\in V}c(v)\sum_{T\ni v|T\in\mathcal{H}}\lambda_T} \ge \frac{1}{2}\frac{\pi(\mathcal{T})}{B}.$$

We conclude that there exists a set $T' \in \mathcal{H}$ such that

$$\frac{\sum_{v\in T'}\pi(v)}{\sum_{v\in T'}c(v)} \ge \frac{1}{2}\frac{\pi(\mathcal{T})}{B}.$$

Indeed, otherwise we would have

$$\begin{aligned}\frac{\sum_{v\in V}\pi(v)\sum_{T\ni v|T\in\mathcal{H}}\lambda_T}{\sum_{v\in V}c(v)\sum_{T\ni v|T\in\mathcal{H}}\lambda_T} &= \frac{\sum_{T\in\mathcal{H}}\lambda_T\sum_{v\in T}\pi(v)}{\sum_{T\in\mathcal{H}}\lambda_T\sum_{v\in T}c(v)}\\ &< \frac{\sum_{T\in\mathcal{H}}\lambda_T\sum_{v\in T}c(v)\cdot\frac{1}{2}\frac{\pi(\mathcal{T})}{B}}{\sum_{T\in\mathcal{H}}\lambda_T\sum_{v\in T}c(v)}\\ &= \frac{1}{2}\frac{\pi(\mathcal{T})}{B}.\end{aligned}$$

By property (28), we get that the second condition of the lemma holds. □

To obtain an $O(\log n)$-approximation for budget problem, we proceed as follows. If $\mathcal{T}$ satisfies condition 1 of Lemma 11, our algorithm outputs the set T' for which the condition holds. Otherwise, the algorithm proceeds with the set T' for which condition 2 of Lemma 11 holds. Clearly, the total profit of T' is at least $\Theta(\frac{1}{\log n})$B-OPT, but it is not a feasible solution for the budget problem, as its total cost exceeds B. Our algorithm trims T' to obtain another set T'' containing r such that

$$\frac{\sum_{v\in T''} \pi(v)}{\sum_{v\in T''} c(v)} \geq \Theta\left(\frac{1}{\log n}\right)\frac{\text{B-OPT}}{B},$$

and

$$\frac{B}{2} \leq \sum_{v\in \tilde{T}} c(v) \leq 2B.$$

A trimming procedure to obtain such a set appears in Guha et al. (1999).

By the discussion above, we conclude

THEOREM 12. *The above algorithm is an $O(\log n)$-approximation for the budget problem.*

6. Concluding remarks.

The packing theorems in this paper point to an interesting interplay between the primal-dual schema and rounding of linear programming relaxations. Indeed, nonconstructive versions of these packing theorems and similar theorems can be deduced directly from the bounds on the dual solution cut packings underlying the related primal-dual algorithms. A better understanding of this issue is desired and might lead to improved algorithms or new applications.

Our results for the budget problem are unsatisfactory. The problem is not known to be harder to approximate than the maximum coverage problem, for which a tight $1 - 1/e$ bound on the approximability is known. Moreover, there is no reason to believe that the problem cannot be approximated without violating the strict budget constraints. We conjecture that there is a polynomial time constant-approximation algorithm for this problem.

Acknowledgment. The second author thanks Mike Saks and David Johnson for stimulating discussions.

REFERENCES

S. Arora and G. Karakostas (2006), *A $2+\epsilon$ approximation algorithm for the k-MST problem*, Math. Program., 107, pp. 491–504.

S. Arora and C. Lund (1997), *Hardness of approximations*, in Approximation Algorithms for NP-Hard Problems, PWS Publishing, Boston, MA.

B. Awerbuch, Y. Azar, A. Blum, and S. Vempala (1998), *New approximation guarantees for minimum-weight k-trees and prize-collecting salesmen*, SIAM J. Comput., 28, pp. 254–262.

A. Blum, R. Ravi, and S. Vempala (1999), *A constant factor approximation for the k-MST problem*, J. Comput. System Sci., 58, pp. 101–108.

R. Carr and S. Vempala (2002), *Randomized metarounding*, Random Structures Algorithms, 20, pp. 343–352.

M. Charikar and S. Guha (2005), *Improved combinatorial algorithms for facility location problems*, SIAM J. Comput., 34, pp. 803–824.

F. Chudak, T. Roughgarden, and D. P. Williamson (2004), *Approximate k-MSTs and k-Steiner trees via the primal-dual method and Lagrangian relaxation*, Math. Program., 100, pp. 411–421.

R. Diestel, (2000), *Graph Theory*, 2nd ed., Graduate Texts in Mathematics 173, Springer-Verlag, Berlin.

N. Garg (1996), *A 3-approximation for the minimum tree spanning k vertices*, in Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science, pp. 302–309.

N. Garg (2005), *Saving an epsilon: A 2-approximation for the k-MST problem in graphs*, in Proceedings of the 37th Annual ACM Symposium on Theory of Computing, pp. 396–402.

M. X. Goemans and D. P. Williamson (1995), *A general approximation technique for constrained forest problems*, SIAM J. Comput., 24, pp. 296–317.

M. Grötschel, L. Lovász, and A. Schrijver (1993), *Geometric Algorithms and Combinatorial Optimization*, 2nd corrected ed., Springer-Verlag, Berlin.

S. Guha, A. Moss, J. Naor, and B. Schieber (1999), *Efficient recovery from power outage*, in Proceedings of the 31st Annual ACM Symposium on Theory of Computing, pp. 574–582.

K. Jain and V. V. Vazirani (2001), *Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation*, J. ACM, 48, pp. 274–296.

D. S. Johnson, M. Minkoff, and S. Phillips (2000), *The prize collecting Steiner tree problem: Theory and practice*, in Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms.

S. Khuller, A. Moss, and S. Naor (1998), *The budgeted maximum coverage problem*, Inform. Process. Lett., 70, pp. 39–45.

P. Klein and R. Ravi (1995), *A nearly best-possible approximation algorithm for node-weighted Steiner trees*, J. Algorithms, 19, pp. 104–115.

A. Moss (2001), *High Profit for Low Cost: Approximation Algorithms in Node-Weighted Graphs*, Ph.D. thesis, Technion, Haifa, Israel.

SIAM J. Comput.
Vol. 37, No. 2, pp. 482–501

TESTING VERSUS ESTIMATION OF GRAPH PROPERTIES*

ELDAR FISCHER[†] AND ILAN NEWMAN[‡]

Abstract. Tolerant testing is an emerging topic in the field of property testing, which was defined in [M. Parnas, D. Ron, and R. Rubinfeld, *J. Comput. System Sci.*, 72 (2006), pp. 1012–1042] and has recently become a very active topic of research. In the general setting, there exist properties that are testable but are not tolerantly testable [E. Fischer and L. Fortnow, *Proceedings of the* 20*th IEEE Conference on Computational Complexity*, 2005, pp. 135–140]. On the other hand, we show here that in the setting of the dense graph model, all testable properties are not only tolerantly testable (which was already implicitly proved in [N. Alon, E. Fischer, M. Krivelevich, and M. Szegedy, *Combinatorica*, 20 (2000), pp. 451–476] and [O. Goldreich and L. Trevisan, *Random Structures Algorithms*, 23 (2003), pp. 23–57]), but also admit a constant query size algorithm that estimates the distance from the property up to any fixed additive constant. In the course of the proof we develop a framework for extending Szemerédi's regularity lemma, both as a prerequisite for formulating what kind of information about the input graph will provide us with the correct estimation, and as the means for efficiently gathering this information. In particular, we construct a probabilistic algorithm that finds the parameters of a regular partition of an input graph using a constant number of queries, and an algorithm to find a regular partition of a graph using a TC_0 circuit. This, in some ways, strengthens the results of [N. Alon, R. A. Duke, H. Lefmann, V. Rödl, and R. Yuster, *J. Algorithms*, 16 (1994), pp. 80–109].

Key words. property testing, graph algorithms, approximation, regularity lemma

AMS subject classifications. 05C85, 05C35, 68Q10

DOI. 10.1137/060652324

1. Introduction.

Combinatorial property testing deals with the following task: For a fixed $\epsilon > 0$ and a fixed property $\mathcal{P}$, distinguish using as few queries as possible (and with probability at least $\frac{2}{3}$) between the case that an input of size m satisfies $\mathcal{P}$, and the case that the input is ϵ-far from satisfying $\mathcal{P}$. In our context the inputs are boolean, and the distance from $\mathcal{P}$ is measured by the minimum number of bits that have to be modified in the input in order to make it satisfy $\mathcal{P}$, divided by the input size m. For the purpose here we are mainly interested in tests that have a number of queries that depends only on the approximation parameter ϵ and is independent of the input size. Properties that admit such algorithms are called *testable*.

The first time a question formulated in terms of property testing was considered was by Blum, Luby, and Rubinfeld [7], and the general notion of property testing was first formally defined by Rubinfeld and Sudan [17]. The first investigation in the combinatorial context was that of Goldreich, Goldwasser and Ron [13], where the testing of combinatorial graph properties (in the "dense" graph model) was first formalized; their framework will also be the one used here. In recent years the field of property testing has enjoyed rapid growth, as witnessed in the surveys [16] and [10].

One of the main goals in the study of graph property testing is the finding of structural characterization results or, failing that, results that identify large classes of

*Received by the editors February 17, 2006; accepted for publication (in revised form) October 25, 2006; published electronically May 22, 2007. This research was supported in part by an Israel Science Foundation grant number 55/03. A preliminary version of this paper appeared in *Proceedings of the* 37*th ACM STOC* (2005) [4].

http://www.siam.org/journals/sicomp/37-2/65232.html

[†]Faculty of Computer Science, Technion–Israel Institute of Technology, Technion City, Haifa 32000, Israel (eldar@cs.technion.ac.il).

[‡]Department of Computer Science, Haifa University, Haifa 31905, Israel (ilan@cs.haifa.ac.il).

properties that are testable. An example of such a large class is the class of partition properties that was identified in [13]. Other classes were identified as testable using the regularity lemma of Szemerédi in [2] and [9]. The regularity lemma is a very useful tool that guarantees the existence of a sort of "short summary" for graphs with any number of vertices. The price is that the involved constants will not have a practical bound, but for theoretical results this lemma currently is the most powerful tool for understanding the essence of graph property testing.

The question of providing a complete structural characterization result for the testable graph properties has been one of the central themes in the research on testing graph properties. A partial result that in some sense characterizes graph properties that consist of only one graph according to their testability is found in [11], also making use of the regularity lemma. Concerning 1-sided graph property testing, where the algorithm is also required to be independent of the number of vertices of the graph, a recent work of Alon and Shapira [5] approaches what is in essence a full characterization. Very recently a complete characterization of the properties that are testable by 2-sided error tests was provided in [3]. In a different angle of the characterization problem, the canonical testers of [14] can be considered as a first hint that testable graph properties are more than just testable. Here we investigate this further, showing that the class of all testable graph properties (with 1-sided or 2-sided algorithms) is in fact identical to a class of properties that admit algorithms with much stringer requirements than those of property testers.

An investigation that goes beyond the original definition of testable properties was initiated by Parnas, Ron, and Rubinfeld [15] and concerns tolerant testers. These are property testers that reject all instances that are far enough from the property $\mathcal{P}$ and accept every instance that is close enough to $\mathcal{P}$ (and not just instances that are in $\mathcal{P}$). Recently, Fischer and Fortnow [12] showed that not all testable properties are also tolerantly testable. Here we prove a positive general result on testable graph properties that involves a much tighter concept. We say that a property is (ϵ, δ)-*estimable* if there exists a probabilistic algorithm making a constant number of queries on any input (independently of the input size) that distinguishes with probability $\frac{2}{3}$ between the case that the input is $(\epsilon-\delta)$-close to some input that satisfies the property, and the case that it is ϵ-far from any input satisfying the property. We call a property *estimable* if it is (ϵ, δ)-estimable for every fixed $\epsilon > 0$ and $\delta > 0$. Thus, if a property is estimable, then there exists an $O(1)$-query algorithm that can estimate the relative distance of an input from the property within any fixed additive constant. Obviously estimability (and also tolerant testing, where we demand only an (ϵ, δ)-estimation for some $\delta > 0$ that may depend on ϵ) is a generalization of the standard testing, and the two notions coincide when we take $\delta = \epsilon$.

Our main result is a proof that all testable graph properties are also estimable. Equivalently, we obtain that for every testable property $\mathcal{P}$ and every $\epsilon > 0$, the property of being ϵ-close to $\mathcal{P}$ is in itself testable. For nongraph properties this is not always true, as shown in [12].

While the famed regularity lemma of Szemerédi is not very applicable in practice, it is quite important theoretically and not only for property testing. Alon et al. [1] have shown that a regular partition can be found in asymptotically the same time complexity as that of matrix multiplication. For many applications of the regularity lemma, one does not need to know the regular partition itself, but only its signature (the pairwise edge densities between its sets). A lemma towards our main result asserts the existence of a randomized algorithm that uses only $O(1)$ queries to the input and approximates the signature of an ϵ-regular partition of a graph to within an additive error of ϵ for any fixed constant $\epsilon > 0$.

As it turns out, this proof also implies a new algorithm that allows the finding of a regular partition using a very low complexity class algorithm (namely TC_0, as opposed to NC_1 which was previously known from [1]).

The rest of the paper is organized as follows. Section 2 contains the most basic definitions and the formal statement of the main result. Section 3 contains definitions and lemmas concerning Szemerédi's regularity lemma, the essence of property testing algorithms in the dense graph model, and the connection between them. Section 4 contains a framework for extending Szemerédi's regularity lemma, leading to the proof of the main result. This proof is based on two main lemmas: One lemma states that knowing the parameters of a certain partition of the graph (which is guaranteed to exist by the extension of Szemerédi's lemma) is enough for knowing how far the input graph is from a graph which a property testing algorithm would accept, and the other lemma states that an approximation of the parameters of such a partition can indeed be calculated with high probability from a small sample of the graph. These two main lemmas are then proved in section 5 (approximating a partition) and section 6 (estimating the distance from the property). Finally, section 7 contains some concluding comments, including a description of the low complexity algorithm for finding an ϵ-regular partition of a graph.

2. The main result. In the following we formally state our main result. We start with the most basic definition of property testing of graphs (in the "dense model" context).

DEFINITION 1. *We say that two graphs G and G' with the same vertex set of size n are* ϵ-close *if the number of vertex pairs that form an edge for one of G and G' but not the other does not exceed $\epsilon\binom{n}{2}$. For a property $\mathcal{P}$ of graphs, we say that G is* ϵ-close to $\mathcal{P}$ *if there exists a graph G' that satisfies $\mathcal{P}$ and is ϵ-close to G. If there exists no such G', then we say that G is* ϵ-far from $\mathcal{P}$. *For properties of combinatorial objects other than graphs, we replace "$\binom{n}{2}$" in the definition above with the size of the corresponding input.*

We call a property ϵ-testable *if there exists a probabilistic algorithm making a constant number of queries on any input (independently of the input size, which is given to the algorithm in advance) that distinguishes with probability $\frac{2}{3}$ between the case that the input satisfies the property, and the case that the input is ϵ-far from any input that satisfies the property. We call a property* testable *if it is ϵ-testable for every fixed constant $\epsilon > 0$.*

Parnas, Ron, and Rubinfeld [15] have started investigating properties (of various combinatorial objects and not just graphs) for which there exists a probabilistic algorithm that apart from being an ϵ-test is also guaranteed to accept (with probability at least $\frac{2}{3}$) any input that is sufficiently close to satisfying the property. In the following we concern ourselves with the strictest possible definition of such properties, in that we want to also accept any input whose distance from the property is only somewhat smaller than the guaranteed rejection distance.

DEFINITION 2. *We call a property* (ϵ, δ)-estimable *if there exists a probabilistic algorithm making a constant number of queries on any input (independently of the input size) that distinguishes with probability $\frac{2}{3}$ between the case that the input is $(\epsilon-\delta)$-close to some input that satisfies the property, and the case that it is ϵ-far from any input satisfying the property. We call a property* estimable *if it is (ϵ, δ)-estimable for every fixed $\epsilon > 0$ and $\delta > 0$.*

We prove that for graph properties (in the dense model), estimation algorithms exist for any property for which there exists a test in the usual sense.

Theorem 2.1. *Every testable property of graphs is also estimable.*

As a corollary from the proofs, we also find an algorithm for constructing an ϵ-regular partition (or a strengthening thereof) of an input graph G using a low complexity (TC_0) algorithm. As the required definitions for stating this result and its motivations are only presented in sections 3 and 4, the result is discussed in full in section 7.

3. The building blocks.

In this section we prepare some tools that are needed for the following discussion. We define and explain the role of regular partitions and show their relevance to predicting the behavior of a given testing algorithm when applied to the input graph.

Starting with this section and throughout the paper, we use the convention that a function defined by the statement of a lemma is indexed with the lemma's number. We make no attempt anywhere to minimize the constants involved and ignore floor and ceiling signs when these make no essential difference for the argument.

For some of the proofs we use the following standard Chernoff-type large deviation inequality (see, e.g., [6, Appendix A]).

Lemma 3.1. *Suppose that $X_1, \dots, X_m$ are m independent boolean random variables, satisfying $\Pr(X_i = 1) = p_i$. Let $E = \sum_{i=1}^m p_i$. Then, $\Pr(|\sum_{i=1}^m X_i - E| \geq \delta m) \leq 2e^{-2\delta^2 m}$.*

In the following we often use one distribution to approximate another. For this the following definition is handy.

Definition 3. *Given two distributions μ and ν over a finite family $\mathcal{H}$ of combinatorial structures, their* variation distance *is defined as $|\mu - \nu| = \frac{1}{2}\sum_{H\in\mathcal{H}} |\Pr_\mu(H) - \Pr_\nu(H)|$.*

We note that the variation distance is just a normalized distance in ℓ_1. In particular $0 \leq |\mu - \nu| \leq 1$ for any ν, μ. The importance of this measure is that if $|\mu - \nu|$ is small, then ν approximates μ well, as asserted by the following well-known lemma for which we provide a proof for completeness.

Lemma 3.2. *If two distributions μ, ν over a finite family $\mathcal{H}$ of combinatorial structures satisfy $|\mu - \nu| \leq \delta$, then for any set $\mathcal{A} \subseteq \mathcal{H}$ we have $|\Pr_\mu(\mathcal{A}) - \Pr_\nu(\mathcal{A})| \leq \delta$.*

Proof. Set $\mathcal{B} = \mathcal{H} \setminus \mathcal{A}$. Because these are probability spaces we have $\Pr_\mu(\mathcal{B}) - \Pr_\nu(\mathcal{B}) = \Pr_\nu(\mathcal{A}) - \Pr_\mu(\mathcal{A})$. Therefore,

$$\begin{aligned}&|\Pr_\mu(\mathcal{A}) - \Pr_\nu(\mathcal{A})|\\ &\quad = \frac{1}{2}|\Pr_\mu(\mathcal{A}) - \Pr_\nu(\mathcal{A})| + \frac{1}{2}|\Pr_\mu(\mathcal{B}) - \Pr_\nu(\mathcal{B})|\\ &\quad \leq \frac{1}{2}\sum_{H\in\mathcal{A}} |\Pr_\mu(H) - \Pr_\nu(H)| + \frac{1}{2}\sum_{H\in\mathcal{B}} |\Pr_\mu(H) - \Pr_\nu(H)| = |\mu - \nu| \qquad \square\end{aligned}$$

The following is also a well-known probabilistic lemma that we will use.

Lemma 3.3. *Let μ be a product distribution over $\{0,1\}^k$, where for $(\alpha_1, \dots, \alpha_k) \in \{0,1\}^k$ we have $\Pr_\mu((\alpha_1, \dots, \alpha_k)) = \prod_{i=1}^k (p_i)^{\alpha_i}(1-p_i)^{1-\alpha_i}$ for a fixed sequence $p_1, \dots, p_k$. Similarly, let ν be a product distribution over $\{0,1\}^k$, with $q_1, \dots, q_k$ replacing $p_1, \dots, p_k$ in the definition above. Then, $|\mu - \nu| \leq \sum_{i=1}^k |p_i - q_i|$.*

Proof. The proof is by induction on k. For $k = 1$ this is immediate from the definition, and for $k > 1$ we use the definition of the variation distance to reduce it to the expression for $k-1$, using extensively the simple inequality $|ab-cd| \leq |a-c|b+c|b-d|$ for $a,b,c,d \geq 0$:

$$\begin{aligned}
&|\mu-\nu| \\
&\quad = \frac{1}{2}\left(\sum_{\alpha_1,\dots,\alpha_{k-1}} \left| p_k \prod_{i=1}^{k-1} (p_i)^{\alpha_i}(1-p_i)^{1-\alpha_i} - q_k \prod_{i=1}^{k-1} (q_i)^{\alpha_i}(1-q_i)^{1-\alpha_i} \right| \right. \\
&\qquad \left. + \sum_{\alpha_1,\dots,\alpha_{k-1}} \left| (1-p_k) \prod_{i=1}^{k-1} (p_i)^{\alpha_i}(1-p_i)^{1-\alpha_i} - (1-q_k) \prod_{i=1}^{k-1} (q_i)^{\alpha_i}(1-q_i)^{1-\alpha_i} \right| \right) \\
&\quad \leq \frac{1}{2} \sum_{\alpha_1,\dots,\alpha_{k-1}} \left(|p_k - q_k| \prod_{i=1}^{k-1} (p_i)^{\alpha_i}(1-p_i)^{1-\alpha_i} \right. \\
&\qquad + q_k \left| \prod_{i=1}^{k-1} (p_i)^{\alpha_i}(1-p_i)^{1-\alpha_i} - \prod_{i=1}^{k-1} (q_i)^{\alpha_i}(1-q_i)^{1-\alpha_i} \right| \\
&\qquad + |(1-p_k) - (1-q_k)| \prod_{i=1}^{k-1} (p_i)^{\alpha_i}(1-p_i)^{1-\alpha_i} \\
&\qquad \left. + (1-q_k) \left| \prod_{i=1}^{k-1} (p_i)^{\alpha_i}(1-p_i)^{1-\alpha_i} - \prod_{i=1}^{k-1} (q_i)^{\alpha_i}(1-q_i)^{1-\alpha_i} \right| \right) \\
&\quad = |p_k - q_k| + \frac{1}{2} \sum_{\alpha_1,\dots,\alpha_{k-1}} \left| \prod_{i=1}^{k-1} (p_i)^{\alpha_i}(1-p_i)^{1-\alpha_i} - \prod_{i=1}^{k-1} (q_i)^{\alpha_i}(1-q_i)^{1-\alpha_i} \right| \\
&\quad \leq \sum_{i=1}^{k} |p_i - q_i|. \qquad \square
\end{aligned}$$

An immediate corollary of Lemma 3.3 is the following lemma (by taking $k = \binom{q}{2}$).

Lemma 3.4. *Suppose that μ and ν are two probability distributions over graphs with the set of vertices $\{v_1,\dots,v_q\}$, where each pair v_iv_j is independently chosen to be an edge with probability $\mu_{i,j}$ and $\nu_{i,j}$, respectively. If $|\mu_{i,j}-\nu_{i,j}| \leq \epsilon/\binom{q}{2}$ for every $1 \leq i < j \leq q$, then the variation distance between μ and ν is bounded by ϵ.*

A crucial notion to the following arguments (as is the case with many other graph property testing results) is Szemerédi's notion of regularity.

Definition 4. *For two nonempty disjoint vertex sets U and V of a graph G, we define the* density *$d(U,V)$ of the pair to be the number of edges of G between U and V, divided by $|U||V|$.*

We say that the pair U,V is ϵ-regular *if for any two subsets U' of U and V' of V, satisfying $|U'| \geq \epsilon|U|$ and $|V'| \geq \epsilon|V|$, the edge densities satisfy $|d(U',V')-d(U,V)| < \epsilon$.*

Although Definition 4 bounds the deviation in densities for any two subsets U', V' that are at least as large as their respective thresholds, it is easy to see that it is enough to require that $|d(U',V')-d(U,V)| \leq \epsilon$ for every two subsets U', V' of size exactly $|U'| = \lceil \epsilon|U| \rceil$ and $|V'| = \lceil \epsilon|V| \rceil$.

Regular pairs behave much like random graphs, as seen in the following well-known lemma.

LEMMA 3.5 (see, e.g., [11, Lemma 4.2] for a proof). *For every k and ϵ there exists $\gamma = \gamma_{3.5}(k, \epsilon)$, so that if $U_1, \dots, U_k$ are disjoint sets of vertices of G such that every two sets form a γ-regular pair, then the following two distributions for picking a graph H with vertices $v_1, \dots, v_k$ have variation distance at most ϵ between them.*

(i) *For every $1 \le i < j \le k$, independently take v_iv_j to be an edge of H with probability $d(V_i, V_j)$.*

(ii) *Pick uniformly and independently a vertex $u_i \in U_i$ for every i, and let v_iv_j be an edge of H if and only if u_iu_j is an edge of G.*

What we need is a "cover" of an entire graph with regular pairs. This idea is formalized in the following.

DEFINITION 5. *Given a graph G, an* equipartition *$\mathcal{A} = \{V_1, \dots, V_k\}$ of G is a partition of its vertex set for which the sizes of any two sets differ by at most* 1*. An equipartition $\mathcal{B} = \{W_1, \dots, W_l\}$ is said to be a* refinement *of $\mathcal{A}$ if all of the sets W_i are each fully contained in some set of $\mathcal{A}$.*

An equipartition $\mathcal{B}$ as above is called ϵ-regular *if at least $(1-\epsilon)\binom{l}{2}$ of the pairs W_i, W_j are ϵ-regular.*

Regular partitions are found using the famed regularity lemma of Szemerédi [18] (see [8, Chapter 7] for a good exposition of the proof).

LEMMA 3.6 (Szemerédi's regularity lemma [18]). *For every k and ϵ there exists $T = T_{3.6}(k, \epsilon)$ such that for every equipartition $\mathcal{A}$ of a graph G with $n \ge N_{3.6}(k, \epsilon)$ vertices into k sets, there exists a refinement $\mathcal{B}$ of $\mathcal{A}$ into $t \le T$ sets which is ϵ-regular.*

We now turn to the behavior of property testers when applied to an input graph G. The most important feature of G would be the count of its small induced subgraphs of any kind, as exemplified in the following definition.

DEFINITION 6. *The* q-statistic *of a graph G is the following probability space over (labeled) graphs with q vertices: Given a labeled graph H with the vertex set $\{v_1, \dots, v_q\}$, the probability for H is exactly the probability that the edge relation of G, when restricted to a uniformly random sequence of q vertices (without repetitions) $w_1, \dots, w_k$, is identical to that of H where each w_i plays the role of v_i. Namely, the q-statistic is just the probability distribution over all (labeled) graphs with q vertices that results from picking at random q distinct vertices of G and considering the induced subgraph.*

Given a family $\mathcal{H}$ of graphs with q vertices, we denote the probability for obtaining a member of $\mathcal{H}$ when drawing a graph according to the q-statistic of G by $\Pr_G(\mathcal{H})$.

Note that in the definition above one could work with isomorphic copies of H rather than labeled graphs. This, however, brings in the extra complication of having to take into account the automorphism group size of H. When dealing with labeled graphs the analysis is simpler.

The importance of knowing the q-statistic of a graph G is in its close connection with the distance of G from a given testable property, proven in [14].

LEMMA 3.7 (canonical testers [14]). *If there is an ϵ-test for a graph property $\mathcal{P}$ that makes a constant number of queries, then there exists such a test that makes its queries by choosing uniformly q distinct vertices of G (for an appropriate constant q) and querying the induced subgraph. In particular, there exists an appropriate family $\mathcal{H}$ of labeled graphs such that any graph G that satisfies $\mathcal{P}$ also satisfies* $\Pr_G(\mathcal{H}) \ge \frac{2}{3}$*, and any graph G that is ϵ-far from satisfying $\mathcal{P}$ satisfies* $\Pr_G(\mathcal{H}) \le \frac{1}{3}$.

The above motivates us to try deducing the q-statistic of the graph from the densities of one of its regular partitions, as per the following definition.

Definition 7. *For an equipartition $\mathcal{A} = \{V_1, \dots, V_t\}$ of G, a (γ, ϵ)-*signature *of $\mathcal{A}$ is a sequence $\mathcal{S} = (\eta_{i,j})_{1 \le i < j \le t}$, such that $|d(V_i, V_j) - \eta_{i,j}| \le \gamma$ for every $i < j$ but at most $\epsilon\binom{t}{2}$ of the pairs. A (γ, γ)-signature is simply referred to as a γ-*signature. *We use just the term* signature *for $\mathcal{S}$ as above when we do not commit to any specific error parameters.*

Given a signature $\mathcal{S} = (\eta_{i,j})_{1 \le i < j \le t}$ as above, the perceived q-statistic *of G according to $\mathcal{S}$ is the following probability distribution over labeled graphs with q vertices: To choose H with the vertex set $v_1, \dots, v_q$, we first choose uniformly and without repetitions a sequence of indices $i_1, \dots, i_q$ from $\{1, \dots, t\}$. We then independently take every $v_k v_l$ for $k < l$ to be an edge of H with probability η_{i_k, i_l} if $i_k < i_l$, and with probability η_{i_l, i_k} if $i_k > i_l$.*

Given a family $\mathcal{H}$ of graphs with q vertices, we denote the probability for obtaining a member of $\mathcal{H}$ according to the perceived q-statistic by $\Pr_{\mathcal{S}}(\mathcal{H})$.

The following lemma shows that for a regular partition, the perceived statistic is indeed close to the statistic of the graph.

Lemma 3.8. *For every q and ϵ there exist $\gamma = \gamma_{3.8}(q, \epsilon)$ and $r = r_{3.8}(q, \epsilon)$, so that for every γ-regular partition $\mathcal{A}$ of G into $t \ge r$ sets, where G has $n \ge N_{3.8}(q, \epsilon, t)$ vertices, and for every γ-signature $\mathcal{S}$ of $\mathcal{A}$, the variation distance between the perceived q-statistic according to $\mathcal{S}$ and the (actual) q-statistic of G is at most ϵ.*

Proof. Recall Definition 3 for the variation distance between two distributions over a combinatorial structure. Here the structure is the set of labeled graphs on q vertices. The perceived statistic distribution is as defined above, and the actual statistic is as defined by the process of picking a random q-size labeled subgraph of G in Definition 6.

Set $r = 7\binom{q}{2}/\epsilon$ and $\gamma = \min\{\epsilon/7\binom{q}{2}, \gamma_{3.5}(q, \epsilon/7)\}$. Let $v_1, \dots, v_q$ be a uniformly random set of q distinct vertices, and let i_j for every $1 \le j \le q$ denote the index for which $v_i \in V_{i_j}$. With probability at least $1 - 4\epsilon/7$, $i_1, \dots, i_q$ are distinct, and, moreover, all the pairs V_{i_j}, V_{i_k} are γ-regular and satisfy $|\eta_{i_j, i_k} - d(V_{i_j}, V_{i_k})| \le \epsilon/7\binom{q}{2}$. Also, note that $\sum_{i=1}^{t} |(|V_i|/n) - 1/t| \le \epsilon/7$ for an appropriate choice of $N_{3.8}(q, \epsilon, t)$.

Finally, for a specific fixed sequence $i_1, \dots, i_q$ for which the above event holds, Lemma 3.5 guarantees that the conditional distribution of the induced graph on $v_1, \dots, v_q$ is not more than $\epsilon/7$-far (in the variation distance) from the distribution on a random graph over $v_1, \dots, v_q$ for which every edge $v_i v_j$ is independently selected with probability $d(V_{i_j}, V_{i_k})$. Noting that $|d(V_{i_j}, V_{i_k}) - \eta_{\min\{i_j, i_k\}, \max\{i_j, i_k\}}| \le \epsilon/7\binom{q}{2}$ and using Lemma 3.4, it follows that the variation distance between the q-statistic of G and the perceived statistic distribution is (after summing all the above error terms) at most ϵ. □

By now we note that knowing an accurate enough signature of a regular partition enables us to estimate the q-statistics of a graph, which in turn enables us to predict the behavior of a property tester (by Lemma 3.7) and thus distinguish between graphs that satisfy the property and graphs which are ϵ-far from satisfying it.

However, for estimability we would like to know more than that. It is not enough to know whether the input graph G has a regular partition that indicates its acceptance by the property tester; we also need to know how far our input graph G is from a graph G' that has such a partition. The problem is that the regular partition that indicates the acceptance of G' may be different from the regular partition found for G. Our technique is to find in G a partition that, in addition to being regular, will be able to "withstand" a repartitioning according to such a G'. This issue and the issue of efficiently finding a signature for such a partition are addressed in the next section.

4. Robust and final partitions and proving Theorem 2.1. To prove the main result, we must first define a framework that allows us to extend the notion of regular partitions. To this end let us first delve a little into the details of the proof of the original regularity lemma. We start with the basic function defined in [18] to track graph partitions with respect to their possible regularity.

DEFINITION 8. *For an equipartition $\mathcal{A}$ of a graph G into t sets, its* index $\mathrm{ind}(\mathcal{A})$ *is defined as* $t^{-2}\sum_{1\le i<j\le t} d^2(V_i,V_j)$. *For a function $f:\mathbb{N}\to\mathbb{N}$ and a constant γ, we say that $\mathcal{A}$ as above is* (f,γ)-robust *if there exists no refinement $\mathcal{B}$ of $\mathcal{A}$ with up to $f(t)$ sets for which* $\mathrm{ind}(\mathcal{B})\ge\mathrm{ind}(\mathcal{A})+\gamma$.

The main lemma used in [18] for proving Szemerédi's lemma can be paraphrased as the following (note that in the proof of Lemma 3.6 as presented in [8, Chapter 7], instead of $\mathrm{ind}(\mathcal{A})$ a similar function that is denoted there by "q" is used, and the equipartitions are allowed to have a small number of "exceptional vertices" not in any set).

LEMMA 4.1 (see [18]; see also [8, Lemma 7.2.4]). *For every ϵ there exist $\gamma=\gamma_{4.1}(\epsilon)$ and $f=f_{4.1}^{(\epsilon)}:\mathbb{N}\to\mathbb{N}$ such that every (f,γ)-robust partition is also ϵ-regular.*

In the original formulation of [18], it is proved that a non–ϵ-regular partition into t sets has a refinement into $\max\{\exp(t),\exp(1/\epsilon)\}$ many sets whose index is larger by at least some $\mathrm{poly}(\epsilon)$ (without explicitly stating Lemma 4.1). With either formulation, the move from Lemma 4.1 to Lemma 3.6 is made through the following simple observation.

Observation 4.2. For every k, γ, and $f:\mathbb{N}\to\mathbb{N}$ there exists $T=T_{4.2}(k,\gamma,f)$, such that every equipartition $\mathcal{A}$ of a graph G with $n\ge N_{4.2}(k,\gamma,f)$ vertices into at most k sets has a refinement $\mathcal{B}$ into at most T sets that is (f,γ)-robust.

Proof. We start by setting $\mathcal{B}=\mathcal{A}$, but if it is not (f,γ)-robust, then we replace it with the refinement showing this, repeating the procedure as many times as necessary. Since the index of a partition is always between 0 and 1, this process will terminate after at most $1/\gamma$ iterations. □

In the following we will also consider robust partitions for choices of f that grow faster than what is required for ϵ-regularity. This means that in some sense we will use a strengthening of the original regularity lemma.

The following definition is clearly a strengthening of the definition of robustness.

DEFINITION 9. *For a function $f:\mathbb{N}\to\mathbb{N}$ and a constant γ, we say that $\mathcal{A}$ as above is* (f,γ)-final *if there exists no partition $\mathcal{B}$ (even one that is not a refinement of $\mathcal{A}$) with at least t and up to $f(t)$ sets for which* $\mathrm{ind}(\mathcal{B})\ge\mathrm{ind}(\mathcal{A})+\gamma$.

The following is an analogue of Observation 4.2 to final partitions. The price is that now we can no longer demand that the final partition be a refinement of a given equipartition.

Observation 4.3. For every k, γ, and $f:\mathbb{N}\to\mathbb{N}$ there exists $T=T_{4.3}(k,\gamma,f)$ such that for every graph G with $n\ge N_{4.3}(k,\gamma,f)$ vertices there exists an equipartition $\mathcal{A}$ into at least k and at most T sets that is (f,γ)-final. □

In fact we do not need the stronger but less flexible condition of finality for our combinatorial statements, but we use it because the parameters of a final partition are easier to detect than those of a robust one. A testing algorithm can actually compute a signature of a final partition like the one that Observation 4.3 guarantees for a graph G, as the following lemma shows.

LEMMA 4.4. *For every k, γ, and $f:\mathbb{N}\to\mathbb{N}$ there exists $q=q_{4.4}(k,\gamma,f)$ such that there exists an algorithm that makes up to q queries to a graph G with*

$n \geq N_{4.3}(k, \frac{1}{2}\gamma, f)$ vertices, computing with probability at least $\frac{2}{3}$ a γ-signature for an (f, γ)-final partition of G into at least k and at most $T_{4.3}(k, \frac{1}{2}\gamma, f)$ sets.

This lemma is proved in section 5 and brings us halfway towards our estimability result.

At this point, if from a signature of a regular partition of G we could estimate how far G is from having a regular partition with a different given signature, then we could use it to estimate how far G is from having a statistic that will cause a canonical tester to accept it with high probability. This we cannot do directly, but we can instead estimate such a difference if we are provided with a signature of a partition that is somewhat more than regular, that is, robust with respect to an appropriate function. We explain in section 6 why a regular partition is not enough while a robust one is. We now present the formal statement of the appropriate result and show how it implies Theorem 2.1.

LEMMA 4.5. *For every q and δ there exist $\gamma = \gamma_{4.5}(q, \delta)$, $s = s_{4.5}(q, \delta)$, and $f = f_{4.5}^{(q,\delta)} : \mathbb{N} \to \mathbb{N}$ with the following property. For every family $\mathcal{H}$ of graphs with q (labeled) vertices there exists a deterministic algorithm that receives as an input only a γ-signature $\mathcal{S}$ for an (f, γ)-robust partition $\mathcal{A}$ with $t \geq s$ sets of a graph G with $n \geq N_{4.5}(q, \delta, t)$ vertices and distinguishes (using no information on G apart from $\mathcal{S}$ and t) given any ϵ between the case that G is $(\epsilon - \delta)$-close to some graph G' for which $\Pr_{G'}(\mathcal{H}) \geq \frac{2}{3}$ and the case that G is ϵ-far from every graph G' for which $\Pr_{G'}(\mathcal{H}) > \frac{1}{3}$.*

This lemma is proved in section 6. Lemmas 4.4 and 4.5 together imply the main result.

Proof of Theorem 2.1. Suppose that $\mathcal{P}$ is a testable graph property, and let ϵ and δ be constants for which we want to (ϵ, δ)-estimate $\mathcal{P}$. As $\mathcal{P}$ is in particular $\frac{1}{2}\delta$-testable, Lemma 3.7 asserts that there exist a constant q and a family $\mathcal{H}$ of graphs on q vertices, such that for every graph G that is in $\mathcal{P}$, $\Pr_G(\mathcal{H}) \geq 2/3$, and for any graph G that is $\frac{1}{2}\delta$-far from $\mathcal{P}$, $\Pr_G(\mathcal{H}) \leq 1/3$.

Set $\gamma = \gamma_{4.5}(q, \frac{1}{2}\delta)$, $f = f_{4.5}^{(q,\delta/2)}$, and $k = s_{4.5}(q, \frac{1}{2}\delta)$, and apply the algorithm provided by Lemma 4.4, with the parameters k, γ, and f, on the input graph G. This algorithm makes up to $q_{4.4}(k, \gamma, f)$ queries to the graph G and with probability at least $\frac{2}{3}$ returns a γ-signature $\mathcal{S}$ of an equipartition of G into at least $s_{4.5}(q, \delta)$ sets and at most $T_{4.3}(k, \frac{1}{2}\gamma, f)$ sets that is (f, γ)-final.

We now apply the algorithm that is provided by Lemma 4.5, with parameters q, $\frac{1}{2}\delta$, and $\epsilon - \frac{1}{2}\delta$, to the signature $\mathcal{S}$ (remember that this is a deterministic algorithm making no additional queries). Due to the choice of parameters, it is guaranteed by Lemma 4.5 that we can distinguish between the case that there is a graph G' that is $(\epsilon - \delta)$-close to G and for which $\Pr_{G'}(\mathcal{H}) \geq \frac{2}{3}$ and the case that G is $(\epsilon - \frac{1}{2}\delta)$-far from every graph G' for which $\Pr_{G'}(\mathcal{H}) > \frac{1}{3}$. In the first case G is accepted, and in the second case it is rejected.

For the above to work we require that

$$n \geq \max\left\{N_{4.3}\left(k, \frac{1}{2}\gamma, f\right), N_{4.5}\left(q, \frac{1}{2}\delta, T_{4.3}\left(k, \frac{1}{2}\gamma, f\right)\right)\right\}.$$

For a smaller n we can just read the entire input and compute its distance from the property to be estimated. We now claim that the above algorithm is indeed an (ϵ, δ)-estimation algorithm for $\mathcal{P}$ for every large enough n.

If G is $(\epsilon - \delta)$-close to $\mathcal{P}$, then by the premises above, it is also $(\epsilon - \delta)$-close to a graph G' for which $\Pr_{G'}(\mathcal{H}) \geq \frac{2}{3}$, and so the first case above will hold as long as $\mathcal{S}$ is in fact a γ-signature of an (f,γ)-robust partition of G, which happens with probability at least $\frac{2}{3}$. Thus G is accepted with probability at least $\frac{2}{3}$.

On the other hand, if G is ϵ-far from $\mathcal{P}$, then by the triangle inequality it is $(\epsilon - \frac{1}{2}\delta)$-far from any graph G' for which $\Pr_{G'}(\mathcal{H}) > \frac{1}{3}$ (because such a G' would be $\frac{1}{2}\delta$-close to satisfying $\mathcal{P}$, as q was chosen to suffice for testing $\mathcal{P}$ with distance parameter $\frac{1}{2}\delta$). Thus, if $\mathcal{S}$ is indeed a γ-signature of an (f,γ)-robust partition, then the algorithm rejects G, and this again happens with probability at least $\frac{2}{3}$.

With both cases covered, the proof is concluded. □

5. Proof of Lemma 4.4. Our strategy as outlined here is rather simple. Let k, γ, and f be as in the formulation of the lemma, and set $T = T_{4.3}(k, \gamma/2, f)$. For every $s \in \{k, \dots, f(T)\}$ we quantize all possible signatures of equipartitions into s parts, choosing such a finite family of signatures so that every possible signature of an s-partition is close enough to one of the chosen signatures. For every such chosen signature we test whether there exists a partition into s sets with densities that are as determined by the signature, allowing for a small slack. This is done using the test of Goldreich, Goldwasser, and Ron for generalized graph partitions [13]. For every positive answer (namely, that such a partition exists) we record the signature and estimate the index of the partition. Having all this information, we set for every s the quantity $M(s)$ that is the largest index of any of the partitions into s sets that we (approximately) know about. We then set s^* to be such that for every s for which $s^* \le s \le f(s^*)$, the records indicate that $M(s) \le M(s^*) + \frac{3}{4}\gamma$. Finally, we output the signature that achieves $M(s^*)$ and claim that it is a signature of an (f,γ)-final equipartition.

To see that such an s^* indeed exists, consider the $(f, \frac{1}{2}\gamma)$-final equipartition $\mathcal{A}$ that is guaranteed by Observation 4.3 for k, γ, and f. $\mathcal{A}$ is a partition into $b \le T$ sets with some signature $\mathcal{S}$. Thus, while passing through all possible signatures of equipartitions into b sets in the process above, the closest signature to $\mathcal{S}$ must have been considered and the corresponding index, which is a good approximation of $\mathrm{ind}(\mathcal{A})$, was computed. Now, as $\mathcal{A}$ is $(f, \frac{1}{2}\gamma)$-final, it follows by the definitions that $s^* = b$ is a valid answer to the output above, assuming that all the index estimations are good enough. Let us now proceed with the formal details.

Set $\epsilon = \gamma/(24 \cdot f^2(T))$. We assume that ϵ^{-1} is an integer without loss of generality, as otherwise we can decrease it a little more (by a factor of less than 2) without changing the essence of the arguments. For every $k \le s \le f(T)$ set $S(s,\epsilon) = \{0, \epsilon, 2\epsilon, \dots, 1\}^{\binom{s}{2}}$. Every $\mathcal{S} \in S(s,\epsilon)$ is clearly associated with a signature of a possible equipartition of G into s sets.

As we have only signatures to work with, we have to use them to estimate the index of a partition.

Definition 10. *In a manner analogous to the definition of the index of a partition, we define the* index *of a signature* $\mathcal{S} = (\eta_{i,j})_{1 \le i < j \le t}$ *to be* $\mathrm{ind}(\mathcal{S}) = t^{-2} \sum_{1 \le i < j \le t} (\eta_{i,j})^2$.

Following is an obvious observation (by a simple calculation) that relates the index of any ϵ-signature of a partition to the index of the partition.

Observation 5.1. Let $\mathcal{A}$ be an equipartition into s sets and assume that $\mathcal{S} = (\eta_{i,j})_{1 \le i < j \le s}$ is an ϵ-signature of $\mathcal{A}$. Then $|\mathrm{ind}(\mathcal{A}) - \mathrm{ind}(\mathcal{S})| \le 3\epsilon$. □

Let G be a graph with n vertices and let s be fixed. Let $0 \le \alpha_{i,j} < \beta_{i,j} \le 1$, $1 \le i < j \le s$ be two sequences of numbers. The following is a special case of a theorem proved by Goldreich, Goldwasser, and Ron [13] (in [13], there are lower and upper bounds on the sizes of the vertex sets, too, but having them here does not make an essential difference).

LEMMA 5.2 (GGR-test of graph partitions [13]). *For a fixed s, let $\mathcal{P}$ be the property of a graph G with n vertices having an equipartition $V_1, \dots, V_s$ of its vertex set, such that $\alpha_{i,j} \leq d(V_i, V_j) \leq \beta_{i,j}$ for every $1 \leq i < j \leq s$ (for fixed, given $\alpha_{i,j} < \beta_{i,j}$).*

Property $\mathcal{P}$ is testable, with a number of queries that is polynomial in ϵ (for every fixed s) and is independent of n.

We use the following guarantee on the approximation of a signature given by a GGR-test.

LEMMA 5.3. *Let $s \geq 2/\epsilon$ be fixed, let $\mathcal{S} = (\eta_{i,j})_{1 \leq i < j \leq s}$ be a signature, and let $\alpha = (\alpha_{i,j})_{1 \leq i < j \leq s}$ and $\beta = (\beta_{i,j})_{1 \leq i < j \leq s}$ be defined by $\alpha_{i,j} = \eta_{i,j} - \epsilon$ and $\beta_{i,j} = \eta_{i,j} + \epsilon$ for $1 \leq i < j \leq s$. Then applying the* GGR*-test on a graph G with s, α, β, and distance parameter ϵ results in the following.*

(i) *If the test accepts with probability more than $\frac{1}{3}$, then there exists an equipartition $\mathcal{A}$ of G into s sets for which $\mathcal{S}$ is an $s^2\epsilon$-signature.*

(ii) *If there is an equipartition $\mathcal{A}$ of G into s sets for which $\mathcal{S}$ is an $(\epsilon, 0)$-signature, then the test accepts with probability at least $\frac{2}{3}$.*

Proof. The first thing to note is that the GGR-property to be tested is exactly the property that $\mathcal{S}$ is an $(\epsilon, 0)$-signature for some partition of G. This immediately yields the second item in the assertion of the lemma.

For the first item, assume that the test accepts with probability more than $\frac{1}{3}$ when applied with s, α, and β. Then there must be a graph G' that is ϵ-close to G and that has an equipartition $\mathcal{A}$ for which $\mathcal{S}$ is an $(\epsilon, 0)$-signature. Thus $\mathcal{A}$, considered as an equipartition of G, must have $|d_G(V_i, V_j) - d_{G'}(V_i, V_j)| \leq \frac{1}{2}s^2\epsilon$ for every $1 \leq i < j \leq s$ (as otherwise G' will be more than ϵ-far from G), and therefore $\mathcal{S}$ is an $s^2\epsilon$-signature for G'. □

We are now ready to conclude this section.

Proof of Lemma 4.4. Suppose that the parameters f, γ, and k are given, and set $T = T_{4.3}(k, \frac{1}{2}\gamma, f)$. For $s \in \{k, \dots, f(T)\}$, let ϵ and $S(s, \epsilon)$ be as defined above, and let $m = \sum_{s=k}^{f(T)} \epsilon^{-\binom{s}{2}}$ be the total number of members in the union of all $S(s, \epsilon)$ for $k \leq s \leq f(T)$.

We use the following procedure for every $s \in \{k, \dots, f(T)\}$.

(i) Initialize $M(s) = 0$. This variable will contain the supposed maximum index of any equipartition into s sets.

(ii) for every $\mathcal{S} = (\eta_{i,j})_{1 \leq i < j \leq t} \in S(s, \epsilon)$, define α and β by $\alpha_{i,j} = \eta_{i,j} - \epsilon$ and $\beta_{i,j} = \eta_{i,j} + \epsilon$ for $1 \leq i < j \leq s$ (just as in Lemma 5.3).

Apply the GGR-test on G with parameters α, β and distance parameter ϵ for $100 \log m$ times. If the majority of the runs accept, then we say that $\mathcal{S}$ was *accepted.* In this case we take $\max\{M(s), \mathrm{ind}(\mathcal{S})\}$ to be the new value of $M(s)$ and record the signature $\mathcal{S}$ if it is the one for which this maximum is obtained. If the test rejects on the majority of the runs, then we do nothing and say that $\mathcal{S}$ was *rejected.*

Note that in the second step above we need to go over all signatures $\mathcal{S} \in S(s, \epsilon)$. It is not hard to generate and go over them in a lexicographic order.

Let s^* be the smallest number in $\{k, \dots, T\}$ such that $M(s^*) + \frac{3}{4}\gamma \geq M(s')$ for every $s' \in \{s^* + 1, \dots, f(s^*)\}$. If there exists such an s^*, output the signature $\mathcal{S}^*$ that achieves the maximum for s^*. Otherwise, the algorithm fails.

It is clear that the algorithm above uses a constant number of queries (on account of using a constant number of GGR-tests). We now need to show that with probability at least $\frac{2}{3}$, the algorithm indeed produces a γ-signature of an (f,γ)-final partition of G into at least k and at most T sets. We conclude the proof using the following claims.

CLAIM 5.4. *With probability at least $\frac{2}{3}$ the following holds. For every $s \in \{k,\dots,f(T)\}$ and every $\mathcal{S} \in S(s,\epsilon)$ which the algorithm accepted, there is an equipartition $\mathcal{A}_{\mathcal{S}}$ into s sets, with $|\mathrm{ind}(\mathcal{A}_{\mathcal{S}}) - \mathrm{ind}(\mathcal{S})| \le 3s^2\epsilon$ and with $\mathcal{S}$ as its $s^2\epsilon$-signature, and for every such s and $\mathcal{S}$ which were rejected by the algorithm, there exists no equipartition $\mathcal{A}_{\mathcal{S}}$ for which $\mathcal{S}$ is an $(\epsilon,0)$-signature.*

Proof. We prove for each of the two parts of the claim that it occurs with probability at least $\frac{5}{6}$, and so it follows that the entire claim holds with probability at least $\frac{2}{3}$. We start with the second part.

Let s and $\mathcal{S}$ be such that $\mathcal{S}$ is an $(\epsilon,0)$-signature for some $\mathcal{A}$. Then by Lemma 5.3 it will be accepted by any one run of the GGR-test (with the corresponding parameters) with probability at least $2/3$. Thus, it will be rejected by the test only if it is rejected by the majority of the $100\log m$ runs, which by Lemma 3.1 will occur with probability at most $1/(6m)$. Hence, with probability at least $5/6$ the test will accept all such $\mathcal{S}$ as above. This proves that the second part of the claim occurs with probability at least $5/6$.

For the first part of the claim, let us assume now that $\mathcal{S}$ is not an $s^2\epsilon$-signature for any possible equipartition of G into s sets. By Lemma 5.3 this means that every run of the GGR-test will reject $\mathcal{S}$ with probability at least $2/3$. Hence, by Lemma 3.1, the probability that $\mathcal{S}$ is accepted by the majority of the runs is no more than $1/6m$. This implies that with probability at least $5/6$, every signature $\mathcal{S}$ that was accepted by our algorithm is an $s^2\epsilon$-signature of some equipartition $\mathcal{A}_{\mathcal{S}}$ of G into s sets, and then by Observation 5.1 this means that $\mathcal{S}$ and $\mathcal{A}_{\mathcal{S}}$ satisfy $|\mathrm{ind}(\mathcal{A}_{\mathcal{S}}) - \mathrm{ind}(\mathcal{S})| \le 3s^2\epsilon$.

We have proved that each of the parts occurs with probability at least $5/6$, and so the claim that both of them hold with probability at least $2/3$ follows. □

CLAIM 5.5. *If the event of Claim 5.4 occurred, then the algorithm succeeds in the following sense: The algorithm does not fail in its last step, and the signature it outputs is an $s^2\epsilon$-signature of some (f,γ)-final partition.*

Proof. We assume that the event of Claim 5.4 indeed occurred, and first show that the algorithm does not fail in the last step.

Set s_1 to be the smallest s for which G has an $(f,\gamma/2)$-final partition into s_1 sets. The fact that such an $s_1 \in \{k,\dots,T\}$ exists is asserted in Observation 4.3. Let $\mathcal{A}$ be the corresponding $(f,\gamma/2)$-final equipartition with the largest index (if there are more than one, then let $\mathcal{A}$ be the first one in the lexicographic order of its signature). Then, by the fact that $\mathcal{A}$ is $(f,\gamma/2)$-final, we have that $\mathrm{ind}(\mathcal{A}) + \gamma/2 \ge \mathrm{ind}(\mathcal{S})$ for any equipartition $\mathcal{S}$ into at least s_1 and at most $f(s_1)$ sets. Also by our choice of $\mathcal{A}$ we have $\mathrm{ind}(\mathcal{A}) \ge \mathrm{ind}(\mathcal{A}')$ for any equipartition $\mathcal{A}'$ into s_1 sets. Let $\mathcal{S} \in S(s_1,\epsilon)$ be the first in lexicographic order such that $\mathcal{S}$ is an $(\epsilon,0)$-signature of $\mathcal{A}$. Obviously there exists such an $\mathcal{S}$ by the choice of $S(s_1,\epsilon)$.

Thus, assuming that the sampler accepted all signatures which were $(\epsilon,0)$-signatures of a corresponding partition, $\mathcal{S}$ was in particular accepted. By Observation 5.1, together with the fact that $\mathrm{ind}(\mathcal{A}) \ge \mathrm{ind}(\mathcal{A}')$ for any equipartition $\mathcal{A}'$ into s_1 sets, it follows that

$$\mathrm{ind}(\mathcal{A}) - 3s^2\epsilon \le M(s_1) \le \mathrm{ind}(\mathcal{A}) + 3s^2\epsilon.$$

Moreover, by combining the inequalities above and Observation 5.1, we get that as long as all signatures that were not $s^2\epsilon$-signatures of some equipartition were rejected, the following holds. For any equipartition $\mathcal{B}$ into s sets with $s_1 \le s \le f(s_1)$ that has a corresponding $s^2\epsilon$-signature $\mathcal{T} \in S(s,\epsilon)$ that was accepted by the algorithm, we have $\text{ind}(\mathcal{T}) \leq \text{ind}(\mathcal{B}) + 3s^2\epsilon \leq \text{ind}(\mathcal{A}) + \gamma/2 + 3s^2\epsilon \leq M(s_1) + \gamma/2 + 6s^2\epsilon$.

Now this implies that $\text{ind}(\mathcal{T}) \leq M(s_1) + \gamma/2 + 6f(s_1)^2\epsilon \leq M(s_1) + \frac{3}{4}\gamma$ by our choice of $\epsilon = \gamma/24f(T)^2$. Thus s_1 is recognized as a candidate for s^*, and hence the sampler will not fail to output some s^* and $\mathcal{S}^*$ (we do not claim that the sampler actually outputs s_1 as s^*, but only that the existence of s_1 ensures that the algorithm does not fail to output something in the last step).

It remains to show that if the event of Claim 5.4 occurs and the sampler outputs a signature $\mathcal{S}^*$ with index s^*, then there exists a corresponding (f,γ)-final equipartition. Indeed, this event implies that there exists an equipartition $\mathcal{A}^*$ into s^* sets so that $\mathcal{S}^*$ is its $s^2\epsilon$-signature. This also means that for all $s \in \{s^*,\dots,f(s^*)\}$ and all signatures $\mathcal{S} \in S(s,\epsilon)$, no such signature satisfying $\text{ind}(\mathcal{S}) > M(s^*) + \frac{3}{4}\gamma$ is an $(\epsilon,0)$-signature of any equipartition of G (as these signatures were rejected by the algorithm). Now if $\mathcal{A}^*$ was not (f,γ)-final, then there would be an equipartition $\mathcal{B}$ with $s \in \{s^*,\dots,f(s^*)\}$ sets for which $\text{ind}(\mathcal{B}) \geq \text{ind}(\mathcal{A}^*) + \gamma \geq M(s^*) + \gamma - 3(s^*)^2\epsilon$. But if we set $\mathcal{S}$ to be an $(\epsilon,0)$-signature of $\mathcal{B}$ (by approximating each pair density of $\mathcal{B}$ by its closest multiple of ϵ), this would imply, by Observation 5.1, that $\text{ind}(\mathcal{S}) \geq M(s^*) + \gamma - 3(s^*)^2\epsilon - 3\epsilon > M(s^*) + \frac{3}{4}\gamma$, a contradiction since such an $\mathcal{S}$ (which would have been accepted by the algorithm) means that $\mathcal{S}^*$ would not be a valid output. ∎

To summarize, by Claim 5.4, with probability at least $\frac{2}{3}$ the sampler accepts all signatures under consideration that are $(\epsilon,0)$-signatures of some corresponding equipartition and rejects all signatures that are not $s^2\epsilon$-signatures of any equipartition. Then, by Claim 5.5, whenever this event occurs the algorithm will output without fail a γ-signature for some (f,γ)-final equipartition. Together this means that with probability at least $\frac{2}{3}$ the algorithm will supply the desired output, concluding the proof of Lemma 4.4. ∎

6. Proof of Lemma 4.5. By Lemma 3.8 (using Lemma 3.7 about canonical testing), if we know a signature of a regular partition of a graph G, then this is enough to distinguish whether the graph satisfies a given testable property or is δ-far from satisfying it. For estimability we would like to go a step further and use a signature of G to approximate its distance from any graph G' that the δ-test may accept.

However, knowing just the signature of a regular partition of G is insufficient, since regular partitions of two graphs of small relative distance might still be quite different (and have quite different signatures). Thus, if G does not satisfy a testable property but is close to satisfying it as witnessed by a graph G', then a regular partition of G with a corresponding signature may still not provide us with information about the regular partition of G' and thus about the distance of G from the property. Instead, our strategy will be to ask for a signature of a partition $\mathcal{A}$ of G that is robust enough to ensure that G' will have a regular partition that is a refinement of $\mathcal{A}$ which is still regular for G. With this setting, we will also be able to calculate a signature in G for the new partition of G', using only the signature of $\mathcal{A}$ in G. This will enable us to compare possible signatures for estimating the distance between G and the hypothetical G'.

We now turn to the formal proof. We first need some definitions about distances of signatures and about how signatures behave under refinements of equipartitions.

Definition 11. *The* distance *between the signatures* $\mathcal{S} = (\eta_{i,j})_{1\leq i<j\leq t}$ *and* $\mathcal{S}' = (\eta'_{i,j})_{1\leq i<j\leq t}$ *is defined as the average density difference* $\sum_{1\leq i<j\leq t} |\eta_{i,j} - \eta'_{i,j}|/\binom{t}{2}$.

Given a signature $\mathcal{S} = (\eta_{i,j})_{1\leq i<j\leq t}$ *for an equipartition* $\mathcal{A}$ *and a refinement*

$\mathcal{B} = \{W_1, \ldots, W_s\}$ *of* $\mathcal{A}$*, the* extension *of* $\mathcal{S}$ *to* $\mathcal{B}$ *is the sequence* $\mathcal{S}' = (\eta'_{i,j})_{1 \le i < j \le s}$ *defined by setting* $\eta'_{i,j} = \eta_{k,l}$ *if there exist* $k \ne l$ *such that* $W_i \subset V_k$ *and* $W_j \subset V_l$*, and by arbitrarily setting* $\eta'_{i,j} = 0$ *if* W_i *and* W_j *are both subsets of the same* V_k*.*

The following follows directly from the above definition (for any equipartition, disregarding the regularity conditions).

Observation 6.1. For every ϵ and s there exist $r = r_{6.1}(\epsilon)$ and $N = N_{6.1}(\epsilon, s)$ satisfying the following. Suppose that G and G' are α-close graphs on the same vertex set of size $n \ge N$, and that $\mathcal{S}$ and $\mathcal{S}'$ are γ and γ' signatures, respectively, of the same equipartition $\mathcal{A}$ of the vertex set of G and G' into $s \ge r$ sets. Then the distance between $\mathcal{S}$ and $\mathcal{S}'$ is at most $\alpha + \epsilon + 2(\gamma + \gamma')$.

Proof (*sketch*). Setting $r = 2/\epsilon$, it is clear that for n large enough the 0-signatures (i.e., the sequences of actual densities) of $\mathcal{A}$ over G and G' differ by no more than $\alpha+\epsilon$. Also, it is not hard to see that the 0-signature and any γ-signature of $\mathcal{A}$ over G differ by no more than 2γ, and, similarly, the 0-signature and any γ'-signature of $\mathcal{A}$ over G' differ by no more than $2\gamma'$. We conclude the proof using the triangle inequality. □

Given a signature for a regular partition of G, we can use it to bound the distance of G from some other graph that shares the same regular partition.

LEMMA 6.2. *For every* ϵ *and* t *there exist* $\gamma = \gamma_{6.2}(\epsilon)$ *and* $N = N_{6.2}(t, \epsilon)$*, such that for every graph* G *with* $n \ge N$ *vertices, if* $\mathcal{S}$ *is a* γ*-signature of a* γ*-regular partition* $\mathcal{A}$ *of* G *with* t *sets, then for every signature* $\mathcal{S}'$ *that is* δ*-close to* $\mathcal{S}$ *for some* δ*, there is a graph* G' *(with the same vertex set) that is* $(\delta + \epsilon)$*-close to* G*, so that* $\mathcal{A}$ *is an* ϵ*-regular partition of* G' *and* $\mathcal{S}'$ *is an* ϵ*-signature thereof.*

Before we continue, we note that the converse is false, as there could be two graphs that share exactly the same signature but are quite far from each other. For example, two graphs chosen uniformly at random from the set of all graphs with a fixed labeled set of n vertices will be with high probability far from each other, but still share the same signature for the same regular partition, namely the all-$\frac{1}{2}$ signature.

Proof of Lemma 6.2. We set $\gamma = \frac{1}{4}\epsilon$. Given G, $\mathcal{A} = \{V_1, \ldots, V_t\}$, $\mathcal{S} = (\eta_{i,j})_{1 \le i < j \le t}$, and $\mathcal{S}' = (\eta'_{i,j})_{1 \le i < j \le t}$, as above, we create G' from G in the following manner.

(i) For every i, the edges within V_i are unchanged.

(ii) For $i < j$ such that $\eta'_{i,j} < d(V_i, V_j)$, every edge of G between V_i and V_j is removed with probability $1 - \eta'_{i,j}/d(V_i, V_j)$, independently of all other probabilistic actions in this construction.

(iii) For $i < j$ such that $\eta'_{i,j} > d(V_i, V_j)$, every vertex pair of G between V_i and V_j that is not an edge becomes one with probability $1 - (1 - \eta'_{i,j})/(1 - d(V_i, V_j))$, independently of all other probabilistic actions in this construction.

Let G' be the resulting graph. For every $X \subseteq V_i$, $Y \subseteq V_j$ let $d'(X, Y) = d_{G'}(X, Y)$ be the pairwise density with regard to G' (Definition 4). We choose $N > 8t^4/(\gamma^3)$. Making extensive use of Lemma 3.1, we now prove two claims. We first prove that with high probability we will get in G' the correct densities.

CLAIM 6.3. *For every* $1 \le i < j \le t$*,* $|d'(V_i, V_j) - \eta'_{i,j}| > 2\gamma$ *with probability at most* $1/(2t^2)$.

Proof. Suppose first that $\eta'_{i,j} < d(V_i, V_j)$. Then, we have $m = d(V_i, V_j) \cdot (n/t)^2$ edges, where each edge is now removed with probability $p = 1 - \eta'_{i,j}/d(V_i, V_j)$ (independently of other edges). Note that the expected number of removed edges is $E = (d(V_i, V_j) - \eta'_{i,j}) \cdot (n/t)^2$ and thus the expected value of $d'(V_i, V_j)$ is exactly $\eta'_{i,j}$. Hence for the event above to occur, the deviation of the number of edges removed from E has to be more than $2\gamma \cdot (n/t)^2$. Now if $d(V_i, V_j) > 2\gamma$, then m is large enough (assuming that n is large enough) for Lemma 3.1 to ensure that the probability that the deviation above is more than $\gamma \cdot (n/t)^2$ is below the claimed bound and thus to imply the statement. For $d(V_i, V_j) < 2\gamma$ the number of removed edges is at most $d(V_i, V_j)$ and thus the event above occurs with probability 1. If $\eta'_{i,j} > d(V_i, V_j)$, then the argument is analogous so we omit it here. □

Note that if $|d'(V_i, V_j) - \eta'_{i,j}| \leq 2\gamma$ for a pair (i, j), then we have $|d'(V_i, V_j) - d(V_i, V_j)| \leq |d'(V_i, V_j) - \eta'_{i,j}| + |\eta'_{i,j} - \eta_{i,j}| + |\eta_{i,j} - d(V_i, V_j)| \leq 2\gamma + |\eta'_{i,j} - \eta_{i,j}| + |\eta_{i,j} - d(V_i, V_j)|$, and by the assumption on the distance between $\mathcal{S}$ and $\mathcal{S}'$ we also know that $\sum_{1 \leq i < j \leq t} |\eta'_{i,j} - \eta_{i,j}| \leq \binom{t}{2}\delta$. We now prove a claim about the regularity of the pairs in G'.

Claim 6.4. *For every $i < j$ for which V_i, V_j is a γ-regular pair in G, this will not be an ϵ-regular pair in G' with probability at most $1/(2t^2)$.*

Proof. Again we assume that $\eta'_{i,j} < d(V_i, V_j)$, as the argument for the complementary case is analogous. Then, for V_i, V_j not to be ϵ-regular with respect to G' there must be some subsets $X \subseteq V_i$ and $Y \subseteq V_j$ of size $\epsilon n/t$ for which $|d'(X, Y) - d'(V_i, V_j)| > \epsilon$. We call such sets a *violation* at (X, Y). However, since $\mathcal{A}$ is γ-regular for G, we have that $|d(X, Y) - d(V_i, V_j)| \leq \gamma$ (over G). Thus a violation at (X, Y) can occur only when the number of removed edges from $e(X, Y)$ deviates from its expectation by more than $(\epsilon - \gamma) \cdot (\epsilon n/t)^2$. Note also that the number of possible edges between X and Y is $m = d(X, Y) \cdot (\epsilon n/t)^2$.

If $d(V_i, V_j) > 2\gamma = \epsilon/2$, then m is large enough (assuming that n is large enough), for Lemma 3.1 to ensure that the probability that the deviation above is more than $(\epsilon - \gamma) \cdot (\epsilon n/t)^2$ is less than $1/(2t)^2 \cdot 2^{-2n/t}$. Thus, by the union bound, the probability that there exists a pair (X, Y) for which a violation occurs is bounded above by $(1/(2t)^2 \cdot 2^{-2n/t})2^{|V_i|+|V_j|} \leq 1/(2t^2)$ as claimed.

If $d(V_i, V_j) < 2\gamma$, then the number of removed edges is at most $2\gamma(n/t)^2$, and thus a violation at (X, Y) cannot occur at all (recall that no edges are added by our procedure in the case $\eta'_{i,j} < d(V_i, V_j)$. □

By the analysis above, the union bound (for every $1 \leq i < j \leq t$) implies that there is such a G' for which the assertions of both claims hold simultaneously for every $1 \leq i < j \leq t$. Thus by the statement of Claim 6.4, $\mathcal{S}'$ is a signature for an ϵ-regular partition of G', being an ϵ-signature thereof by the statement of Claim 6.3. In addition, Claim 6.3 implies (as noted right after its proof) that for every pair V_i, V_j, at most a $2\gamma + |\eta'_{i,j} - \eta_{i,j}| + |\eta_{i,j} - d(V_i, V_j)|$ fraction of edges are removed or added while moving from G to G'.

Summing this for all pairs and recalling that $|\eta_{i,j} - d(V_i, V_j)| \leq \gamma$ for all but a γ-fraction of the pairs (due to $\mathcal{S}$ being a γ-signature of G) as well as that $\mathcal{S}$ and $\mathcal{S}'$ are δ-close, we get that the total distance between G and G' is bounded by $2\gamma + \delta + (1 - \gamma)\gamma + \gamma \cdot 1 \leq \delta + 4\gamma \leq \delta + \epsilon$. □

In general, even if G' and G are close enough graphs (but not too close), a regular partition of G is not necessarily regular for G'. Instead, we will look at a refinement of the partition of G that is regular for G'. However, a refinement of a regular partition is not necessarily in itself regular, nor is its signature close to the corresponding extension of the original signature. For this we turn to robustness, with the aid of a lemma about the index of a refinement. The following lemma was proved in [2, Lemma 3.7] (using the Cauchy–Schwartz inequality), although in essence it was also already implicitly proved in [18], in the proof of Lemma 4.1.

Lemma 6.5 (see [2, Lemma 3.7]). *For every ϵ and t there exist $\gamma = \gamma_{6.5}(\epsilon)$ and $N_{6.5}(t, \epsilon)$ satisfying the following. Assume that $\mathcal{A}$ is an equipartition of a graph G with $n \geq N_{6.5}(t, \epsilon)$ vertices into s sets, and that $\mathcal{B}$ is a refinement of $\mathcal{A}$ into at most*

t sets. Assume further that $\mathcal{S}$ is any γ-signature of $\mathcal{A}$ and that $\mathcal{T}$ is its extension to $\mathcal{B}$. If $\mathcal{B}$ satisfies $\mathrm{ind}(\mathcal{B}) \leq \mathrm{ind}(\mathcal{A}) + \gamma$, then $\mathcal{T}$ is an ϵ-signature for $\mathcal{B}$.

The following lemma about the index of a refinement never decreasing too much was also implicitly proved in the course of several regularity-related proofs. See, for example, [8, Lemma 7.2.2].

LEMMA 6.6. *For every ϵ and t there exists $N = N_{6.6}(t, \epsilon)$, so that for every equipartition $\mathcal{A}$ of G with $n \geq N$ vertices into s sets and every refinement $\mathcal{B}$ of $\mathcal{A}$ into at most t sets, $\mathrm{ind}(\mathcal{B}) \geq \mathrm{ind}(\mathcal{A}) - \epsilon$.*

Proof (*sketch*). If t divides n (and hence so does s), then we would have $\mathrm{ind}(\mathcal{B}) \geq \mathrm{ind}(\mathcal{A})$ as a direct consequence of the Cauchy–Schwartz inequality (see, e.g., [8, Lemma 7.2.2]): Set $\mathcal{A} = \{V_i | 1 \leq i \leq s\}$ and $\mathcal{B} = \{W_{i,k} | 1 \leq i \leq s, 1 \leq k \leq t/s\}$, where $\{W_{i,1}, \ldots, W_{i,t/s}\}$ are assumed to be exactly the members of $\mathcal{B}$ that are contained in V_i. It is clear that for all $1 \leq i < j \leq s$ we have that $d(V_i, V_j)$ is the average of the sequence $\langle d(W_{i,k}, W_{j,l}) | 1 \leq k, l \leq t/s\rangle$. Hence, the square of $d(V_i, V_j)$ is at most the average of the squares of $\langle d(W_{i,k}, W_{j,l}) | 1 \leq k, l \leq t/s\rangle$, and from here it is easy to see that $\mathrm{ind}(\mathcal{B}) \geq \mathrm{ind}(\mathcal{A})$.

If t does not divide n, then we may lose on the difference between $\mathrm{ind}(\mathcal{A})$ and $\mathrm{ind}(\mathcal{B})$ on account of rounding errors, but for an appropriate choice of N this loss would be less than ϵ. $\square$

We can now prove the existence of a refinement for $\mathcal{A}$ that is also regular with respect to G', provided that $\mathcal{A}$ is robust enough.

LEMMA 6.7. *For every ϵ there exist $\gamma = \gamma_{6.7}(\epsilon)$ and $f = f^{(\epsilon)}_{6.7} : \mathbb{N} \to \mathbb{N}$ satisfying the following. Suppose that $\mathcal{A}$ is an (f, γ)-robust partition of a graph G into s sets and that $\mathcal{S}$ is a γ-signature of $\mathcal{A}$, where G has $n \geq N_{6.7}(s, \epsilon)$ vertices. Then for every G' that shares the same vertex set as G, there exists a refinement $\mathcal{B}$ of $\mathcal{A}$ into $t \leq T_{3.6}(s, \epsilon)$ sets which is ϵ-regular for both G and G'. Moreover, the corresponding extension of $\mathcal{S}$ to $\mathcal{B}$ is an ϵ-signature with respect to G.*

Proof. We set $\gamma = \min\{\frac{1}{2}\gamma_{4.1}(\epsilon), \gamma_{6.5}(\epsilon)\}$, and for every $k \in \mathbb{N}$ we set $f(k) = f^{(\epsilon)}_{4.1}(T_{3.6}(k, \epsilon))$. We set N to be the maximum over the respective functions of all lemmas that are used in the following (this will be explained later on).

Given a partition $\mathcal{A}$ as above and assuming that $N \geq N_{3.6}(s, \epsilon)$, Lemma 3.6 asserts that there is a refinement $\mathcal{B}$ of $\mathcal{A}$ that partitions $V(G')$ into at most $t \leq T_{3.6}(s, \epsilon)$ sets and is ϵ-regular with respect to G'.

Lemma 6.6, assuming that $N \geq N_{6.6}(T_{3.6}(s, \epsilon), \gamma)$, asserts that $\mathrm{ind}(\mathcal{B}) \geq \mathrm{ind}(\mathcal{A}) - \gamma \geq \mathrm{ind}(\mathcal{A}) - \frac{1}{2}\gamma_{4.1}(\epsilon)$ over G (the last inequality is by the choice of γ). This implies that $\mathcal{B}$ is $(f^{(\epsilon)}_{4.1}, \gamma_{4.1}(\epsilon))$-robust with respect to G, as otherwise it would mean that there is a refinement $\mathcal{C}$ with at most $f^{(\epsilon)}_{4.1}(t)$ sets for which $\mathrm{ind}(\mathcal{C}) > \mathrm{ind}(\mathcal{B}) + \gamma_{4.1}(\epsilon)$, but this would imply that $\mathrm{ind}(\mathcal{C}) > \mathrm{ind}(\mathcal{A}) + \gamma_{4.1}(\epsilon) - \frac{1}{2}\gamma_{4.1}(\epsilon)$, which contradicts the robustness requirement of $\mathcal{A}$. Hence we conclude by Lemma 4.1 (applied to $\mathcal{B}$) that $\mathcal{B}$ is also ϵ-regular with respect to G. This proves that the refinement $\mathcal{B}$ is as needed.

In addition, the original robustness requirement for $\mathcal{A}$ ensures that the index of $\mathcal{B}$ with respect to G is no more than $\mathrm{ind}(\mathcal{A}) + \gamma_{6.5}(\epsilon)$. Hence, Lemma 6.5 ensures that the extension of $\mathcal{S}$ is an ϵ-signature for $\mathcal{B}$ with respect to G, as required. $\square$

In the course of the proof of the above, we also make the following observation.

Observation 6.8. If $\mathcal{A}$ is an $(f^{(\epsilon)}_{6.7}, \gamma_{6.7}(\epsilon))$-robust partition of a graph G into s sets, where G has $n \geq N_{6.7}(s, \epsilon)$ vertices, and $\mathcal{B}$ is any refinement of $\mathcal{A}$ with $t \leq T_{3.6}(s, \epsilon)$ sets, then the extension of any $\gamma_{6.7}(\epsilon)$-signature of $\mathcal{A}$ to $\mathcal{B}$ is an ϵ-signature of $\mathcal{B}$ (over G). $\square$

We are now ready for the proof of Lemma 4.5. The intuition of the proof is the following: Assume that $\mathcal{S}$ is a γ-signature of an equipartition $\mathcal{A}$ that is (f,γ)-robust for G, for a small enough γ and a fast enough growing f. Our decision whether to accept or reject G is based on checking whether there is a refinement $\mathcal{B}$ of $\mathcal{A}$ (with not too many sets) for which the extension $\mathcal{T}$ of $\mathcal{S}$ is close enough to a signature $\mathcal{T}'$ for which the perceived q-statistic satisfies $\Pr_{\mathcal{T}'}(\mathcal{H}) \geq \frac{1}{2}$. If such a refinement exists, then we accept G, and otherwise we reject G.

Now, if there is an $(\epsilon-\delta)$-close graph G' for which $\Pr_{G'}(\mathcal{H}) \geq \frac{2}{3}$, then G will be accepted, as close enough graphs have close signatures (Observation 6.1), and f and γ will be chosen so that $\mathcal{B}$ will be regular enough for both G and G' (as implied by Lemma 6.7), so that the signature $\mathcal{T}'$ of $\mathcal{B}$ with respect to G' (which is close to $\mathcal{T}$) is such that $\Pr_{\mathcal{T}'}(\mathcal{H})$ approximates $\Pr_{G'}(H)$ well enough so that it does not fall below $\frac{1}{2}$. On the other hand, if G is accepted on account of some signature $\mathcal{T}'$ close to $\mathcal{T}$ for which $\Pr_{\mathcal{T}'}(\mathcal{H}) \geq \frac{1}{2}$, then Lemma 6.2 asserts that there is a close enough G' to G, for which the partition $\mathcal{B}$ is regular enough, and for which $\mathcal{T}'$ is indeed a signature ensuring that $\Pr_{G'}(H)$ is close enough to $\Pr_{\mathcal{T}'}(\mathcal{H})$ so that it is larger than $\frac{1}{3}$. We now choose the actual parameters and present the formal proof.

Proof of Lemma 4.5. We set the values $\gamma = \gamma_{6.7}(\gamma_0)$, $s = \max\{r_{3.8}(q,\frac{1}{6}), r_{6.1}(\frac{1}{6}\delta)\}$, and $f(k) = f_{6.7}^{(\gamma_0)}(k)$, where

$$\gamma_0 = \min\left\{\frac{1}{6}\delta,\ \gamma_{3.8}\left(q,\frac{1}{6}\right),\ \gamma_{6.2}\left(\min\left\{\frac{1}{2}\delta, \gamma_{3.8}\left(q,\frac{1}{7}\right)\right\}\right)\right\}.$$

We set N to be the maximum over all respective functions of the lemmas and arguments used in the following (we omit here the exact details of the implicitly assumed lower bounds on n).

Given a γ-signature $\mathcal{S}$ for an (f,γ)-robust partition $\mathcal{A}$ into $t \geq s$ sets, we do the following. We check whether there could be any refinement $\mathcal{B}$ of $\mathcal{A}$ with at most $T_{3.6}(t,\gamma_0)$ sets, for which the extension $\mathcal{T}$ of $\mathcal{S}$ to $\mathcal{B}$ is $(\epsilon - \frac{1}{2}\delta)$-close to any signature $\mathcal{T}'$ such that the perceived q-statistic according to $\mathcal{T}'$ satisfies $\Pr_{\mathcal{T}'}(\mathcal{H}) \geq \frac{1}{2}$. If there exists such a signature, then we accept G, and otherwise we reject it. Note that the existence of the refinement $\mathcal{B}$ depends only on the provided signature $\mathcal{S}$, so we do not make here any additional queries to the graph G. We now prove the two directions that tie the existence of such a $\mathcal{T}'$ with the existence of a corresponding graph G'.

Proof of the first direction. Suppose that G' is any graph that is $(\epsilon-\delta)$-close to G, and for which $\Pr_{G'}(\mathcal{H}) \geq \frac{2}{3}$. We will show that G is accepted by the above procedure. We use here only that $\gamma_0 \leq \min\{\frac{1}{6}\delta, \gamma_{3.8}(q,\frac{1}{6})\}$ in the expressions for γ, s, and f.

Indeed let $\mathcal{A}$ be an (f,γ)-robust partition of G into $t \geq s$ sets and let $\mathcal{S}$ be a γ-signature of $\mathcal{A}$. By Lemma 6.7, there exists a refinement $\mathcal{B}$ of $\mathcal{A}$ into at most $T_{3.6}(t,\gamma_0)$ sets, so that $\mathcal{B}$ is γ_0-regular for both G and G'. Moreover, denoting by $\mathcal{T}$ the corresponding extension of $\mathcal{S}$, we have that $\mathcal{T}$ is a γ_0-signature of $\mathcal{B}$ with respect to G. By the upper bound on γ_0 this implies that $\mathcal{B}$ is $\gamma_{3.8}(q,\frac{1}{6})$-regular for both G and G' and that $\mathcal{T}$ is a $\frac{1}{6}\delta$-signature of $\mathcal{B}$ with respect to G.

Let $\mathcal{T}'$ be the 0-signature of $\mathcal{B}$ over G'. Lemma 3.8 implies (using $\mathcal{B}$ and G') that the perceived statistics with respect to $\mathcal{T}'$ and the actual statistics of G' are of variation distance at most $\frac{1}{6}$. Thus, Lemma 3.2 implies that $\Pr_{\mathcal{T}'}(\mathcal{H}) \geq \frac{2}{3} - \frac{1}{6} = \frac{1}{2}$. In addition, by Observation 6.1 (since $\mathcal{B}$ has at least $r_{6.1}(\frac{1}{6}\delta)$ sets and assuming that n is large enough), $\mathcal{T}'$ is $(\epsilon - \frac{1}{2}\delta)$-close to $\mathcal{T}$ on account of G and G' being $(\epsilon-\delta)$-close graphs. Thus, $\mathcal{T}$ and $\mathcal{T}'$ provide a witness that the procedure above accepts G. □

Proof of the second direction. Let $\mathcal{A}$ be an (f,γ)-robust partition of G into $t \geq s$

sets and let $\mathcal{S}$ be a γ-signature of $\mathcal{A}$. Assume that there is a refinement $\mathcal{B}$ of $\mathcal{A}$ into at most $T_{3.6}(t, \gamma_0)$ sets, for which the extension $\mathcal{T}$ of $\mathcal{S}$ to $\mathcal{B}$ is $(\epsilon - \frac{1}{2}\delta)$-close to a signature $\mathcal{T}'$ such that the perceived q-statistic according to $\mathcal{T}'$ satisfies $\Pr_{\mathcal{T}'}(\mathcal{H}) \geq \frac{1}{2}$.

We will show that there is a graph G' that is ϵ-close to G and for which $\Pr_{G'}(\mathcal{H}) > \frac{1}{3}$. We use here the fact that $\gamma_0 \leq \gamma_{6.2}(\min\{\frac{1}{2}\delta, \gamma_{3.8}(q, \frac{1}{7})\})$ in the expressions for γ, s, and f.

Indeed, Observation 6.8 (regarding $\mathcal{B}$ as a possible refinement of $\mathcal{A}$ with respect to G) asserts that $\mathcal{T}$ is a γ_0-signature of $\mathcal{B}$ (with respect to G), which by the upper bound on γ_0 means that it is a $\gamma_{6.2}(\min\{\frac{1}{2}\delta, \gamma_{3.8}(q, \frac{1}{7})\})$-signature for $\mathcal{B}$ with respect to G.

Now Lemma 6.2 (applied on $\mathcal{T}$ as an appropriate signature of $\mathcal{B}$ and the relatively close signature $\mathcal{T}'$) implies that there is a graph G' that is $(\epsilon - \frac{1}{2}\delta + \frac{1}{2}\delta)$-close to G, namely ϵ-close to G, and for which $\mathcal{T}'$ is a $\gamma_{3.8}(q, \frac{1}{7})$-signature of $\mathcal{B}$, which in turn is $\gamma_{3.8}(q, \frac{1}{7})$-regular over G'. By Lemma 3.8 about the closeness of the q-statistic of G' to the perceived one and Lemma 3.2, $\Pr_{G'}(\mathcal{H}) \geq \frac{1}{2} - \frac{1}{7} > \frac{1}{3}$ as required. □

With both directions proven, the correctness of the above algorithm is now established. □

7. Concluding comments.

Efficient calculation of regular partitions. The main result of [1] is an algorithm that, for a fixed ϵ, calculates for an input graph G an ϵ-regular partition thereof. The algorithm is proved there to be in NC_1 and with deterministic time (in its nonparallel version) that is the same as that of matrix multiplication. By carefully reviewing the proof of Lemma 4.4 we can strengthen the first part of their result. First we give a formal definition for the computational complexity of our algorithms.

DEFINITION 12. *A* TC_0 solution *for a problem is an efficient (polynomial time in* n*) algorithm for constructing a polynomial size (in* n*) circuit for every* n *that gives a correct answer for every input instance of this size, where the height of the circuit is independent of* n *and the circuit consists solely of unlimited fan-in AND ($\wedge$) gates, OR ($\vee$) gates, threshold gates (a threshold gate, for inputs* $y_1, \ldots, y_m$ *and a given-in-advance parameter* t*, checks whether* $\sum_{i=1}^{m} y_i \geq t$*), and negations ($\neg$). By contrast, an* NC_1 solution *allows circuits with only negations and fan-in* 2 *AND/OR gates, but in which the circuit height is* $O(\log n)$.

By our methods we are able to prove the following.

THEOREM 7.1. *For every* k*,* γ*, and* $f : \mathbb{N} \to \mathbb{N}$ *there exists a* TC_0 *solution that for an input graph* G *with* $n \geq N_{4.3}(k, \frac{1}{2}\gamma, f)$ *vertices computes an* (f, γ)*-final partition of* G *into at least* k *and at most* $T_{4.3}(k, \frac{1}{2}\gamma, f)$ *sets.*

Proof (*sketch*). First we show how to calculate only a signature for such a partition. We follow the proof of Lemma 4.4. We recall that whenever the algorithm in the proof needs to accept or reject a signature $\mathcal{S}$, it makes a constant number of iterations of a GGR-test. Here we will instead reject or accept $\mathcal{S}$ based on an estimation of the acceptance probability of one GGR-test. For this end we first construct a deterministic circuit for every possible choice of the queries from G that the GGR-test can make. The queries of a GGR-test are made by first uniformly choosing a constant number of vertices of G; hence there is a polynomial number of such choices, and for each one of them we can use a constant size circuit to know whether the test would have accepted had it made these queries. Then we collect all the outputs of all these circuits through one threshold gate, setting the threshold to be equal to half of the number of the inputs of the gate. Thus we will (deterministically) accept $\mathcal{S}$ if and only if the corresponding GGR-test would have accepted with probability at least $\frac{1}{2}$.

Clearly we can state and prove for this procedure a (deterministic) replacement for Claim 5.4.

In the original algorithm of Lemma 4.4 there were no other queries made apart from those coming from the constant number of instances of the GGR-test. Given all the acceptance and rejection decisions of the signatures above, whose number is independent of n, we can now find s^* and $\mathcal{S}^*$ as in the algorithm of Lemma 4.4 using an additional constant number of gates. A claim analogous to Claim 5.5 will also work here to ensure that this output is correct.

To find the actual final partition of G, we turn again to the proof in [13] of Lemma 5.2. In addition to the test itself, it is proved in [13] that it is possible with high probability to find a constant query size oracle for placing every vertex of G in its correct set of the partition. In our case we will go over all possible oracles (again there is a polynomial number of such oracles, as the randomized oracle was built in [13] using a constant number of queries to the graph), and for every possibility we use threshold gates to check whether its densities are indeed within the parameters of the corresponding s^* and $\mathcal{S}^*$ (noting that there is only a constant number of possibilities for the values of s^* and $\mathcal{S}^*$). □

Comparing the above theorem to the main result of [1], it is a strengthening both in the types of partitions it can find (finding ϵ-regular partitions through Lemma 4.1), and in the complexity class of the algorithm (TC_0 as compared to NC_1). On the other hand, if we consider the running time of the nonparallel version of the algorithm and are concerned only with regular partitions, then the algorithm of [1] still performs significantly better than the one here.

Robust partitions and variants of regularity. A variant of the regularity lemma that required the existence of both a partition and a regular refinement thereof in the graph G played a central role in [2], [9], and [5]. That variant can also be proved using the notion of robust partitions; in fact, the proof in [2] of the corresponding variant is similar in essence to some of the methods used here for proving Observation 4.2 and Lemma 6.7, so the framework here can be viewed as a generalization of the previous frameworks.

Reducing the number of queries. One can reduce somewhat the number of queries of our test if instead of Lemma 6.7 a more complicated lemma (but with better parameters) about the existence of a partition that is final for both G and G' is proved (rather than starting with a partition $\mathcal{A}$ that is only final for G). However, such an approach would make for a more complicated proof and for a more complicated estimation algorithm that will have to find the parameters for all possible final partitions.

This improvement in the number of queries still would not have made the test practical, since as long as the regularity lemma is used in such a form, the estimation will require a number of queries that is at least a tower in some function of the number of queries of the original testing algorithm. For this reason we aimed here for proof simplicity instead. It would be interesting if this (or any other graph testing result whose only known proof depends on the regularity lemma) can be proved using alternative methods that would provide a saner dependency of the parameters.

REFERENCES

[1] N. Alon, R. A. Duke, H. Lefmann, V. Rödl, and R. Yuster, *The algorithmic aspects of the regularity lemma*, J. Algorithms, 16 (1994), pp. 80–109.

[2] N. Alon, E. Fischer, M. Krivelevich, and M. Szegedy, *Efficient testing of large graphs*, Combinatorica, 20 (2000), pp. 451–476.

[3] N. Alon, E. Fischer, I. Newman, and A. Shapira, *A combinatorial characterization of the testable graph properties: It's all about regularity*, in Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC), 2006, pp. 251–260.

[4] N. Alon and A. Shapira, *Every monotone graph property is testable*, in Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC), 2005, pp. 128–137.

[5] N. Alon and A. Shapira, *A characterization of the (natural) graph properties testable with one-sided error*, in Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2005, pp. 429–438.

[6] N. Alon and J. H. Spencer, *The Probabilistic Method*, 2nd ed., Wiley, New York, 2000.

[7] M. Blum, M. Luby, and R. Rubinfeld, *Self-testing/correcting with applications to numerical problems*, J. Comput. System Sci., 47 (1993), pp. 549–595.

[8] R. Diestel, *Graph Theory*, 2nd ed., Springer-Verlag, New York, 2000.

[9] E. Fischer, *Testing graphs for colorability properties*, Random Structures Algorithms, 26 (2005), pp. 289–309.

[10] E. Fischer, *The art of uninformed decisions: A primer to property testing*, Bull. Eur. Assoc. Theor. Comput. Sci. EATCS, 75 (2001), pp. 97–126.

[11] E. Fischer, *The difficulty of testing for isomorphism against a graph that is given in advance*, SIAM J. Comput., 34 (2005), pp. 1147–1158.

[12] E. Fischer and L. Fortnow, *Tolerant versus intolerant testing for Boolean properties*, in Proceedings of the 20th Annual IEEE Conference on Computational Complexity, 2005, pp. 135–140.

[13] O. Goldreich, S. Goldwasser, and D. Ron, *Property testing and its connection to learning and approximation*, J. ACM, 45 (1998), pp. 653–750.

[14] O. Goldreich and L. Trevisan, *Three theorems regarding testing graph properties*, Random Structures Algorithms, 23 (2003), pp. 23–57.

[15] M. Parnas, D. Ron, and R. Rubinfeld, *Tolerant property testing and distance approximation*, J. Comput. System Sci., 72 (2006) pp. 1012–1042.

[16] D. Ron, *Property testing (a tutorial)*, in Handbook of Randomized Computing, Vol. II, S. Rajasekaran, P. M. Pardalos, J. H. Reif, and J. D. P. Rolim, eds., Kluwer Academic Publishers, Boston, 2001, pp. 597–649.

[17] R. Rubinfeld and M. Sudan, *Robust characterizations of polynomials with applications to program testing*, SIAM J. Comput., 25 (1996), pp. 252–271.

[18] E. Szemerédi, *Regular partitions of graphs*, in Problèmes combinatoires et théorie des graphes, Colloq. Internat. CNRS 260, J. C. Bermond, J. C. Fournier, M. Las Vergnas, and D. Sotteau, eds., CNRS, Paris, 1978, pp. 399–401.

SIAM J. Comput.
Vol. 37, No. 2, pp. 502–521

DEFINABILITY OF LANGUAGES BY GENERALIZED FIRST-ORDER FORMULAS OVER $(\mathbb{N}, +)$*

AMITABHA ROY† AND HOWARD STRAUBING†

Abstract. We consider an extension of first-order logic by modular quantifiers of a fixed modulus q. Drawing on collapse results from finite model theory and techniques of finite semigroup theory, we show that if the only available numerical predicate is addition, then sentences in this logic cannot define the set of bit strings in which the number of 1's is divisible by a prime p that does not divide q. More generally, we completely characterize the regular languages definable in this logic. The corresponding statement, with addition replaced by arbitrary numerical predicates, is equivalent to the conjectured separation of the circuit complexity class ACC from NC^1. Thus our theorem can be viewed as proving a highly uniform version of the conjecture.

Key words. finite model theory, circuit complexity, semigroup theory

AMS subject classifications. 68Q70, 68Q19, 03C98, 68Q17, 20M35

DOI. 10.1137/060658035

1. Background.

The circuit complexity class $ACC(q)$ is the family of languages recognized by constant-depth polynomial-size families of circuits containing unbounded fan-in AND, OR, and MOD_q gates for some fixed modulus $q > 0$. It is known that if q is a prime power and p is a prime that does not divide q, then $ACC(q)$ does not contain the language L_p consisting of all bit strings in which the number of 1's is divisible by p (see Razborov [17] and Smolensky [19]). But for moduli q that have distinct prime divisors, little is known, and the task of separating ACC, the union of the $ACC(q)$, from NC^1 is an outstanding unsolved problem in circuit complexity.

$ACC(q)$ has a model-theoretic characterization as the family of languages definable in an extension of first-order logic which contains predicate symbols for arbitrary relations on the natural numbers, and in which special "modular quantifiers" of modulus q occur along with ordinary quantifiers (see Barrington et al. [3] and Straubing [20]). Since there are languages that are complete for NC^1 under constant-depth reductions, in order to separate NC^1 from ACC, it is sufficient to show that for each $q > 1$ there is a language in NC^1 that does not belong to $ACC(q)$. This suggests that one might be able to attack the problem by model-theoretic means. However, the problem has resisted solution by this or any other method, and little progress has been made since the appearance of Smolensky's work.

Recently, Krebs, Lange, and Reifferscheid [11] raised the possibility of proving the separation for logics with a restricted class of numerical predicates. It is already known (see Straubing, Thérien, and Thomas [21]) that if the only available numerical predicate is $<$, then all the languages definable with ordinary and modular quantifiers of modulus q are regular, and all the groups in the syntactic monoids of these languages are solvable, of cardinality dividing a power of q. This implies, for example, that if q is odd, then one cannot define the set of bit strings with an even number of 1's in this logic. The natural next step is to allow the ternary relation $x + y = z$ on the natural

*Received by the editors April 24, 2006; accepted for publication (in revised form) January 4, 2007; published electronically May 22, 2007.

http://www.siam.org/journals/sicomp/37-2/65803.html

†Computer Science Department, Boston College, Chestnut Hill, MA 02476 (aroy@cs.bc.edu, straubin@cs.bc.edu).

numbers. One can prove the analogue of the separation between AC^0 and NC^1 in this setting by purely model-theoretic means, without recourse to results from circuit complexity (originally proved by Lynch [14]; the question is discussed at length in Barrington et al. [5]). In the present paper we extend this to formulas with ordinary and modular quantifiers over the numerical predicate $x+y=z$. This can be viewed as proving the separation between ACC and NC^1 in a highly uniform setting (recently, a circuit interpretation of this logic was given by Behle and Lange [9]).

We note that natural uniform versions of AC^0 and ACC result when one allows both addition and multiplication as numerical predicates (see Barrington, Immerman, and Straubing [4]). These formulas behave very differently and are much harder to analyze by model-theoretic means. So separating ACC from NC^1 even in this natural uniform setting still appears to be a very difficult problem.

We find it more convenient to first work in the setting of infinite bit strings that contain finitely many 1's. We view such a string as a particularly rudimentary structure (a linearly ordered finite set of 1's) embedded in the natural numbers. We are then faced with the question of how much of the expressive power of the larger structure (in this case, the integers under addition) is needed to express properties of the embedded structure (for instance, that the number of 1's is even). This is precisely the kind of problem considered in the study of "embedded finite models," and we are able to draw upon various collapse results that already appear in the model-theoretic literature. We obtain our result by first showing, in section 3, that it is sufficient to consider sentences that only quantify over positions in a bit string that contain a 1. The underlying quantifier-elimination procedure, while rather complicated in the case of modular quantifiers, is based on an idea that goes back to Presburger [16]. In section 4, we use another model-theoretic collapse, this one based on Ramsey's theorem, to show that it is sufficient to consider sentences in which the only numerical predicate is $<$, which can be analyzed by known semigroup-theoretic methods. Semigroup theory has been used in the past to obtain rather weak lower bounds for computations by circuits and branching programs (see, e.g., Barrington and Straubing [6]). By coupling the algebra in this way with ideas from model theory, we are able to extend its reach.

Nurmonen [15] establishes different nonexpressibility results for sentences with modular quantifiers, using a version of Ehrenfeucht–Fraïssé games. Schweikardt [18] proves nonexpressibility results for logics with different generalized quantifiers over the base $(\mathbb{N}, +)$. Extension of the Ramsey property to generalized quantifiers is discussed in Benedikt and Libkin [10]. We have relied heavily on the account of collapse results for embedded finite models contained in two expository works by Libkin: the survey article [12] and the book [13].

Of course, we are most interested in proving the separation over arbitrary numerical predicates or, at the very least, over a class of numerical predicates that includes both addition and multiplication. In the final section we discuss both the prospects for generalizing the present work, and the obstacles to doing so.

2. Notation and statement of result. We consider first-order logic $FO[+]$ with a single ternary relation $x + y = z$. Formulas are interpreted in the natural numbers $\mathbb{N}$. We adjoin to this logic a single unary relation π. The resulting formulas are interpreted in bit strings, with $\pi(x)$ taken to mean that the bit in position x is 1. In fact we can consider several such interpretations: in finite bit strings ($w \in \{0,1\}^*$), in infinite bit strings ($w \in \{0,1\}^{\mathbb{N}}$), and in infinite bit strings with a finite number of 1's ($w \in \{0,1\}^*0^\omega$, where 0^ω denotes an infinite sequence of 0's). A sentence ϕ in this logic accordingly defines three sets of strings:

$$L_\phi^{fin} = \{w \in \{0,1\}^* : w \models \phi\},$$

$$L_\phi^{\infty} = \{w \in \{0,1\}^{\mathbb{N}} : w \models \phi\},$$

and

$$L_\phi^{fs} = \{w \in \{0,1\}^* 0^\omega : w \models \phi\}.$$

(The letters "*fs*" stand for "finite support.")

For example, let ϕ be the sentence

$$\exists x \exists y ((x = y + y) \wedge \pi(x)),$$

which asserts that there is a 1 in an even-numbered position. Note that for this sentence L_ϕ^{fs} is a proper subset of L_ϕ^∞, and that $L_\phi^{fs} = L_\phi^{fin} 0^\omega$.

We denote this logic by $FO[\pi, +]$. More generally, if $\mathcal{R}$ is any set of relations on $\mathbb{N}$, we denote the analogous logic by $FO[\pi, \mathcal{R}]$. We define the languages L_ϕ^{fin}, etc., in exactly the same way.

To this apparatus we adjoin *modular quantifiers* $\exists^{r \bmod q}$ for a fixed modulus q and $0 \leq r < q$. The interpretation of $\exists^{r \bmod q} x\ \phi$ is, informally, "the number of positions x for which ϕ holds is congruent to r modulo q." More precisely, let $\phi(x, y_1, \ldots, y_k)$ be a formula with free variables $x, y_1, \ldots, y_k$. Let $w \in \{0,1\}^*$ or $w \in \{0,1\}^{\mathbb{N}}$, and let $a_1, \ldots, a_k < |w|$. (If w is infinite, this last condition is automatically satisfied for any natural numbers a_i.) Then we define

$$w \models (\exists^{r \bmod q} x\ \phi)(a_1, \ldots, a_k)$$

iff

$$|\{b < |w| : w \models \phi(b, a_1, \ldots, a_k)\}| \equiv r \pmod{q}.$$

(In particular, for infinite strings w, this implies that the set $\{b < |w| : w \models \phi(b, a_1, \ldots, a_k)\}$ is finite.) For example, the sentence

$$\exists^{0 \bmod 2} x\ \pi(x)$$

defines, in all three interpretations, the set of strings with an even number of 1's.

We denote this logic by $(FO + MOD_q)[\pi, +]$.

Here is our main result. Let $m > 1$, and let L_m denote the set of all finite bit strings in which the number of 1's is divisible by m.

THEOREM 2.1. *If m is a prime that does not divide q, then there is no sentence ϕ in $(FO + MOD_q)[\pi, +]$ such that $L_\phi^{fin} = L_m$ or $L_\phi^\infty = L_m 0^\omega$.*

Remark. If we consider instead the family $\mathcal{N}$ of all relations on $\mathbb{N}$, then the family of languages in $\{0,1\}^*$ definable by sentences in $(FO + MOD_q)(\pi, \mathcal{N})$ is precisely the nonuniform circuit complexity class $ACC(q)$ (see [3, 20]). If we let × denote multiplication in $\mathbb{N}$, then $(FO + MOD_q)[\pi, +, \times]$ is the natural uniform version of $ACC(q)$ (see [4]). For these logics, the analogues of Theorem 2.1 are equivalent to the conjectured separation of $ACC(q)$ and NC^1 in the nonuniform and uniform cases, respectively. Thus our theorem can be thought of as establishing this separation in a highly uniform setting.

In our proof of Theorem 2.1, we will use some notions from the algebraic theory of finite automata: To each regular language $L \subseteq \Sigma^*$ there is associated a finite monoid $M(L)$ (the *syntactic monoid* of L) and a homomorphism $\mu_L : \Sigma^* \to M(L)$ (the *syntactic morphism* of L) such that the value $\mu_L(w)$ determines whether or not $w \in L$. That is, there is a subset X of $M(L)$ such that $L = \mu_L^{-1}(X)$. ($M(L)$ is the *smallest* monoid with this property: It is the monoid of transformations on the states of the minimal automaton of L induced by elements of Σ^*. The homomorphism μ_L maps a word w to the transformation it induces, and X is the set of transformations that take an initial state to an accepting state.)

If $L \subseteq \Sigma^*$ and $\lambda \in \Sigma$, we say λ is a *neutral letter* for L if for any $u, v \in \Sigma^*$, $u\lambda v \in L$ iff $uv \in L$. In other words, deleting or inserting occurrences of λ does not affect a word's membership in L. In the algebraic setting, λ is a neutral letter for L iff $\mu_L(\lambda)$ is the identity of $M(L)$. For example, each of the languages $L_m \subseteq \{0,1\}^*$ defined above has 0 as a neutral letter.

3. Collapse to active-domain formulas. While our goal is to prove a result about definability of sets of finite strings, most of our argument concerns definability of sets of infinite strings. An easy reduction makes the connection between the two models.

LEMMA 3.1. *Let ϕ be a sentence of $(FO+MOD_q)[\pi,+]$ and let $L = L_\phi^{fin}$. Then there is a sentence ϕ' of $(FO+MOD_q)[\pi,+]$ such that*

$$L_{\phi'}^{fs} = L_{\phi'}^{\infty} = L0^\omega.$$

Proof. We define a formula $\phi[\leq x]$ with a single free variable x by rewriting it from the innermost quantifier outward, replacing each instance of

$$\mathcal{Q}z\alpha,$$

where $\mathcal{Q}$ is the quantifier $\exists$ or $\exists^{r \bmod q}$, by

$$\mathcal{Q}z((z \leq x) \wedge \alpha).$$

Then $L0^\omega$ is defined by the sentence

$$\exists x(\forall y(\pi(y) \to y \leq x) \wedge \phi[\leq x]). \qquad \square$$

Remark. Obviously, Lemma 3.1 holds for any of the logics $(FO+MOD_q)[\pi,\mathcal{R}]$ in which $\leq$ is definable.

An *active-domain formula* in $(FO+MOD_q)[\pi,+]$ is one in which every quantifier occurs in the form

$$\mathcal{Q}x(\pi(x) \wedge \alpha),$$

where $\mathcal{Q}$ is either the ordinary existential quantifier or a modular quantifier, and α is a formula. We call these *active-domain quantifiers*. In other words, we allow quantification only over positions that contain the bit 1. Libkin [12] sketches a proof that one can replace every formula in $FO[\pi,+]$ by an equivalent active-domain formula, provided one extends the signature (the *natural-active collapse*). Here we generalize this result to formulas that contain modular quantifiers. (We should add that the collapse to active-domain formulas holds for arbitrary finite structures—for instance, graphs—embedded in $(\mathbb{N},+)$, not just sequences of 1's. One proves, in general, that any formula is equivalent to one in which quantification only ranges over elements of the embedded structure.)

We consider the logic

$$(FO + MOD_q)[\pi, +, <, 0, 1, \{\equiv_s \colon s > 1\}],$$

in which $+$ is now treated as a binary function, 0 and 1 are constants, and $\equiv_s$ is a binary relation symbol denoting congruence modulo s. Of course, all these new constants and relations are definable in $FO[+]$, but we need to include them formally as part of the language in order to carry out the quantifier elimination.

THEOREM 3.2. *Let ϕ be a formula of $(FO + MOD_q)[\pi, +, <, 0, 1, \{\equiv_s \colon s > 1\}]$, with free variables in $\{x_1, \ldots, x_r\}$. Then there is an active-domain formula ψ in the same logic such that for all $w \in \{0,1\}^*0^\omega$ and $a_1, \ldots, a_r \in \mathbb{N}$, we have $w \models \phi(a_1, \ldots, a_r)$ iff $w \models \psi(a_1, \ldots, a_r)$.*

Proof. The proof is by induction on the construction of ϕ. There is nothing to prove in the base case of quantifier-free formulas. For the inductive step, we assume

$$\phi = \mathcal{Q}z\ \phi', \tag{3.1}$$

where $\mathcal{Q}$ is either an existential quantifier ($\exists$) or a modular quantifier ($\exists^{k \bmod q}$) and ϕ' is a formula such that any quantifier appearing in ϕ' is an active-domain quantifier. We assume that ϕ' has free variables $x_1, x_2, \ldots, x_r$ and bound variables (hence active-domain variables) $y_1, y_2, \ldots, y_s$.

Notation. We shall write $\hat{\mathbf{v}}^m$ to denote the tuple $(v_1, v_2, \ldots, v_m)$. When m is obvious from the context or is irrelevant, we simply write $\hat{\mathbf{v}}$ and refer to the ith coordinate as $\hat{\mathbf{v}}_i$.

Terms in our logic are expressions of the form

$$a_0 + a_1 w_1 + \cdots + a_k w_k,$$

where the a_i are in $\mathbb{N}$ and the w_i are variables. Atomic formulas have the form

$$\sigma = \tau, \quad \sigma < \tau, \quad \sigma > \tau, \quad \sigma \equiv_m \tau, \quad \pi(\sigma),$$

where σ, τ are terms. We can eliminate atomic formulas of the form $\pi(\sigma)$ by introducing a new active-domain variable y and replacing the atomic formula by

$$\exists y(\pi(y) \wedge y = \sigma).$$

We can rewrite each atomic formula $\sigma = \tau$ in ϕ that involves z as $nz = \rho$, where ρ does not involve z. Strictly speaking, ρ is not a term in our logic, since we do not have subtraction available, so this must be regarded as a shorthand for $nz + \rho_1 = \rho_2$, where ρ_1, ρ_2 are genuine terms that do not involve z. Later we will view the expression ρ as defining a partial function on $\mathbb{N}^{r+s}$. Similarly, we rewrite other atomic formulas involving z as

$$nz < \rho, \quad nz > \rho, \quad nz \equiv_m \rho,$$

where ρ does not involve z.

Let l be the least common multiple of the coefficients of z in these atomic formulas. Then since

$$\begin{aligned} nz = \rho \quad &\text{iff } lz = (l/n)\rho, \\ nz < \rho \quad &\text{iff } lz < (l/n)\rho, \\ nz \equiv_m \rho \quad &\text{iff } lz \equiv_{m(l/n)} (l/n)\rho \end{aligned}$$

we can suppose that z always appears with the same coefficient l in every atomic subformula of ϕ'.

Making a change of variable $z \mapsto lz$, we see that ϕ is equivalent to the following formula:

$$\mathcal{Q}\, z\ (z \equiv_l 0 \wedge \phi'),$$

where if z occurs in an atomic formula, it occurs with coefficient 1, and where each such formula has the form $z = \rho$, $z < \rho$, $z > \rho$, $z \equiv_m \rho$, where ρ does not involve z.

Atomic formulas in ϕ' of the form $z \equiv_m \rho$ can be replaced by

$$\bigvee_{i=0}^{m-1} (z \equiv_m i \wedge \rho \equiv_m i),$$

so we may suppose that in every such atomic formula ρ is a constant in $\mathbb{N}$. Let l' be the least common multiple of the moduli occurring in such atomic formulas. Then ϕ is equivalent to

$$\mathcal{Q}\, z \bigvee_{j=0}^{l'-1} \Big[z \equiv_{l'} j \quad \wedge \phi'_j \Big], \tag{3.2}$$

where ϕ'_j is the formula obtained from ϕ' upon replacing each congruence $z \equiv_m i$ by **true** or **false**, depending on whether this is consistent with $z \equiv_{l'} j$.

If $\mathcal{Q} = \exists$ in (3.2), then we can rewrite it as

$$\bigvee_{j=0}^{l'-1} \exists z \Big[z \equiv_{l'} j \quad \wedge \phi'_j \Big]. \tag{3.3}$$

Suppose $\mathcal{Q} = \exists^{k \bmod q}$. Observe that if $\alpha_1, \ldots, \alpha_t$ are pairwise mutually exclusive, then we can rewrite

$$\exists^{k \bmod q} z \bigvee_{i=1}^{t} \alpha_i$$

as

$$\bigvee \bigwedge_{i=1}^{t} \exists^{k_i \bmod q} z\ \alpha_i,$$

where the disjunction is over all t-tuples $(k_1, \ldots, k_t) \in \mathbb{Z}_q^t$ for which $\sum_{i=1}^{t} k_i = k$. Thus we can rewrite (3.2) as a boolean combination of formulas of the form

$$\exists^{k' \bmod q} z \Big[z \equiv_{l'} j \quad \wedge \phi'_j \Big].$$

We can thus assume that ϕ has the form

$$\mathcal{Q}\, z\ \Big((z \equiv_d c) \wedge \phi' \Big),$$

where $\mathcal{Q}$ is an ordinary existential or ordinary modular quantifier and ϕ' is an active-domain formula in which every atomic formula involving z is either of the form $z < \rho$, $z = \rho$, or $z > \rho$.

We now fix an instantiation of $\hat{\mathbf{x}}^r$, the free variables of ϕ, by a tuple $\hat{\mathbf{a}}^r \in \mathbb{N}^r$. To simplify the notation, we will not make explicit reference to $\hat{\mathbf{a}}^r$ in the remainder of the proof. Each ρ appearing on the right-hand side of one of our atomic formulas accordingly defines a partial function g from $\mathbb{N}^s$ into $\mathbb{N}$, where s is the number of active-domain variables. We set $\rho(t_1, t_2, \ldots, t_s)$ to be the value obtained by substituting $t_i \in \mathbb{N}$ for the variable y_i, $1 \leq i \leq s$, in ρ if this value is nonnegative; $\rho(t_1, \ldots, t_s)$ is undefined otherwise. We let $\{g_i : i \in I\}$ denote the set of these partial functions.

Let $w \in \{0,1\}^* 0^\omega$, and let $D \subseteq \mathbb{N}$ denote the set of positions in w that contain 1's. (That is, D is the active domain of w.) Let

$$\mathcal{B} = \bigcup_{i \in I} \{g_i(\hat{\mathbf{y}}) | \hat{\mathbf{y}} \in D^s\}.$$

Write $\mathcal{B}$ as an ordered set $\{b_0, b_1, \ldots, b_{p-1}\}$, where $b_0 < b_1 < b_2 < \cdots < b_{p-1}$. We denote by (a, b) the set $\{x \in \mathbb{N} : a < x < b\}$. By an *interval* in $\mathcal{B}$, we will mean either the leftmost interval $(-1, b_0)$, intervals of the form (b_i, b_{i+1}) for $0 \leq i \leq p-2$, or the rightmost interval (b_{p-1}, ∞).

LEMMA 3.3. *If there exists an integer z_0 in an interval in $\mathcal{B}$ such that*

$$w \models \phi'(z_0),$$

then

$$w \models \phi'(z_0')$$

for every z_0' in the interval. (That is, if an interval contains a witness, then every point in the interval is a witness.)

Proof. The proof is by induction on the construction of ϕ'. We will show that for every subformula ψ of ϕ' and every instantiation $\hat{\mathbf{d}}$ of the free active-domain variables by a tuple over D, $w \models \psi(z_0, \hat{\mathbf{d}})$ implies $w \models \psi(z_0', \hat{\mathbf{d}})$.

Since all atomic formulas of ϕ' that involve z have one of the forms $z < g_j(\hat{\mathbf{y}})$, $z = g_j(\hat{\mathbf{y}})$, or $z > g_j(\hat{\mathbf{y}})$ for some $j \in I$, and since $g_j(\hat{\mathbf{d}}) \in \mathcal{B}$ for all tuples $\hat{\mathbf{d}}$ over D, the claim holds for the atomic subformulas of ϕ'. The property clearly is preserved under boolean operations. Now suppose that the property holds for some subformula ψ of ϕ', and that $y_1, \ldots, y_j$ are the free active-domain variables in ψ. Our hypothesis applied to ψ implies that if z_0 and z_0' belong to the same interval of $\mathcal{B}$, then

$$\{\hat{\mathbf{d}} \in D^j : w \models \psi(z_0, \hat{\mathbf{d}})\} = \{\hat{\mathbf{d}} \in D^j : w \models \psi(z_0', \hat{\mathbf{d}})\}.$$

In particular, for each fixed $d_2, \ldots, d_j \in D$,

$$\{d_1 \in D : w \models \psi(z_0, \hat{\mathbf{d}})\} = \{d_1 \in D : w \models \psi(z_0', \hat{\mathbf{d}})\},$$

so, in particular, these two sets have the same cardinality. Thus if $\mathcal{Q}$ is either an existential or modular quantifier,

$$w \models \mathcal{Q} y_1(\pi(y_1) \wedge \psi(z_0, d_2, \ldots, d_j))$$

iff

$$w \models \mathcal{Q} y_1(\pi(y_1) \wedge \psi(z_0', d_2, \ldots, d_j)).$$

Thus the property is preserved under active-domain quantification. $\square$

We define the function $\chi_{c,d} : \mathbb{Z} \to \mathbb{Z}$ as follows:

$$\chi_{c,d}(\alpha) = \begin{cases} (c-\alpha) \bmod d & \text{if } \alpha \not\equiv_d c, \\ d & \text{otherwise.} \end{cases}$$

Corollary 3.4. *Let (l, r) be an interval in $\mathcal{B}$ such that $l \equiv_{dq} \alpha$. Then*

$$w \models \{(z_0 \equiv_d c) \wedge \phi'(z_0)\}$$

for some $z_0 \in (l, r)$ iff

$$l + \chi_{c,d}(\alpha) < r \quad \textit{and} \quad w \models \phi'(l + \chi_{c,d}(\alpha)).$$

Proof. Lemma 3.3 implies that if there is a witness at all in the interval (l, r), then *any* integer z_0 in the interval such that $z_0 \equiv_d c$ would be a witness. The integer $l + (c - \alpha) \bmod d$ satisfies this requirement if $c \not\equiv_d \alpha$. If $c \equiv_d \alpha$, then the integer $l + d$ satisfies the requirement. □

Remark. We count witnesses in two iterations: First, we count the number modulo q of witnesses z (if they exist) strictly contained in intervals (l, r), where $l < z < r$ and l, r are successive points in $\mathcal{B}$, and then we *separately* count points of $\mathcal{B}$ which are themselves witnesses. As a result, we need to distinguish the cases $c \equiv l \bmod d$ and $c \not\equiv l \bmod d$ in our formulas. The function $\chi_{c,d}$ enables us to distinguish between the two cases.

We also have a special property concerning the infinite interval (b_{p-1}, ∞), as follows.

Corollary 3.5. *Let $b_{p-1} \equiv_{dq} \alpha$. If*

$$w \models \exists^{k \bmod q} z \ \{(z \equiv_d c) \wedge \phi'\},$$

then

$$w \not\models \phi'(b_{p-1} + \chi_{c,d}(\alpha)).$$

Proof. If

$$w \models \phi'(b_{p-1} + \chi_{c,d}(\alpha)),$$

then Lemma 3.3 implies that every $z_0 \in (b_{p-1}, \infty)$ such that $z_0 \equiv_d c$ would be a witness. However,

$$w \models \exists^{k \bmod q} z \ \{(z \equiv_d c) \wedge \phi'\}$$

implies that there are only a finite number of witnesses. □

We also note the following fact.

Lemma 3.6. *Let $l, r \in \mathbb{N}$, where $l \leq r$, and let $c, d, q, \alpha, \beta \in \mathbb{N}$ be such that*

$$l \equiv \alpha \bmod dq \quad \textit{and} \quad r \equiv \beta \bmod dq.$$

Let $\eta_q(\alpha, \beta)$ denote the number modulo q of integers x in (l, r) such that $x \equiv_d c$. Then $\eta_q(\alpha, \beta)$ depends only on α, β, c, d, q.

Proof. Since the number mod q of points $x \equiv_d c$ in the interval (l, r) does not change under the maps $r \mapsto r + adq$, $l \mapsto l + bdq$ (where $a, b \in \mathbb{Z}$), $\eta_q(\alpha, \beta)$ is independent of the actual values of l and r. □

Remark 3.1. An explicit formula for $\eta_q(\alpha,\beta)$ is

$$\eta_q(\alpha,\beta) \equiv 1 + \frac{\beta - \alpha - (c-\alpha) \bmod d - (\beta - c) \bmod d}{d} - \delta \pmod{q},$$

where

$$\delta = \begin{cases} 2 & \text{if } \alpha \equiv_d c \text{ and } \beta \equiv_d c, \\ 1 & \text{exactly one of } \alpha \text{ or } \beta \text{ is } \equiv_d c, \\ 0 & \text{otherwise.} \end{cases}$$

However, the point of Lemma 3.6 is that $\eta_q(\alpha,\beta)$ depends only on the constants α, β, c, d, q, and so wherever it appears in our formulas, say, in the form $\eta_q(\alpha,\beta) \equiv_q \gamma$ (see, e.g., the formula $\mathrm{CountZero}(x,y)$ below), we can replace this by true or false. This renders the exact form of the expression $\eta_q(\alpha,\beta)$ irrelevant.

We now proceed to the quantifier elimination by building an active-domain formula equivalent to $\phi = \exists^{k \bmod q} z((z \equiv_d c) \wedge \phi'(z))$. The idea is to write a formula that counts, modulo q, the number of witnesses to $(z \equiv_d c) \wedge \phi'(z)$ in each interval of $\mathcal{B}$. At each step of the argument we show how to express some property of w in our language. Our initial result will be a formula in which the arbitrary quantifier is replaced by quantification over elements of $\mathcal{B}$, but in the end we will show how to rewrite these in terms of active-domain quantifiers.

Membership in $\mathcal{B}$: The formula $\mathrm{Member}(x)$ asserts that $x \in \mathcal{B}$:

$$\exists^a \hat{\mathbf{y}} \bigvee_{i \in I} (g_i(\hat{\mathbf{y}}) = x),$$

where $\exists^a \hat{\mathbf{y}}\ \alpha$ is an abbreviation for

$$\exists y_1(\pi(y_1) \wedge \exists y_2(\pi(y_2) \wedge \cdots \exists y_s(\pi(y_s) \wedge \alpha) \cdots)).$$

(x,y) is an interval: The formula $I(x,y)$ asserts that x and y are successive elements of $\mathcal{B}$:

$$\begin{aligned} &(x<y) \wedge \mathrm{Member}(x) \wedge \mathrm{Member}(y) \\ &\wedge \neg \exists z\ \big(\mathrm{Member}(z) \wedge \big\{(x<z) \vee (z<y)\big\}\big). \end{aligned} \tag{3.4}$$

The interval (x,y) in $\mathcal{B}$ has $0 \bmod q$ *witnesses*: This is expressed by the sentence $\mathrm{InteriorPointCountZero}(x,y)$:

$$I(x,y) \wedge \mathrm{CountZero}(x,y),$$

where *CountZero*(x,y) is

$$\begin{aligned} &\bigvee_{\substack{0 \le \alpha \le dq-1 \\ 0 \le \beta \le dq-1}} \Big[(x \equiv_{dq} \alpha) \wedge (y \equiv_{dq} \beta) \\ &\wedge \Big\{(x + \chi_{c,d}(\alpha) < y) \implies \Big(\neg \phi'(x + \chi_{c,d}(\alpha)) \vee \eta_q(\alpha,\beta) \equiv_q 0\Big)\Big\}\Big]. \end{aligned}$$

Remark 3.2. Since the function $\chi_{c,d}(\alpha)$ depends only on the constants c, d, and α, we can substitute the value of $\chi_{c,d}(\alpha)$ wherever it appears in our formulas, for example, in the formula for $\mathrm{CountZero}(x,y)$ above. Thus it is not necessary to express $\chi_{c,d}(\alpha)$ in terms of a boolean formula. A similar comment holds for $\eta_q(\alpha, \beta)$ (see Remark 3.1).

Interval (x, y) *in* $\mathcal{B}$ *contains* $\gamma \bmod q$ *witnesses, where* $\gamma \not\equiv_q 0$: This is expressed by the sentence InteriorPointCountNonZero(x, y, γ):

$$I(x, y) \wedge \text{CountNonZero}(x, y, \gamma),$$

where *CountNonZero*(x, y, γ) is

$$\bigvee_{\substack{0\le\alpha\le dq-1\\0\le\beta\le dq-1}} \Big[(x \equiv_{dq} \alpha) \wedge (y \equiv_{dq} \beta)$$
$$\wedge\ \ (x + \chi_{c,d}(\alpha) < y)\ \ \wedge\ \ \phi'(x + \chi_{c,d}(\alpha)) \wedge \eta_q(\alpha, \beta) \equiv_q \gamma\Big].$$

Interval (x, y) *in* $\mathcal{B}$ *contains* $\gamma \bmod q$ *witnesses*: This is expressed by the sentence InteriorPointCount(x, y, γ):

$$\begin{aligned}&(\gamma \equiv_q 0 \implies \text{InteriorPointCountZero}(x, y))\\ \wedge\, &(\gamma \not\equiv_q 0 \implies \text{InteriorPointCountNonZero}(x, y, \gamma)).\end{aligned}$$

Minimum and maximum elements of $\mathcal{B}$: The formula for Min(x) is

$$\text{Member}(x) \wedge \neg\exists y(\text{Member}(y) \wedge y < x).$$

We define Max(x) similarly.

The leftmost interval contains $\gamma \bmod q$ *witnesses*: The formula $W(\gamma)$ given by

$$\begin{aligned}\exists x\ \Big[\text{Min}(x) \wedge \Big\{&(\gamma \equiv_q 0) \implies \text{CountZero}(0, x)\Big\}\\ \wedge \Big\{&(\gamma \not\equiv_q 0) \implies \text{CountNonZero}(0, x, \gamma)\Big\}\Big]\end{aligned}$$

says that the interval $(0, b_0)$ contains $\gamma \bmod q$ witnesses. We have to modify this depending on whether or not 0 is itself a witness. Thus if $c \neq 0$, we set $\mathcal{C}_L(\gamma)$ to be $W(\gamma)$; otherwise, we set it to be $\phi'(0) \wedge W(\gamma - 1)$.

The rightmost interval contains no witnesses: This is expressed by $\mathcal{C}_R$:

$$\exists\, x\ \left\{\text{Max}(x) \wedge \bigwedge_{0\le\alpha\le dq-1} \{(x \equiv_{dq} \alpha) \to \neg\phi'(x + \chi_{c,d}(\alpha))\}\right\}.$$

Number $\bmod q$ *of intervals* (b_i, b_{i+1}) *containing* $\gamma \bmod q$ *witnesses*: The sentence $H(\delta, \gamma)$ asserts that there are $\delta \bmod q$ intervals (x, y) with endpoints in $\mathcal{B}$ having $\gamma \bmod q$ witnesses:

$$\text{H}(\delta, \gamma) = \exists^{\delta \bmod q}\, x\ \exists\, y\ \text{InteriorPointCount}(x, y, \gamma).$$

Number $\bmod q$ *of witnesses from intervals* (b_i, b_{i+1}): The formula $\mathcal{C}_{\text{int}}(\gamma)$ asserts that the number of witnesses contained in intervals (b_i, b_{i+1}), where $b_i, b_{i+1} \in \mathcal{B}$, is congruent to $\gamma \bmod q$:

$$\bigvee_{\substack{0\le\gamma_j\le q-1\\0\le j\le q-1\\\sum_{j=0}^{q-1} j\gamma_j\equiv\gamma \bmod q}} \bigwedge_{i=0}^{q-1} H(i, \gamma_i).$$

Number mod q *of witnesses from* $\mathcal{B}$: The sentence $\mathcal{C}_{\mathcal{B}}(\gamma)$ asserts that the number of witnesses $b_i \in \mathcal{B}$ is congruent to γ mod q:

$$\exists^{\gamma \bmod q}\, l\ (\mathrm{Member}(l) \wedge (l \equiv_d c) \wedge \phi'(l)).$$

Total number mod q *of witnesses*: The sentence $\mathcal{C}_{\mathrm{tot}}(\gamma)$ asserts that the total number of witnesses is congruent to γ modulo q:

$$\bigvee_{\substack{0 \le \gamma_1, \gamma_2, \gamma_3 \le q-1 \\ \gamma_1+\gamma_2+\gamma_3 \equiv_q \gamma}} (\mathcal{C}_{\mathcal{B}}(\gamma_1) \wedge \mathcal{C}_L(\gamma_2) \wedge \mathcal{C}_{\mathrm{int}}(\gamma_3)).$$

Thus $\exists^{k \bmod q} z\big\{(z \equiv_d c) \wedge \phi(z)\big\}$ is equivalent to the sentence

$$\mathcal{T} = \mathcal{C}_{\mathrm{tot}}(k) \wedge \mathcal{C}_R.$$

Note that $\mathcal{T}$ is *almost* active-domain. The non–active-domain quantifiers in $\mathcal{T}$ are of the form

$$\exists\, x\ \{\mathrm{Member}(x) \wedge \mathcal{T}'(x)\}\ \text{ or of the form }\ \exists^{k \bmod q}\, x\ \{\mathrm{Member}(x) \wedge \mathcal{T}'(x)\}.$$

In the first case, we can replace the ordinary existential quantifier in front of x by the definition of Member(x) to get an active-domain formula of the form

$$\exists^a \hat{\mathbf{y}} \bigvee_{i \in I} \mathcal{T}'(g_i(\hat{\mathbf{y}})).$$

Rewriting the second formula with active-domain quantifiers is more complicated. Let $g_1, \ldots, g_m$ be the partial functions, and let $\mathcal{B}_i$ be the set of points in $g_i(D^s)$ that are not in $g_j(D^s)$ for any $j > i$. Since $\mathcal{B}$ is the disjoint union of the $\mathcal{B}_i$, we can rewrite

$$\exists^{k \bmod q}\, x\ \{\mathrm{Member}(x) \wedge \mathcal{T}'(x)\}$$

as a boolean combination of sentences of the form

$$\exists^{k' \bmod q}\, x\ \{\mathrm{Member}_j(x) \wedge \mathcal{T}'(x)\}, \tag{3.5}$$

where Member$_j(x)$ asserts that x belongs to $\mathcal{B}_j$. It is easy enough writing an active-domain formula that asserts that x is in $\mathcal{B}_j$, but how do we count the number of elements in $\mathcal{B}_j$ with a given property?

Let $\prec$ denote the lexicographic ordering on D^s. We can express $\hat{\mathbf{y}} \prec \hat{\mathbf{y}}'$ as a boolean combination of the formulas $y_i < y_i'$ and $y_i = y_i'$. Let $LL_i(\hat{\mathbf{y}})$ be the formula

$$\neg\exists^a \hat{\mathbf{y}}'((g_i(\hat{\mathbf{y}}) = g_i(\hat{\mathbf{y}}')) \wedge (\hat{\mathbf{y}} \prec \hat{\mathbf{y}}')).$$

This asserts that $\hat{\mathbf{y}}$ is the lexicographically maximal s-tuple yielding the value $g_i(\hat{\mathbf{y}})$ under g_i. (Implicit in this is the assertion that $g_i(\hat{\mathbf{y}})$ is defined, which is expressed by a simple inequality.) We can thus rewrite our formula (3.5) as

$$\exists^{k' \bmod q}(\hat{\mathbf{y}} \in D^s) \left(LL_j(y) \wedge \mathcal{T}'(g_j(\hat{y})) \wedge \neg\exists \hat{\mathbf{y}}' \bigvee_{i > j} (g_i(\hat{\mathbf{y}}') = g_j(\hat{\mathbf{y}})) \right).$$

Finally, we note that modular quantification over s-tuples of elements of D is expressible in terms of modular quantification over active-domain elements. Indeed,

$$\exists^{k \bmod q}(y_1, y_2)\ \alpha$$

is equivalent to the disjunction of

$$\bigwedge_{i=0}^{q-1} \exists^{i \bmod q} y_1\ \exists^{f(i) \bmod q} y_2\ \alpha \tag{3.6}$$

over all functions f from $\mathbb{Z}_q$ to itself such that $\sum_{i=0}^{q-1} if(i) = k$, and we can extend this inductively to quantification over tuples of arbitrary size.

We have said nothing about how to eliminate ordinary non–active-domain quantifiers. This case is treated in Libkin [12], which was the starting point for the present proof. The argument follows the same pattern, but is much simpler, since we do not need to count either points in the images of the g_i or points in their domains. We merely have to assert that there exists some $u \in \mathcal{B}$ such that

$$\left\{ \bigvee_{\substack{0 \le e \le d-1 \\ u+e \ge 0 \\ u+e \equiv_d c}} \phi'(u+e) \right\} \vee \left\{ \bigvee_{\substack{0 \le e \le d-1 \\ u-e \ge 0 \\ u-e \equiv_d c}} \phi'(u-e) \right\}$$

holds, and this is easily carried out using the Member formula introduced earlier. □

4. Collapse to formulas with $<$ as the only numerical predicate.

4.1. Ramsey property. Our discussion here closely parallels that of Libkin [13]. Let $\mathcal{R}$ be any set of relations on $\mathbb{N}$, and let $\phi(x_1, \ldots, x_k)$ be an active-domain formula in $(FO + MOD_q)[\pi, \mathcal{R}]$. We say that ϕ has the *Ramsey property* if for each infinite subset X of $\mathbb{N}$ there exists an infinite subset Y of X and an active-domain formula $\psi(x_1, \ldots, x_k)$ in $(FO + MOD_q)[\pi, <]$ that satisfies the following condition: If $w \in \{0,1\}^* 0^\omega$ and all the 1's in w are in positions belonging to Y, then for all $a_1, \ldots, a_k \in Y$,

$$w \models \phi(a_1, \ldots, a_k) \text{ iff } w \models \psi(a_1, \ldots, a_k).$$

Lemma 4.1. *Let $\mathcal{N}$ be the set of all relations on $\mathbb{N}$. Every active-domain formula in $(FO + MOD_q)[\pi, \mathcal{N}]$ has the Ramsey property.*

Proof. The Ramsey property for an assortment of generalized quantifiers is proved by Benedikt and Libkin [10] (also in [13, Lemma 13.15, p. 259]) by using induction on the quantifier depth. While they do not explicitly consider the modular quantifiers that we use here, there is no essential change in the proof. For clarity, we include the inductive step for modular quantifiers.

Let $\phi(\hat{\mathbf{x}}) = \exists^{k \bmod q} y\ [\pi(y) \wedge \phi_1(y, \hat{\mathbf{x}}^r)]$ be an active-domain formula in $(FO + MOD_q)[\pi, \mathcal{N}]$. By the induction hypothesis (from Lemma 13.15 in [13]), for each infinite subset X of $\mathbb{N}$ there exists an infinite subset Y of X and an active-domain formula $\psi_1(y, \hat{\mathbf{x}})$ in $(FO + MOD_q)[\pi, <]$ such that if $w \in \{0,1\}^* 0^\omega$ and all the 1's in w are in positions belonging to Y, then for all $\hat{\mathbf{a}}^r \in Y^r$ and $b \in Y$, $w \models \psi_1(b, \hat{\mathbf{a}})$ iff $w \models \phi_1(b, \hat{\mathbf{a}})$. This implies that for every $\hat{\mathbf{a}}$,

$$\{b \in Y | w \models \psi_1(b, \hat{\mathbf{a}})\} = \{b \in Y |\ w \models \phi_1(b, \hat{\mathbf{a}})\}.$$

Let $\psi(\hat{\mathbf{x}}) = \exists^{k \bmod q} y \; [\pi(y) \wedge \psi_1(y, \hat{\mathbf{x}})]$. Then for every w such that its 1's are in Y and $\hat{\mathbf{a}} \in Y^r$, $w \models \psi(\hat{\mathbf{a}})$ iff

$$|\{b \in Y | w \models \psi_1(b, \hat{\mathbf{a}})\}| \equiv_q k.$$

This happens iff

$$|\{b \in Y | w \models \phi_1(b, \hat{\mathbf{a}}\}| \equiv_q k$$

since the two sets are identical. Thus $w \models \psi(\hat{\mathbf{a}})$ iff $w \models \phi(\hat{\mathbf{a}})$. □

The Ramsey property allows us to capture a subset of a language expressible by a formula ϕ (which satisfies the Ramsey property) using a new formula over a very limited vocabulary (the only numerical predicate allowed is $<$). This limited vocabulary restricts the kind of language that can be expressed.

LEMMA 4.2. *Let $L_\psi = \{w | \, w \in \{0,1\}^*\}$ be the set of finite bit strings defined by an active-domain sentence $\psi \in (FO + MOD_q)[\pi, <]$.*

(i) *The language L_ψ is regular. Moreover, the syntactic monoid $M(L_\psi)$ contains only solvable groups whose order divides a power of q.*

(ii) *L_ψ has 0 as a neutral letter.*

(iii) *Let $z \in \Sigma^*$. Then $z \in L_\psi$ iff $z0^\omega \models \psi$.*

Proof. Condition (i) is a result of Straubing, Thérien, and Thomas [21]. Inserting or deleting 0's from any string satisfying ψ does not alter the truth value of any atomic formula of the form $x < y$, provided the variables represent positions containing 1, which is the case here, since ψ is active-domain. Conditions (ii) and (iii) then follow by an easy induction on the quantifier depth. □

4.2. Proof of Theorem 2.1. Let m be a prime that does not divide q, and suppose, contrary to the claim in the theorem, that L_m is defined by a sentence ϕ of $(FO + MOD_q)[\pi, +]$. By Lemma 3.1, Theorem 3.2, and Lemma 4.1, there exists an active-domain sentence ψ of $(FO + MOD_q)[\pi, <]$ and an infinite subset Y of $\mathbb{N}$ such that for all $w \in \{0,1\}^*0^\omega$ in which all 1's are in positions belonging to Y, $w \models \psi$ iff $w \in L_m 0^\omega$. Let L_ψ denote the set of *finite* bit strings that satisfy ψ. We prove the following lemma.

LEMMA 4.3. *$L_m = L_\psi$.*

Proof. We first show that $L_\psi \subseteq L_m$. Let $z' \in L_\psi$. We pad z' with 0's so that the 1's in the new padded string z'' appear in positions included in the set Y. Since $z'' \in L_\psi$ (by Lemma 4.2 (ii)), we conclude that $z''0^\omega \models \psi$ (by Lemma 4.2 (iii)). Since the 1's in $z''0^\omega$ appear in positions in Y, $z''0^\omega \models \phi$. Hence $z''0^\omega \in L_m 0^\omega$, so $z'' \in L_m$. Removing additional neutral letter 0's introduced while padding z', we conclude that $z' \in L_m$.

The opposite inclusion ($L_m \subseteq L_\psi$) is proved by reversing each step above. □

Since the syntactic monoid of L_m is the cyclic group Z_m and that of L_ψ has groups of order dividing a power of q (via Lemma 4.2), we have a contradiction since $(m, q) = 1$. Thus L_m cannot be defined by a sentence in $(FO + MOD_q)[\pi, +]$. This completes the proof.

4.3. Other nondefinability results. Here we show how to extend Theorem 2.1 to prove nonexpressibility results for other languages. We begin by removing the restriction to binary alphabets.

Let Σ be a finite alphabet and let us consider languages definable in the logic $\mathcal{L}_{q,\Sigma,+} = (FO + MOD_q)[\{\pi_\sigma : \sigma \in \Sigma\}, +]$, where each π_σ is a unary predicate: $\pi_\sigma x$ is interpreted to mean that the letter in position x is σ. We designate a special letter

$\lambda \in \Sigma$, and say that a formula is active-domain (with respect to λ) if every existential and modular quantifier $\mathcal{Q}$ occurs in the form $\mathcal{Q}x((\vee_{\sigma\neq\lambda}\pi_\sigma x) \wedge \alpha)$. Note that we need never use the atomic formula $\pi_\lambda x$, even in non–active-domain formulas, as it is equivalent to the conjunction of the $\neg\pi_\sigma x$ over all letters σ not equal to λ. All the preceding results hold in this broader setting, with no changes to their proofs. We thus have the following theorem.

THEOREM 4.4. *Let $L \subseteq \Sigma^*$, with $\lambda \in \Sigma$ a neutral letter for L. If L is definable in $\mathcal{L}_{q,\Sigma,+}$, then it is definable by a sentence of $(FO + MOD_q)[\{\pi_\sigma : \sigma \in \Sigma\}, <]$. In particular, L is regular, and every group in $M(L)$ is solvable, with cardinality dividing a power of q.*

The foregoing theorem allows us to give an effective characterization of all the *regular* languages in $\mathcal{L}_{q,\Sigma,+}$.

THEOREM 4.5. *Let $L \subseteq \Sigma^*$ be regular. L is definable in $\mathcal{L}_{q,\Sigma,+}$ iff for all $t > 0$ every group in $\mu_L(\Sigma^t)$ is solvable and has cardinality dividing a power of q.*

The reduction to the neutral letter case is somewhat involved, so we delegate the proof to the next section. The same property is known to characterize the regular languages in $ACC(q)$, provided that the conjectured separation of $ACC(q)$ and NC^1 holds [3].

Since L is regular, there exist integers k and l such that $\mu_L(\Sigma^{k+l}) = \mu_L(\Sigma^k)$ (since $\mu_L(\Sigma^t) \subseteq M(L)$ for all $t \geq 0$ and $M(L)$ is finite). Thus we can effectively enumerate all the sets $\mu_L(\Sigma^t)$ and all their subgroups. We thus have the following result.

COROLLARY 4.6. *Given an integer $q > 1$ and a regular language $L \subseteq \Sigma^*$, the question of whether L is definable in $\mathcal{L}_{q,\Sigma,+}$ is decidable.*

Here is an application of Theorem 4.5. Let G be a finite group and let $\Sigma \subseteq G$ be a set of generators of G. We treat G as a finite alphabet; to each word $w \in \Sigma^*$ we assign the group element $\phi(w)$ that results by multiplying together the letters of w. The *word problem for G* (with respect to Σ) is the language $\{w \in \Sigma^* : \phi(w) = 1\}$. Barrington [2] showed that the word problem for any finite nonsolvable group is complete for NC^1 with respect to constant-depth reductions, so that the conjectured separation of ACC from NC^1 is equivalent to the assertion that no such word problem belongs to ACC. We can verify directly that no such word problem L is definable in $\mathcal{L}_{q,\Sigma,+}$: L is a regular language, and it is easy to check that $M(L) = G$ and $\mu_L = \phi$. If G is nonsolvable, then its commutator subgroup G' is also nonsolvable, and thus every element of G' is the image of a word over Σ of length divisible by $|G|$ (each commutator is an image of a word of the form $uvu^{|G|-1}v^{|G|-1}$, where $u, v \in \Sigma$). We can pad each of these words with a sufficient number of copies of $\sigma^{|G|}$ (for some fixed $\sigma \in \Sigma$) so that they all have the same length t. Thus $G' \subseteq \phi(\Sigma^t)$. Since G' is nonsolvable, Theorem 4.5 now implies that L is not definable is $\mathcal{L}_{q,\Sigma,+}$.

THEOREM 4.7. *No word problem of a finite nonsolvable group is definable in any $\mathcal{L}_{q,\Sigma,+}$.*

Note that it is precisely the nonsolvability of G, rather than the relation between $|G|$ and q, that is at issue here: For instance, a word problem of the alternating group of degree 5, whose cardinality is 60, is not definable in $\mathcal{L}_{30,\Sigma,+}$ even though the cardinality and modulus are consistent. On the other hand, the word problem for any solvable group of order 60 is definable in this logic.

5. Proof of Theorem 4.5.

5.1. Two essential lemmas. Let Σ be a finite alphabet. We prove that definability in $(FO + MOD_q)[\{\pi_\sigma : \sigma \in \Sigma\}, +, <]$ (which we denote by $\mathcal{L}$ for the rest of this section) is preserved under inverse length-multiplying morphisms and quotients.

Remark. Note that we are admitting $x < y$ as an atomic formula, rather than simply defining it in terms of $+$. This is largely a matter of convenience; we could still carry out the proof if we allowed only $+$ as a numerical predicate.

Given a language $L \subseteq \Sigma^*$ and strings $u, v \in \Sigma^*$, we define the language $u^{-1}Lv^{-1} = \{w \in \Sigma^* | uwv \in L\}$.

LEMMA 5.1. *Let $L \subseteq \Sigma^*$ be definable in $\mathcal{L}$. Then $u^{-1}Lv^{-1}$ is also definable in $\mathcal{L}$.*

Proof. It suffices to prove that $\sigma^{-1}L$ and $L\sigma^{-1}$ are definable in $\mathcal{L}$ for each $\sigma \in \Sigma$. We exhibit a proof of the first of these assertions by constructing a formula $\psi[\phi]$ for $\sigma^{-1}L$ given a formula ϕ for L. (We omit the almost identical proof of the second assertion.) To accomplish this, we encode each position x in σv by a pair of positions (x_1, x_2) in v: We map x to $(1, x-1)$ if $x > 0$, and to $(0, 0)$ if $x = 0$. Note that this requires $|v| \geq 2$, so we must treat the case where $|v| < 2$ separately. The encoding is clearly injective; let us denote its inverse by α.

We first write a formula $\psi_1[\phi]$ satisfied by all strings $v \in \Sigma^*$ of length 0 or 1 such that $\sigma v \models \phi$ (such a formula is trivial to write since there are only three strings to consider). For strings v of length ≥ 2, we show how to construct $\psi_2[\phi]$ by recursion over the term structure of ϕ. The final formula $\psi[\phi]$ is $\psi_1[\phi] \wedge \psi_2[\phi]$.

Our inductive hypothesis is the following: Given a formula $\phi(x_1, x_2, \ldots, x_k)$, there exists a formula $\psi_2[\phi](x_{1,1}, x_{2,1}, \ldots, x_{1,k}, x_{2,k})$ such that if $|v| \geq 2$, then $v \models \psi_2[\phi](b_{1,1}, b_{2,1}, \ldots, b_{1,k}, b_{2,k})$ iff $\sigma v \models \phi(\alpha(b_{1,1}, b_{2,1}), \ldots, \alpha(b_{1,k}, b_{2,k}))$. In particular, the former condition can hold only if all the $\alpha(b_{1,k}, b_{2,k})$ are defined. When there are no free variables then $\sigma v \models \phi$ iff $v \models \psi_2[\phi]$ as desired (if $|v| \geq 2$).

The formula $\psi_2[\phi]$ is defined below, depending on the following choices for ϕ:

(1) $Q_\tau(x)$: If $\tau \neq \sigma$, then $\psi_2[\phi] = (x_1 = 1) \wedge Q_\tau(x_2)$; otherwise set $\psi_2[\phi] = (Q_\sigma(x_2) \wedge x_1 = 1) \vee (x_1 = 0)$.

(2) $x + y = z$: We enumerate the subcases depending on the number of x_1, y_1, z_1 equal to 0:

$$\begin{aligned}\psi_2[\phi] = ((x_1 = y_1 = z_1 = 1) &\wedge (x_2 + y_2 + 1 = z_2))\\ \vee ((x_1 = 0) \wedge (y_1 = z_1 = 1) &\wedge (y_2 = z_2))\\ \vee ((y_1 = 0) \wedge (x_1 = z_1 = 1) &\wedge (x_2 = z_2))\\ \vee (x_1 = y_1 = z_1 = 0)&.\end{aligned}$$

(3) $\neg\phi_1$: $\psi_2[\phi] = \neg\psi_2[\phi_1]$.

(4) $\phi_1 \wedge \phi_2$: $\psi_2[\phi] = \psi_2[\phi_1] \wedge \psi_2[\phi_2]$.

(5) $\exists x\ \phi_1$: $\psi_2[\phi] = \exists\, (x_1, x_2)\ \psi_2[\phi]$.

(6) $\exists^{a \bmod q} x\ \phi_1$:

$$\begin{aligned}\psi_2[\phi] = (&\exists^{a \bmod q}(x_1, x_2)(x_1 = 1) \wedge \psi_2[\phi_1] \wedge \neg\exists(x_1, x_2)(x_1 = 0 \wedge \psi_2[\phi]))\\ &\vee (\exists^{a-1 \bmod q}(x_1, x_2)(x_1 = 1) \wedge \psi_2[\phi_1] \wedge \exists(x_1, x_2)(x_1 = 0 \wedge \psi_2[\phi_1])).\end{aligned}$$

Note that both modular and existential quantification over tuples (x_1, x_2) can be expressed as a boolean combination of quantification over x_1 and x_2 (see (3.6) and the remarks preceding it). Also note that we strictly cannot have terms like $(x + 1) = y$ in our logic as we have written above; we still use these as a (clearer) shorthand for the more elaborate formula $\neg\exists z((x < z) \wedge (z < y)) \wedge (x \neq y)$. □

LEMMA 5.2. *Let Σ, Γ be finite alphabets and let $f : \Gamma^* \to \Sigma^*$ be a homomorphism such that $f(\Gamma) \subseteq \Sigma^r$ for some fixed $r > 0$. If $L \subseteq \Sigma^*$ is definable in $\mathcal{L}$, then $f^{-1}(L) \subseteq \Gamma^*$ is also definable in $\mathcal{L}$.*

Proof. Let ϕ be a formula in $\mathcal{L}$, such that $w \in L$ iff $w \models \phi$. We construct (via recursion over the term structure of ϕ) a formula $\psi[\phi]$ in $\mathcal{L}$ such that for any $v \in \Gamma^*$, $v \models \psi[\phi]$ iff $f(v) \models \phi$. Once again, we do this by encoding each position in $f(v)$ by a pair of positions in v. In this case, x is encoded by $(x \bmod r, \lfloor x/r \rfloor)$. Note that this requires $|v| \geq r$, so again we will have to treat the finite number of exceptions separately. The inverse of this encoding, $\alpha(x_1, x_2) = rx_1 + x_1$, is defined iff $x_1 < r$.

We first write a formula $\psi_1[\phi]$ satisfied by the (finite) set of strings $v \in \Gamma^*$, where $|v| < r$ and $f(v) \models \phi$:

$$\psi_1[\phi] = \bigvee_{\substack{v=\sigma_0\sigma_1\ldots\sigma_{s-1}\\ s<r\\ f(v)\models\phi}} \bigwedge_{i=0}^{s-1} Q_{\sigma_i}(i).$$

For strings v of length $\geq r$, we show how to construct $\psi_2[\phi]$ by recursion over the term structure of ϕ. The final formula $\psi[\phi]$ is $\psi_1[\phi] \wedge \psi_2[\phi]$.

Our inductive hypothesis is the following: Given a formula $\phi(x_1, x_2, \ldots, x_k)$, there exists a formula $\psi_2[\phi](x_{1,1}, x_{2,1}, \ldots, x_{1,k}, x_{2,k})$ such that if $|v| \geq r$, then $v \models \psi_2[\phi](b_{1,1}, b_{2,1}, \ldots, b_{1,k}, b_{2,k})$ iff $f(v) \models \phi(\alpha(b_{1,1}, b_{2,1}), \ldots, \alpha(b_{1,k}, b_{2,k}))$. In particular, the former condition can hold only if all the $\alpha(b_{1,k}, b_{2,k})$ are defined. When there are no free variables then $f(v) \models \phi$ iff $v \models \psi_2[\phi]$ as desired (if $|v| \geq r$).

We set $\psi[\phi] = \psi_1[\phi] \wedge \psi_2[\phi]$, where the formula $\psi_2[\phi]$ is defined recursively, depending on the following choices for ϕ:

(1) $Q_\sigma x$: Then $\psi_2[\phi] = (\bigvee_{f(\gamma)_i=\sigma} Q_\gamma(x_2) \wedge x_1 = i)$.

(2) $x + y = z$: This would imply that $x_1 + y_1 - z_1 = r(z_2 - x_2 - y_2)$. Since $1 \leq x_1 + y_1 - z_1 < 2r$ and $r | (x_1 + y_1 - z_1)$, the left-hand side $x_1 + y_1 - z_1$ is either r or 0 (and this determines the right-hand side's values). Thus

$$\psi_2[\phi] = (x_1 + y_1 = z_1 \wedge x_2 + y_2 = z_2) \vee (x_1 + y_1 = z_1 + r \wedge x_2 + y_2 + 1 = z_2).$$

(3) $\neg\phi_1$: $\psi_2[\phi] = \neg\psi_2[\phi_1]$.

(4) $\phi_1 \wedge \phi_2$: $\psi_2[\phi] = \psi_2[\phi_1] \wedge \psi_2[\phi_2]$.

(5) $\exists\, x\ \phi_1$: $\psi_2[\phi] = \exists\,(x_1, x_2)\ ((x_1 < r) \wedge \psi_2[\phi_1])$.

(6) $\exists^{a \bmod q}\, x\ \phi_1$: $\psi_2[\phi] = \exists^{a \bmod q}\,(x_1, x_2)\ ((x_1 < r) \wedge \psi_2[\phi_1])$.

As in Lemma 5.1, both modular and existential quantification over tuples (x_1, x_2) can be expressed as a boolean combination of quantification over x_1 and x_2 (see (3.6) and the remarks preceding it). □

5.2. Reduction to the neutral-letter case. We prove that every group contained in $\mu_L(\Sigma^t)$ (in the statement of Theorem 4.5) is the syntactic monoid of a (regular) language with a neutral letter definable in $\mathcal{L}$. Then Theorem 4.4 implies that every such group has to be solvable and has cardinality dividing a power of q. This reduction to the neutral letter case is done in Lemma 5.3 below. Note that the reverse direction follows easily from Straubing, Thérien, and Thomas [21]: If every group in $\mu_L(\Sigma^t)$ is solvable and has order dividing a power of q, then every subgroup of $M(L)$ is solvable and has order dividing a power of q, and this implies that L is definable in $(FO + MOD_q)[\{\pi_\sigma : \sigma \in \Sigma\}, <]$ and hence is definable in $\mathcal{L}_{q,\Sigma,+}$.

Lemma 5.3. *Let $L \subseteq \Sigma^*$ be regular, and suppose L is definable in $\mathcal{L}$. Then for every $t \geq 0$ and every group $G \subseteq \mu_L(\Sigma^t)$, there exists a finite alphabet $\Gamma = \Gamma_G$ and a language $L_G \subseteq \Gamma^*$, such that L_G has a neutral letter and is definable in $\mathcal{L}$. Moreover, L_G is regular and $M(L_G) = G$.*

Proof. We define the finite alphabet

$$\Gamma = \{\gamma_w : w \in \Sigma^t,\ \mu_L(w) \in G\}.$$

The map $\gamma_w \mapsto w$ extends to a homomorphism f from Γ^* into Σ^* such that $f(\Gamma) \subseteq \Sigma^t$. We define

$$L_G = \{v \in \Gamma^* : \mu_L(f(v)) = e\},$$

where e is the identity of G.

Note that L_G has a neutral letter γ_v, where $\mu_L(v) = e$. We will show shortly that L_G is definable by a sentence of $\mathcal{L}$. First note that L_G is regular: It is recognized by a deterministic finite automaton (DFA) with state set G, initial and accepting state e, and state transitions

$$\gamma_w : g \mapsto g\mu_L(w).$$

Every state of this automaton is accessible from the initial state, and equivalent states must be identical, because of cancellation in the group. Thus this is the minimal DFA of L_G, and consequently $M(L_G) = G$.

It remains to establish the claim about definability of L_G in $\mathcal{L}$. It is well known that if $K \subseteq \Sigma^*$ is regular, then for each $m \in M(K)$, the set $\mu_K^{-1}(m)$ is a finite boolean combination of languages of the form

$$u^{-1}Kv^{-1} = \{w \in \Sigma^* : uwv \in K\},$$

where $u, v \in \Sigma^*$. We have

$$L_G = f^{-1}(\mu_L^{-1}(e)),$$

so our claim will follow if we can show that definability is preserved under the language operations

$$K \mapsto u^{-1}Kv^{-1}$$

and

$$K \mapsto f^{-1}(K).$$

This was established by Lemmas 5.1 and 5.2. □

6. Monadic predicates. In this section, we consider definability of languages in first-order logic with modular quantifiers when we allow monadic (i.e., arity 1) numerical predicates. More specifically, we consider the logic $MON_q = (FO + MOD_q)[<, \{\pi_\sigma\}_{\sigma\in\Sigma}, \theta_1, \theta_2, \ldots, \theta_r]$, where θ_i, $1 \leq i \leq r$, are bit-valued functions on $\mathbb{N}$.

We define a map

$$\widehat{\ }: \Sigma^* \to (\Sigma \times \{0,1\}^r)^*$$

as follows:

$$w = \sigma_0\sigma_1 \ldots \sigma_{n-1} \mapsto \widehat{w} = (\sigma_0, \gamma_{1,0}, \gamma_{2,0}, \ldots, \gamma_{r,0}) \cdots (\sigma_{n-1}, \gamma_{1,n-1}, \gamma_{2,n-1}, \ldots, \gamma_{r,n-1}),$$

where $\gamma_{i,j} = \theta_i(j)$. Given $L \subseteq \Sigma^*$, we denote $\widehat{L} = \{\widehat{w} |\ w \in L\}$.

Lemma 6.1. *Let ϕ be a sentence in MON_q. There exists a language $K \subseteq (\Sigma \times \{0,1\}^r)^*$ such that* (a) *K is regular, and every group in the syntactic monoid of K is solvable with order dividing a power of q;* (b) *if $w \in \Sigma^*$, then $w \models \phi$ iff $\widehat{w} \in K$.*

Proof. We take the formula ϕ and rewrite it by replacing every occurrence of $\theta_i(x)$ by the disjunction of all $\pi_{(\sigma,v)}x$, with $\sigma \in \Sigma$ and $v \in \{0,1\}^r$, for which the ith component of v is 1. Likewise we replace every occurrence of $\pi_\sigma x$ by the disjunction of $\pi_{(\sigma,v)}$ over all $v \in \{0,1\}^r$. By [21], the resulting sentence $\widehat{\phi}$, interpreted in words over $\Sigma \times \{0,1\}^r$, defines a regular language K whose syntactic monoid possesses the desired property, and it is clear that $w \models \phi$ iff $\widehat{w} \models \hat{\phi}$. (Observe that not every element of K is $\widehat{w}$ for some $w \in \Sigma^*$.) □

We need the following lemma, which follows from Ramsey's theorem.

Lemma 6.2. *Consider a k-coloring of the set $\{(i,j) | \ 1 \leq i \leq j\} \subseteq \mathbb{N} \times \mathbb{N}$. Then there is an infinite sequence $\{i_j\}$ with*

$$1 \leq i_1 < i_2 < \cdots$$

such that all (i_j, i_{j+1}) have the same color.

(We note that the full strength of Ramsey's theorem is not required here, as we do not need all (i_j, i_k) with $j < k$ to have the same color. A weaker combinatorial principle, along the lines of the Erdös–Szekeres theorem on the existence of long monotone subsequences of arbitrary sequences, will suffice. See [6], which is the source for the kind of argument that we use in the present section.)

Theorem 6.3. *If L is a language with a neutral letter definable in MON_q, then it is regular and definable in $(FO + MOD_q)[<, \{\pi_\sigma\}_{\sigma \in \Sigma}]$. Furthermore, the syntactic monoid of L is solvable and every group in the syntactic monoid has order dividing a power of q.*

Proof. We let $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_t\}$ and let $\lambda \in \Sigma$ be the neutral letter for L (so that $\lambda = \sigma_i$ for some i).

We extend the notation we used to define the function $\widehat{\ }$: We set $\widehat{(w,i)}$ to be the suffix of length $|w|$ of $\widehat{vw}$, where v is any string of length i. This is independent of the choice of the string v. Note that $\widehat{w} = \widehat{(w,0)}$.

Suppose L is definable by a sentence ϕ in MON_q. Let $K \subseteq (\Sigma \times \{0,1\}^r)^*$ be the regular language whose existence is proved in Lemma 6.1. Let M be its syntactic monoid and let $\mu : (\Sigma \times \{0,1\}^r)^* \to M$ be its syntactic morphism. Furthermore let $X \subseteq M$ be such that $K = \mu^{-1}(X)$.

Let $w = \tau_1\tau_2 \ldots \tau_n \in \Sigma^*$ and $|w| = n$. We color (i,j), $1 \leq i < j$, $i,j \in \mathbb{N}$, by $(m_{\sigma_1}, m_{\sigma_2}, \ldots, m_{\sigma_t})$, where $m_{\sigma_k} = \mu(\sigma_k\widehat{\lambda^{j-i-1}, i-1}) \in M$. By Lemma 6.2, there is a sequence $1 \leq i_1 < i_2 < \cdots < i_{n+1}$ such that (i_j, i_{j+1}), $0 \leq j \leq n$, have the same color. Define

$$\mathrm{pad}(w) = \lambda^{i_1-1}\tau_1\lambda^{i_2-i_1-1}\tau_2 \ldots \lambda^{i_n-i_{n-1}-1}\sigma_n\lambda^{i_{n+1}-i_n-1}.$$

Since λ is a neutral letter for L, $w \in L$ iff $\mathrm{pad}(w) \in L$. Observe that

$$\mu(\widehat{\mathrm{pad}(w)}) = m_0 m_{\tau_1} m_{\sigma_2} \cdots m_{\tau_n},$$

where $m_0 = \mu(\widehat{\lambda^{i_1-1}})$. Thus $w \in L$ iff $m_0\nu(w) \in X$, where $\nu : \Sigma^* \to M$ is the homomorphism defined by $\nu(\sigma_i) = m_{\sigma_i}$ for $1 \leq i \leq t$. This implies that there is a set $Y = \{m \in M | \ m_0 m \in X\}$ such that $w \in L$ iff $\nu(w) \in Y \subset M$. Thus L is regular, and its syntactic monoid is a quotient of a submonoid of M, which implies that all the groups in $M(L)$ are solvable and have order dividing a power of q. The conclusion about logical definability of L now follows from the results of [21]. □

7. Directions for further research. In many steps of the algorithm for reducing a sentence defining L_m to an active-domain sentence, we introduced ordinary quantifiers even when the original formula had only modular quantifiers. If there were a way to avoid this, we could also prove, by the same techniques, that the language $0^*1\{0,1\}^*$ cannot be defined by a formula over $(\mathbb{N},+)$ having only modular quantifiers. If addition is replaced by arbitrary numerical predicates, this statement is equivalent to the conjecture that the circuit complexity class CC^0 does not contain the language 1^*. (CC^0 is the class of languages recognized by constant-depth, polynomial-size circuit families in which every gate is a MOD_q gate for a fixed modulus q. See Barrington, Straubing, and Thérien [7].)

One can ask in general for what classes $\mathcal{C}$ of numerical predicates we have that every language in $(FO + MOD_q)[\mathbb{N},\mathcal{C}]$ with a neutral letter is regular and definable using only the ordering in $<$. The question is discussed at length in [5], and in [12] in the more general context of collapse results for embedded finite models. One can investigate whether, as is the case for first-order logic, finite VC-dimension of $(\mathbb{N},\mathcal{C})$ implies the collapse. This would require generalizing the results of Baldwin and Benedikt [1] to modular quantifiers.

But there is a limit to how far we can push this approach. We are really interested in proving our result over a base of arbitrary numerical predicates, or at the very least over the base $\{+,\times\}$. However, with $\{+,\times\}$, one can express all problems in the arithmetic hierarchy (section 4 in [5]). Specifically, it is possible in this logic to define the set of infinite strings with an even number of 1's in first-order logic without using modular quantifiers! Let $E(x)$ be the numerical predicate "the binary expansion of x contains an even number of 1's," and $B(x,y)$ the predicate "bit y in the binary expansion of x is 1." Then the set of infinite bit strings with an even number of 1's is defined by

$$\exists x(E(x) \wedge \forall y(\pi(y) \leftrightarrow B(x,y))).$$

Both E and B are definable over $(+,\times)$. This shows that we cannot extend the collapse arguments to these richer logics. It also shows (since we know, from circuit complexity, that first-order sentences cannot define PARITY for *finite* strings) that there are important differences between finite and infinite strings as regards definability.

One possible approach to formulas with more general numerical predicates is to try to prove some version of the collapse results for sentences interpreted in finite strings that are known to define regular languages. We do know, for example, thanks to the circuit lower bounds, that regular languages definable by first-order sentences with arbitrary numerical predicates are all definable in $FO[<,\{\equiv_s\colon s>1\}]$, and the same holds even when we add modular quantifiers of fixed prime modulus, so we do indeed have a collapse result, although this has never been proved directly by model-theoretic means (see [3] and [20]).

Acknowledgments. We acknowledge helpful discussions with Klaus-Jörn Lange, Denis Thérien, David Mix Barrington, and the late Clemens Lautemann. We also thank Andreas Krebs for pointing out to us that our original proofs of Lemmas 5.1 and 5.2 were unnecessarily long and complicated, and for supplying the simpler arguments given here.

REFERENCES

[1] J. Baldwin and M. Benedikt, *Stability theory, permutations of indiscernibles and embedded finite models*, Trans. Amer. Math. Soc., 352 (2000), pp. 4937–4969.

[2] D. M. Barrington, *Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1*, J. Comput. System Sci., 38 (1989), pp. 150–164.

[3] D. M. Barrington, K. Compton, H. Straubing, and D. Thérien, *Regular languages in NC^1*, J. Comput. System Sci., 44 (1992), pp. 478–499.

[4] D. M. Barrington, N. Immerman, and H. Straubing, *On uniformity in NC^1*, J. Comput. System Sci., 41 (1990), pp. 274–306.

[5] D. M. Barrington, N. Immerman, C. Lautemann, N. Schweikardt, and D. Thérien, *First-order expressibility of languages with neutral letters or the Crane Beach conjecture*, J. Comput. System Sci., 70 (2005), pp. 101–127.

[6] D. M. Barrington and H. Straubing, *Superlinear lower bounds for bounded-width branching programs*, J. Comput. System Sci., 50 (1995), pp. 374–381.

[7] D. M. Barrington, H. Straubing, and D. Thérien, *Nonuniform automata over groups*, Inform. and Comput., 89 (1990), pp. 109–132.

[8] D. M. Barrington and D. Thérien, *Finite monoids and the fine structure of NC^1*, J. ACM, 35 (1988), pp. 941–952.

[9] C. Behle and K.-J. Lange, *$FO[<]$-uniformity*, in Proceedings of the IEEE Conference on Computational Complexity, 2006, pp. 183–189.

[10] M. Benedikt and L. Libkin, *Relational queries over interpreted structures*, J. ACM, 47 (2000), pp. 644–680.

[11] A. Krebs, K.-J. Lange, and S. Reifferscheid, *Characterizing TC^0 in terms of infinite groups*, in Proceedings of the 22nd International Symposium on Theoretical Aspects of Computer Science (STACS'05), Lecture Notes in Comput. Sci. 3404, Springer, Berlin, 2005, pp. 496–507.

[12] L. Libkin, *Embedded finite models and constraint databases*, in Finite Model Theory and Its Applications, E. Grädel et al., eds., Springer, New York, 2005.

[13] L. Libkin, *Elements of Finite Model Theory*, Springer, New York, 2004.

[14] J. F. Lynch, *On sets of relations definable by addition*, J. Symbolic Logic, 47 (1982), pp. 659–668.

[15] J. Nurmonen, *Counting modulo quantifiers on finite structures*, Inform. and Comput., 160 (2000), pp. 183–207.

[16] M. Presburger, *Ueber die Vollstaendigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt*, in Comptes Rendus du I congrès de Mathématiciens des Pays Slaves, Warsaw, Poland, 1929, pp. 92–101.

[17] A. A. Razborov, *Lower bounds for the size of circuits of bounded depth with basis* $\{\wedge, \oplus\}$, Math. Notes Soviet Acad. Sci., 41 (1987), pp. 333–338.

[18] N. Schweikardt, *Arithmetic, first-order logic, and counting quantifiers*, ACM Trans. Comput. Log., 6 (2005), pp. 634–671.

[19] R. Smolensky, *Algebraic methods in the theory of lower bounds for Boolean circuit complexity*, in Proceedings of the 19th ACM Symposium on Theory of Computing (STOC), 1987, pp. 77–82.

[20] H. Straubing, *Finite Automata, Formal Logic and Circuit Complexity*, Birkhäuser, Boston, 1994.

[21] H. Straubing, D. Thérien, and W. Thomas, *Regular languages defined with generalized quantifiers*, Inform. and Comput., 118 (1995), pp. 289–301.

SIAM J. Comput.
Vol. 37, No. 2, pp. 522–551

LINES AND FREE LINE SEGMENTS TANGENT TO ARBITRARY THREE-DIMENSIONAL CONVEX POLYHEDRA*

HERVÉ BRÖNNIMANN†, OLIVIER DEVILLERS‡, VIDA DUJMOVIĆ§, HAZEL EVERETT¶, MARC GLISSE¶, XAVIER GOAOC¶, SYLVAIN LAZARD¶, HYEON-SUK NA‖, AND SUE WHITESIDES**

Abstract. Motivated by visibility problems in three dimensions, we investigate the complexity and construction of the set of tangent lines in a scene of three-dimensional polyhedra. We prove that the set of lines tangent to four possibly intersecting convex polyhedra in $\mathbb{R}^3$ with a total of n edges consists of $\Theta(n^2)$ connected components in the worst case. In the generic case, each connected component is a single line, but our result still holds for arbitrarily degenerate scenes. More generally, we show that a set of k possibly intersecting convex polyhedra with a total of n edges admits, in the worst case, $\Theta(n^2k^2)$ connected components of maximal free line segments tangent to at least four polytopes. Furthermore, these bounds also hold for possibly occluded lines rather than maximal free line segments. Finally, we present an $O(n^2k^2 \log n)$ time and $O(nk^2)$ space algorithm that, given a scene of k possibly intersecting convex polyhedra, computes all the *minimal* free line segments that are tangent to any four of the polytopes and are isolated transversals to the set of edges they intersect; in particular, we compute at least one line segment per connected component of tangent lines.

Key words. computational geometry, 3D visibility, visibility complex, visual events

AMS subject classifications. 65D18, 68U05

DOI. 10.1137/S0097539705447116

1. Introduction.

Computing visibility relations in a three-dimensional (3D) environment is a problem central to computer graphics and engineering tasks such as radio propagation simulation and fast prototyping. Examples of visibility computations include determining the view from a given point, and computing the umbra and penumbra cast by a light source. In many applications, visibility computations are well known to account for a significant portion of the total computation cost. Consequently a large body of research is devoted to speeding up visibility computations through the use of data structures (see [14] for a survey).

One such structure, the visibility complex [16, 23], encodes visibility relations by partitioning the set of maximal free line segments. The size of this partition is intimately related to the number of maximal free line segments tangent to four objects in the scene; for a scene of n triangles in $\mathbb{R}^3$, the complex can have size $\Theta(n^4)$ in the

*Received by the editors February 28, 2005; accepted for publication (in revised form) November 6, 2006; published electronically June 5, 2007. This research was initiated at the McGill-INRIA Workshop on Computational Geometry in Computer Graphics, 2002, co-organized by H. Everett, S. Lazard, and S. Whitesides, and held at the Bellairs Research Institute of McGill University. This research was supported by the McGill-ISA collaborative INRIA project, NSERC, FCAR, and NSF. A preliminary version of this paper appeared in *Proceedings of the 20th Annual ACM Symposium on Computational Geometry*, New York, 2004, pp. 46–55.

http://www.siam.org/journals/sicomp/37-2/44711.html

†Polytechnic University, Brooklyn, NY 11201 (hbr@poly.edu).

‡INRIA Sophia-Antipolis, Sophia-Antipolis, France (Olivier.Devillers@inria.fr).

§School of Computer Science, Carleton University, Ottawa, ON, Canada (vida@scs.carleton.ca).

¶LORIA–INRIA Lorraine, University Nancy 2, Nancy, France (everett@loria.fr, glisse@loria.fr, goaoc@loria.fr, lazard@loria.fr).

‖School of Computing, Soongsil University, Seoul, South Korea (hsnaa@computing.ssu.ac.kr).

**School of Computer Science, McGill University, Montréal, QC, Canada (sue@cs.mcgill.ca).

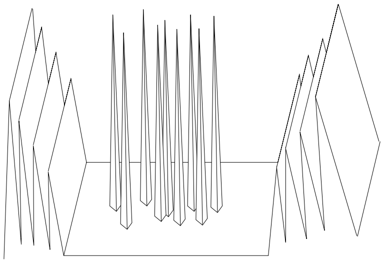


FIG. 1.1. *A terrain of size n with $\Omega(n^4)$ maximal free line segments tangent in four points.*

worst case [16], even when the triangles form a terrain (see [11] or Figure 1.1). The complex is thus potentially enormous, which has hindered its application in practice. However, there is evidence, both theoretical and practical, that this estimation is pessimistic. The lower bound examples, which are carefully designed to exhibit the worst-case behavior, are unrealistic in practice. For realistic scenes, Durand, Drettakis, and Puech [15] observed a quadratic growth rate, albeit for rather small scenes. For random scenes, Devillers et al. [12] proved that the expected size of the visibility complex is much smaller; for uniformly distributed unit balls the expected size is linear and for polygons or polyhedra of bounded aspect ratio and similar size it is at most quadratic. Also, in two dimensions, while the worst-case complexity of the visibility complex is quadratic, experimental results strongly suggest that the size of the visibility complex of a scene consisting of scattered triangles is linear [10].

While these results are encouraging, most scenes are not random. In fact, most scenes have a lot of structure which we can exploit; a scene is typically represented by many triangles which form a much smaller number of convex patches. In particular, if a scene consists of k *disjoint* convex polyhedra with a total of n edges, then under a strong general position assumption, the number of maximal free line segments tangent to four of the polyhedra is at most $O(n^2k^2)$; this follows directly from the bound proved in [17] on the number of combinatorial changes of the silhouette map viewed from a point moving along a straight line, and was also later proved in [8]. We present in this paper a generalization of these results. After preliminary definitions, we give a detailed account of our results and then present related previous work.

Preliminary definitions. We consider a scene that consists of a finite number of polytopes, not necessarily disjoint, not necessarily fully dimensional, and in arbitrary position. The definitions below are standard, yet carefully phrased in a way that remains valid in those situations.

A *polytope* is the convex hull of a point set. A plane is *tangent* to a polytope if it intersects the polytope and bounds a closed half-space that contains the polytope. A face, an edge, or a vertex of a polytope in $\mathbb{R}^3$ is the 2-, 1-, or 0-dimensional intersection of the polytope with a tangent plane. Note that, with this usual definition of polytopes, edges and faces are closed and are not subdivided in any way.

A line or segment is *tangent* to a polytope (whether or not the latter is fully dimensional) if it intersects the polytope and is contained in a tangent plane. In a given plane, a line is tangent to a polygon if it intersects the polygon and bounds a closed half-plane that contains the polygon. With these definitions, given a polygon in a plane π, and a line contained in π that intersects the relative interior of this polygon, the line is tangent to the polygon when considered as a polytope in $\mathbb{R}^3$, but not tangent to the polygon in the plane π.

The set of lines in $\mathbb{R}^3$ has a natural topological structure, namely, that of Plücker space [25]. The set of lines tangent to at least four polytopes is a subspace, whose *connected components* correspond to lines that can be continuously moved one into the other while remaining tangent to at least four polytopes.[1] A line or line segment is *free* if it is tangent to each polytope that its relative interior intersects;[2] otherwise it is *occluded*. A free line segment is a *maximal free line segment* if it is not properly contained in another free line segment. The space of line segments also has a natural topological structure, and the *connected components* of maximal free line segments tangent to at least four among the k polytopes are defined similarly as for lines.

A *support vertex* of a line is a polytope vertex that lies on the line. A *support edge* of a line is a polytope edge that intersects the line but has no endpoint on it (a support edge intersects the line at only one point of its relative interior). A *support* of a line is one of its support vertices or support edges. The supports of a segment are defined similarly. Notice that it follows from the definition of polytopes that any line has at most two supports in any given polytope.

A line is *isolated with respect to* a set of edges and vertices if the line cannot be moved continuously while remaining a common transversal to these edges and vertices. Furthermore, we say that a set $\mathcal{S}$ of edges and vertices *admits an isolated transversal* if these edges and vertices admit a common transversal that is isolated with respect to $\mathcal{S}$. Finally, a line is *isolated* if it is isolated with respect to a set of some, and hence all, of its supports.

Our results. In this paper we present two types of results, combinatorial bounds and algorithms.

Combinatorial bounds. We generalize the result of [8, 17] in two ways. First, we consider polytopes that may *intersect*. We show that among k polytopes of total complexity n, the number of lines tangent to any four of them is in the worst case either infinite or $\Theta(n^2k^2)$. The most surprising aspect of this result is that the bound (which is tight) is the same whether the polytopes intersect or not. This is in sharp contrast to the 2D case, where the number of tangents of two convex polygons is always four if disjoint and could be linear in the size of the polygons if they intersect. Second, we consider polytopes in *arbitrary position*: we drop all general position assumptions. The polytopes may intersect in any way; they may overlap or coincide. They may degenerate to polygons, segments, or points. While four polytopes in general position (as defined in [8]) admit a finite number of common tangents, four polytopes in arbitrary position may admit an infinite number of common tangents which can be partitioned into connected components.

Our main results are, more precisely, the following.

THEOREM 1.1. *Given k polytopes in $\mathbb{R}^3$ with n edges in total, there are, in the worst case, $\Theta(n^2k^2)$ connected components of maximal free line segments tangent to at least four of the polytopes. This bound also holds for connected components of possibly occluded lines tangent to at least four of the polytopes.*

These results improve the trivial bound of $O(n^4)$. Note that, when $k \neq 4$, neither of the two results stated in Theorem 1.1 implies the other since a line tangent to at least four among k polytopes may contain many, but does not necessarily contain any,

[1] The set of polytopes to which the line is tangent might change during the motion.

[2] When the polytopes are fully dimensional, a segment is free if it does not intersect the interior of any of them. Our definition ensures that a segment is free also when it intersects and is coplanar with a two-dimensional (2D) polytope. The endpoints of a free segment may also lie on the boundary of a polytope.

maximal free line segments tangent to four polytopes.

When $k = 4$, Theorem 1.1 implies that there are $\Theta(n^2)$ connected components of lines tangent to the four polytopes, an improvement on the previously known upper bound of $O(n^3 \log n)$ which follows from the same bound on the complexity of the set of line transversals to a set of polyhedra (here four) with n edges in total [1]. Moreover, we prove a tighter bound when one of the four polytopes has few edges.

Theorem 1.2. *Given three polytopes with n edges in total and one polytope with m edges, there are, in the worst case, $\Theta(mn)$ connected components of lines tangent to the four polytopes.*

We also prove the following result which is more powerful, though more technical, than Theorem 1.1. Whereas Theorem 1.1 bounds the number of connected components of tangents, Theorem 1.3 bounds the number of isolated tangents with some notion of multiplicity. For example, the line in Figure 1.2 is counted $\binom{k}{2}$ times, which is the number of minimal sets of vertices that admit that line as an isolated transversal. Although neither theorem implies the other, we will prove in Proposition 3.4 that the upper bound of Theorem 1.1 is easily proved using Theorem 1.3.

Theorem 1.3. *Given k polytopes in $\mathbb{R}^3$ with n edges in total, there are, in the worst case, $\Theta(n^2k^2)$ minimal sets of open edges and vertices, chosen from some of the polytopes, that admit a possibly occluded isolated transversal that is tangent to these polytopes.*

Algorithm. We now turn our attention to the computation of all free segments that are isolated transversals to their set of supports and tangent to the corresponding polytopes. Durand, Drettakis, and Puech [16] proposed an algorithm for this problem with worst-case time complexity $O((n^3 + p) \log n)$, where p is the output size; this algorithm, based on a double-sweep, has proved to be difficult to implement. Durand, Drettakis, and Puech also presented an algorithm with $\Theta(n^5)$ worst-case time complexity that incorporates interesting heuristics leading to reasonable performance in practice [15]. We present an algorithm that uses, in the worst case, $O(n^2k^2 \log n)$ time and $O(nk^2)$ space, is readily implementable, and uses only simple data structures. The polytopes may intersect and be in arbitrary position. A preliminary version of this algorithm was described for disjoint convex polyhedra in Goaoc's Ph.D. thesis [19].

Theorem 1.4. *Given k polytopes in $\mathbb{R}^3$ with n edges in total, we can compute, in $O(n^2k^2 \log n)$ time and $O(nk)$ space, all the possibly occluded lines that are isolated transversals to their set of supports and tangent to the corresponding polytopes. We can also compute, in $O(n^2k^2 \log n)$ time and $O(nk^2)$ space, all the minimal free segments that are isolated transversals to their set of supports and tangent to the corresponding polytopes.*

It should be noted that our algorithm does not provide the endpoints (possibly at infinity) of the maximal free segments. Computing the endpoints of each such segment can be done by shooting rays in $O(\log^2 n)$ time per ray using $O((nk)^{2+\epsilon})$ preprocessing time and storage [3]. Such ray-shooting data structures are not, however, readily implementable. Alternatively, each ray-shooting query can be answered in $O(k \log n)$ time after $O(n \log n)$ preprocessing time and using additional $O(n)$ space by applying the Dobkin–Kirkpatrick hierarchy on each polytope [13].

To emphasize the importance of considering intersecting polytopes, observe that computer graphics scenes often contain nonconvex objects. These objects, however, can be decomposed into sets of convex polyhedra. Notice that simply decomposing these objects into convex polyhedra with disjoint interiors may induce a scene of much higher complexity than a decomposition into intersecting polytopes. Moreover,

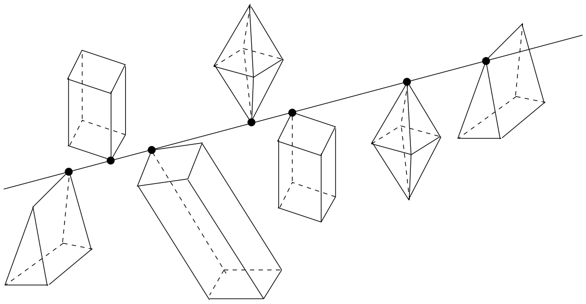


FIG. 1.2. *A line tangent at a vertex of each of k polytopes.*

TABLE 1.1

Published bounds on the complexity of the set of free lines or maximal free line segments among objects of total complexity n. The expected complexities are given for the uniform distribution of the balls centers.

	Worst-case	Expected
Free lines to a polyhedron	$\Theta(n^4)$ (trivial)	
Free lines above a polyhedral terrain	$O(n^3 2^{c\sqrt{\log n}})$ [20, 22]	
Free lines among disjoint homothetic polytopes	$\Omega(n^3)$ [4]	
Free lines among unit balls	$\Omega(n^2)$ [12], $O(n^{3+\epsilon})$ [2]	$\Theta(n)$ [12]
Max. free segments above a polyhedral terrain	$\Theta(n^4)$ [11]	
Isolated maximal free segments among k generic disjoint convex polyhedra	$\Theta(n^2 k^2)$ [17, 8]	
Max. free segments among unit balls	$\Omega(n^2)$ [12], $O(n^4)$	$\Theta(n)$ [12]

the decomposition of a polyhedron into interior-disjoint polytopes may introduce new tangents which were not present in the original scene; indeed a line tangent to two polytopes along a shared face is not tangent to their union.

The importance of considering polytopes in arbitrary position comes from the fact that graphics scenes are full of degeneracies both in the sense that four polytopes may admit infinitely many tangents and that polytopes may share edges or faces. There may actually be more connected components of tangents when the objects are in degenerate position; this is, for instance, the case for line segments [9]. Also, we could not find a perturbation argument that guarantees the preservation of all (or at least a constant fraction of) the connected components of tangents, and we do not believe that finding such a perturbation is a simple matter.

Related results. Previous results on this topic include those that bound the complexity of sets of free lines or free line segments among different sets of objects. They are summarized in Table 1.1.

Recently, Agarwal et al. [2] proved that the set of free lines among n unit balls has complexity $O(n^{3+\epsilon})$. Devillers et al. showed a simple bound of $\Omega(n^2)$ [12] for this problem, and Koltun recently sketched a bound of $\Omega(n^3)$ (private communication, 2004).

The complexity of the set of free line segments among n balls is trivially $O(n^4)$. Devillers and Ramos showed that the set of free line segments can have complexity $\Omega(n^3)$ (private communication, 2001; see also [12]). When the balls are unit size, the $\Omega(n^2)$ lower bound for the set of free lines holds. A lower bound of $\Omega(n^4)$ that applies to either case was recently sketched by Glisse (private communication, 2004).

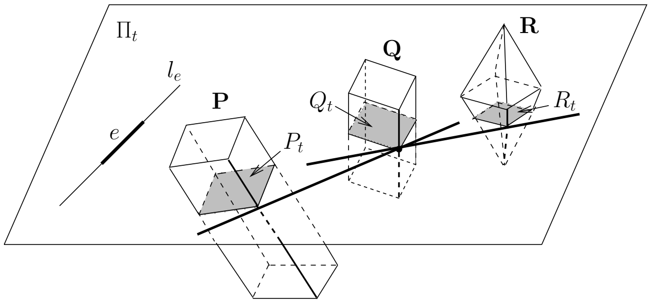


FIG. 2.1. *Plane Π_t contains edge e and intersects polytopes* **P**, **Q**, *and* **R** *in polygons P_t, Q_t, and R_t.*

We mention two results for polyhedral environments. Halperin and Sharir [20] and Pellegrini [22] proved that, in a polyhedral terrain with n edges, the set of free lines has near-cubic complexity. De Berg, Everett, and Guibas [4] showed an $\Omega(n^3)$ lower bound on the complexity of the set of free lines (and thus free segments) among n disjoint homothetic convex polyhedra.

This paper is organized as follows. We prove the upper bounds of Theorems 1.1, 1.2, and 1.3 in sections 2 and 3, and the lower bounds in section 4. In section 5, we present our algorithm for computing free segments.

2. Main lemma. We prove in this section a lemma which is fundamental for the proofs of the upper bounds of Theorems 1.1, 1.2, and 1.3. Consider four polytopes **P**, **Q**, **R**, and **S** in $\mathbb{R}^3$, with p, q, r, and $s \geqslant 1$ edges, respectively, and let e be an edge of **S**.

MAIN LEMMA. *There are $O(p + q + r)$ isolated lines intersecting e and tangent to* **P**, **Q**, **R**, *and* **S**, *excluding those that lie in planes that contain e and are tangent to all four polytopes.*

The proof of the Main Lemma is rather complicated because it handles polytopes which may intersect as well as all the degenerate cases. To assist the reader, we first give an overview of the proof. We then state preliminaries and definitions in section 2.2. In sections 2.3 and 2.4, we bound the number of "generic tangent lines." In section 2.5, we bound the number of "nongeneric tangent lines." Finally, in section 2.6, we pull these results together to conclude the proof of the Main Lemma.

2.1. Proof overview. The proof is inspired by a method which was, to our knowledge, first used in [6] (and later in [5, 17, 8]). We present here an overview of the proof in which we do not address most of the problems arising from degeneracies. In particular, some definitions and remarks will require more elaboration in the context of the complete proof.

We sweep the space with a plane Π_t rotating about the line containing e. The sweep plane intersects the three polytopes **P**, **Q**, and **R** in three, possibly degenerate or empty, convex polygons denoted P_t, Q_t, and R_t, respectively (see Figure 2.1). During the sweep, we track the *bitangents*, that is, the lines tangent to P_t and Q_t, or to Q_t and R_t, in Π_t. As the sweep plane rotates, the three polygons deform, and the bitangents move accordingly. Every time two bitangents become aligned during the sweep, the common line they form is tangent to **P**, **Q**, and **R**.

In any given instance of the sweep plane Π_t, we consider the pairs of bitangents (one involving P_t and Q_t and the other Q_t and R_t) that share a vertex of Q_t (see

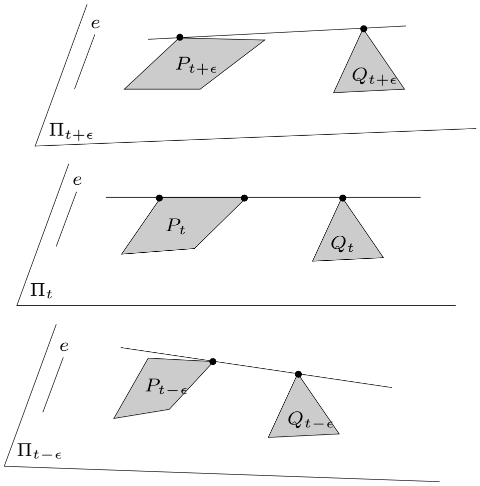


FIG. 2.2. *A bitangent to P_t and Q_t is tangent to P_t along an edge. The plane Π_t is F-critical.*

Figure 2.1). The isolated lines intersecting e and tangent to **P**, **Q**, **R**, and **S** are isolated transversals with respect to a tuple of supports that consists of e and the supports of two such bitangents. We consider all *candidate* such tuples of supports as the sweep plane rotates.

Such a tuple induced by an instance of the sweep plane changes as the plane rotates only when a support of a bitangent changes. We define *critical planes* in such a way that the supports of the bitangents do not change as the sweep plane rotates between two consecutive critical planes. As the sweep plane rotates, the supports of a bitangent change if a support starts or ceases to be swept, or if, during its motion, the bitangent becomes tangent to one of the polygons along an edge of that polygon (see Figure 2.2). In the latter case, this means that the bitangent crosses a face or contains an edge of one of the polytopes. We thus define two types of critical planes: an instance of the sweep plane is critical if it contains a vertex of one of the polytopes, or if it contains a line that lies in the plane containing a face of one of the polytopes, and is tangent to another of the polytopes (see Figures 2.2 and 2.3). We will show that the number of critical planes is $O(p + q + r)$.

When the polytopes intersect there may exist a linear number of bitangents in an instance of the sweep plane (two intersecting convex polygons may admit a linear number of bitangents, as is the case for two regular n-gons where one is a rotation of the other about its center). Thus there can be a linear number of candidate tuples induced by any instance of the sweep plane, and the linear number of critical planes leads to a quadratic bound on the total number of distinct candidate tuples. In the detailed proof of the lemma, we amortize the count of candidate tuples over all the critical planes to get a linear bound on the number of distinct candidate tuples and thus on the number of isolated lines intersecting e and tangent to **P**, **Q**, **R**, and **S**; this bound will, however, not hold for those isolated lines that lie in planes that contain e and are tangent to all four polytopes. Indeed, the number of such isolated tangent lines can be quadratic in degenerate cases; for instance, four polytopes such that a plane contains edge e and a face of linear complexity from each of the other polytopes may admit in this plane a quadratic number of such isolated tangent lines (one through each of a quadratic number of pairs of vertices).

2.2. Preliminaries and definitions. We can assume without loss of generality that **P**, **Q**, **R**, and **S** have *nonempty interiors.* Indeed, since the set of isolated tangent lines to the four polytopes is zero-dimensional, there is always room to extend any polytope with empty interior in such a way that none of the original isolated tangent lines are lost.

We say that a line *properly* intersects a polygon if it intersects its relative interior. In what follows, we use this definition only when the line and polygon are coplanar. Notice that a line that contains a segment is tangent to the segment as well as properly intersects it.

Let l_e be the line containing e and let Π_t denote the sweep plane parameterized by $t \in [0, \pi]$ such that Π_t contains the line l_e for all t and $\Pi_0 = \Pi_\pi$. Each plane Π_t intersects the three polytopes **P**, **Q**, and **R** in three, possibly degenerate or empty, convex polygons, P_t, Q_t, and R_t, respectively (see Figure 2.1).

For any t, a *bitangent* to polygons P_t and Q_t is a line tangent to P_t and Q_t in Π_t (the line may intersect the polygon R_t in any way, possibly not at all). For any t, let a *(P_t, Q_t)-tuple* be the unordered set of all supports in **P** and **Q** of one of the bitangents to polygons P_t and Q_t. Note that a support in **P** may be identical to a support in **Q**, in which case the (P_t, Q_t)-tuple does not contain duplicates. Also note that a (P_t, Q_t)-tuple consists of exactly one support in **P** and one support in **Q** (possibly identical) except when the corresponding bitangent is tangent to **P** (or **Q**) along a face (either intersecting the face properly or containing one of its edges); then the (P_t, Q_t)-tuple contains two supports in **P** (or **Q**) instead of one. A **PQ**-*tuple* is a set of edges and vertices that is a (P_t, Q_t)-tuple for some t. We define similarly the *(Q_t, R_t)-tuples* and **QR**-*tuples.*

We say that a (P_t, Q_t)-tuple is *maximal for some t* if it is not contained in any other (P_t, Q_t)-tuple for the same t. Note that a (P_t, Q_t)-tuple is nonmaximal for some t if and only if all its supports intersect Π_t in one and the same point, and P_t and Q_t are not equal to one and the same point (see Figure 2.5(b)).

For any t, let a *(P_t, Q_t, R_t)-tuple* be the union of a (P_t, Q_t)-tuple and a (Q_t, R_t)-tuple that share at least one support in **Q**. A (P_t, Q_t, R_t)-tuple is maximal for some t if it is not contained in any other (P_t, Q_t, R_t)-tuple for the same t. A **PQR**-*tuple* is a set of edges and vertices that is a (P_t, Q_t, R_t)-tuple for some t. Note that a **PQR**-tuple typically consists of three supports, one from each polytope, and consists, in all cases, of at most two supports in **P**, at most three supports in **Q**, and at most two supports in **R**.

A line intersecting e and tangent to **P**, **Q**, **R**, and **S** is called a *generic tangent line* if and only if it intersects **S** only on e and is tangent to P_t, Q_t, and R_t in some plane Π_t. Otherwise it is called a *nongeneric tangent line.* A nongeneric tangent line properly intersects a face of **S** or properly intersects P_t, Q_t, or R_t in some plane Π_t. In the latter case P_t, Q_t, or R_t is a face or an edge of **P**, **Q**, or **R** lying in Π_t; thus a nongeneric tangent line is (in both cases) tangent to **P**, **Q**, **R**, and **S** in a plane containing a face or two edges of these polytopes, a degenerate situation.

In the following three subsections, we bound the number of generic and nongeneric tangent lines. It is helpful to keep in mind that, as observed earlier, two convex polygons in a plane Π_t (such as P_t and Q_t) may admit a linear number of tangents if they intersect.

2.3. Generic tangent lines.

LEMMA 2.1. *The set of supports in* **P**, **Q**, *and* **R** *of a generic tangent line is a* **PQR**-*tuple.*

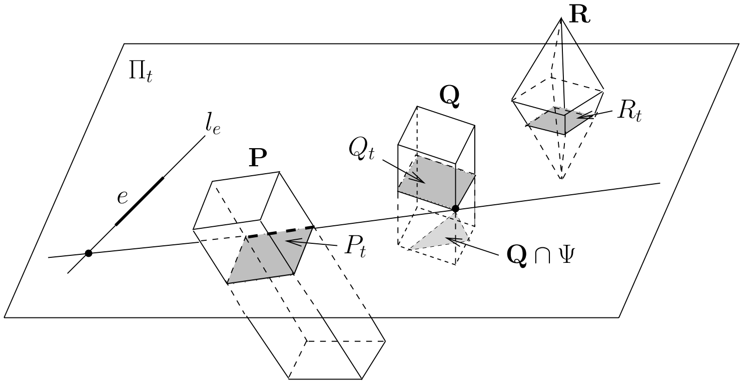


FIG. 2.3. *Plane Π_t is F-critical: it contains a line that lies in a plane Ψ containing a face of $\mathbf{P}$ such that the line is tangent to $\mathbf{Q} \cap \Psi$ at a point not on l_e.*

Proof. Any generic tangent line ℓ is tangent in Π_t to P_t, Q_t, and R_t for some value t. Thus the set of supports of ℓ in $\mathbf{P}$ and $\mathbf{Q}$ (resp., in $\mathbf{Q}$ and $\mathbf{R}$) is a (P_t, Q_t)-tuple (resp., a (Q_t, R_t)-tuple). Moreover, the (P_t, Q_t)-tuple and the (Q_t, R_t)-tuple contain the same supports in $\mathbf{Q}$, and thus their union is a (P_t, Q_t, R_t)-tuple, hence a $\mathbf{PQR}$-tuple. □

We now define the *critical planes* Π_t in such a way that, as we will later prove, the set of (P_t, Q_t, R_t)-tuples is invariant for t ranging strictly between two consecutive critical values. We introduce two types of critical planes: the *V-critical* and *F-critical planes*.

A plane Π_t is *V-critical* if it contains a vertex of $\mathbf{P}$, $\mathbf{Q}$, or $\mathbf{R}$, not on l_e. (The constraint that the vertex does *not* lie on l_e ensures that the number of V-critical planes is finite even in degenerate configurations.) A plane Π_t is *F-critical* relative to an ordered pair of polytopes $(\mathbf{P}, \mathbf{Q})$ if (see Figure 2.3) it contains a line ℓ such that

(i) ℓ lies in a plane $\Psi \neq \Pi_t$ containing a face of $\mathbf{P}$, and

(ii) ℓ is tangent in Ψ to polygon $\mathbf{Q} \cap \Psi$ or $\mathbf{P} \cap \Psi$, at some point not on l_e.

For simplicity, we do not require that ℓ is tangent to $\mathbf{P}$; this leads to overestimating the number of common tangents to $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{S}$ but only by an asymptotically negligible amount. Note that not all lines in Ψ tangent to $\mathbf{Q}$ are tangent to the polygon $\mathbf{Q} \cap \Psi$ when that polygon is a face or edge of $\mathbf{Q}$ lying in Ψ. Note also that we define Π_t to be F-critical when ℓ is tangent to $\mathbf{P} \cap \Psi$ at some point not on l_e only for handling the very degenerate case where $\mathbf{Q} \cap \Psi$ is an edge of $\mathbf{Q}$ and there exists a line in Ψ that properly intersects $\mathbf{Q} \cap \Psi$ and is tangent to $\mathbf{P} \cap \Psi$ along an edge that has an endpoint on l_e (see Figure 2.4). Note finally that if $\ell \in \Pi_t$ satisfies (i) and is tangent, in Ψ, to $\mathbf{P} \cap \Psi$ at some point not on l_e, then polytope $\mathbf{Q}$ plays no role and thus Π_t is F-critical relative to $(\mathbf{P}, \mathbf{Q})$ for all polytopes $\mathbf{Q}$.

F-critical planes relative to $(\mathbf{Q}, \mathbf{P})$, $(\mathbf{Q}, \mathbf{R})$, and $(\mathbf{R}, \mathbf{Q})$ are defined similarly. A plane Π_t is *F-critical* if it is F-critical relative to pairs of polytopes $(\mathbf{P}, \mathbf{Q})$, $(\mathbf{Q}, \mathbf{P})$, $(\mathbf{Q}, \mathbf{R})$, or $(\mathbf{R}, \mathbf{Q})$.

The values of t corresponding to critical planes Π_t are called *critical values*. We call *V-critical* and *F-critical events* the ordered pairs (t, o), where t is a critical value and o is a vertex or line depending on the type of critical event. In a V-critical event, o is a vertex of $\mathbf{P}$, $\mathbf{Q}$, or $\mathbf{R}$ that belongs to $\Pi_t \setminus l_e$. In an F-critical event, o is a line lying in some plane Π_t and satisfying conditions (i)–(ii) above. A *critical event* is a V-critical or F-critical event.

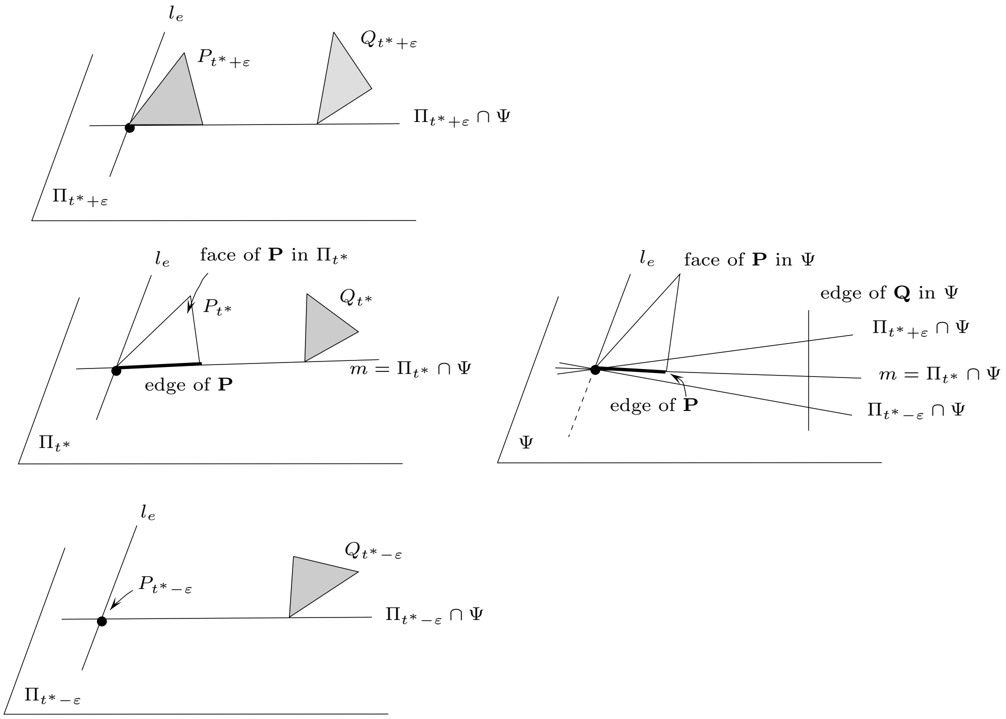


FIG. 2.4. *Plane Π_{t^*} contains a line m such that* (i) *m lies in a plane $\Psi \neq \Pi_{t^*}$ containing a face of* $\mathbf{P}$ *and* (ii) *m is tangent to polygon $\mathbf{P} \cap \Psi$ at some point not on l_e; however, m is not tangent to $\mathbf{Q} \cap \Psi$. If the definition of F-critical planes did not consider such a plane Π_{t^*} to be F-critical, then Lemma* 2.3 *would not hold. Indeed the set u of supports of line $\Pi_{t^*-\varepsilon} \cap \Psi$ is a maximal (P_t, Q_t)-tuple for some but not all t in any open neighborhood of t^*, and, although Π_{t^*} is V-critical, there exists no V-critical event (t^*, v) such that u contains v or an edge with endpoint v.*

LEMMA 2.2. *There are at most $\frac{2}{3}(p+q+r)$ V-critical events and $\frac{8}{3}(p+2\,q+r)$ F-critical events.*

Proof. The number of V-critical events is at most the total number of vertices of $\mathbf{P}$, $\mathbf{Q}$, and $\mathbf{R}$, and hence is less than two-thirds the total number of edges of $\mathbf{P}$, $\mathbf{Q}$, and $\mathbf{R}$. We now count the number of F-critical events relative to polytopes $(\mathbf{P}, \mathbf{Q})$. Let Ψ be a plane containing a face of $\mathbf{P}$, and suppose that for some plane Π_t, line $\ell = \Pi_t \cap \Psi$ satisfies conditions (i)–(ii). Plane Ψ does not contain l_e because otherwise both l_e and ℓ lie in the two distinct planes Ψ and Π_t; thus $\ell = l_e$, but then ℓ cannot satisfy condition (ii). Furthermore, ℓ and l_e intersect or are parallel since they both lie in Π_t. Thus if $\Psi \cap l_e$ is a point, then ℓ contains it, and otherwise $\Psi \cap l_e = \emptyset$ and ℓ is parallel to l_e.

If $\Psi \cap l_e$ is a point, there are at most four candidates for a line ℓ in plane Ψ going through $\Psi \cap l_e$ and tangent to $\mathbf{Q} \cap \Psi$ or $\mathbf{P} \cap \Psi$ at some point not on l_e. Likewise, if $\Psi \cap l_e$ is empty, there are at most four candidates for a line ℓ in plane Ψ that is parallel to l_e and tangent to $\mathbf{Q} \cap \Psi$ or $\mathbf{P} \cap \Psi$. In either case, each candidate line is contained in a unique plane Π_t, for $t \in [0, \pi]$, since $\ell \neq l_e$ (ℓ contains a point not on l_e). Hence, a face of $\mathbf{P}$ generates at most four F-critical events relative to $(\mathbf{P}, \mathbf{Q})$. Therefore the number of critical events relative to $(\mathbf{P}, \mathbf{Q})$ is at most $\frac{8}{3}\,p$ since the number of faces of a polytope is at most two-thirds the number of its edges. Hence

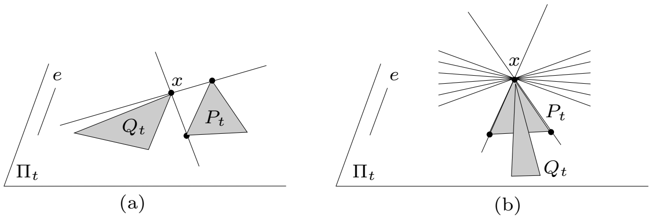


FIG. 2.5. *Lines through x in Π_t and tangent to P_t and Q_t.*

the number of critical events relative to $(\mathbf{P},\mathbf{Q})$, $(\mathbf{Q},\mathbf{P})$, $(\mathbf{Q},\mathbf{R})$, and $(\mathbf{R},\mathbf{Q})$ is at most $\frac{8}{3}(p+2\,q+r)$. □

The following lemma states that the critical planes have the desired property. Let u_e be the set of supports of l_e in $\mathbf{P}$ and $\mathbf{Q}$ and let u denote some (P_t, Q_t)-tuple.

LEMMA 2.3. *Let t^* be the endpoint of a maximal interval*[3] *throughout which $u \neq u_e$ is a maximal (P_t, Q_t)-tuple. Then t^* is a critical value. Moreover, there exists a V-critical event (t^*, v) or an F-critical event (t^*, m) such that u contains v or an edge with endpoint v, or u is contained in the set of supports of m.*

The proof of this lemma is rather long and intricate; we postpone it to section 2.4. Note that, as stated, this lemma applies only under the assumptions that u is maximal and distinct from u_e. These assumptions are made in order to simplify the proof of Lemma 2.3; we do not suggest that the lemma is false without them.

LEMMA 2.4. *Any edge or vertex of $\mathbf{P}$ or $\mathbf{Q}$ is in at most two $\mathbf{PQ}$-tuples that are maximal (P_t, Q_t)-tuples for all t in any given nonempty interval*[3] *of $\mathbb{R}/\pi\mathbb{Z}$.*

Proof. Let $\tilde{t}$ be an element of a nonempty interval $\mathcal{I}$ of $\mathbb{R}/\pi\mathbb{Z}$ and x be an edge or vertex of $\mathbf{P}$ or $\mathbf{Q}$. If x does not intersect $\Pi_{\tilde{t}}$, then no $(P_{\tilde{t}}, Q_{\tilde{t}})$-tuple contains x. If x intersects $\Pi_{\tilde{t}}$ in one point, then there are, in general, at most two lines in $\Pi_{\tilde{t}}$ going through x and tangent to $P_{\tilde{t}}$ and $Q_{\tilde{t}}$ (see Figure 2.5(a)); in all cases there are at most three $(P_{\tilde{t}}, Q_{\tilde{t}})$-tuples containing x (see Figure 2.5(b)); however, at most two of them are maximal. If x intersects $\Pi_{\tilde{t}}$ in more than one point, x is an edge lying in $\Pi_{\tilde{t}}$. Then any line in $\Pi_{\tilde{t}}$ intersecting x and tangent to $P_{\tilde{t}}$ and $Q_{\tilde{t}}$ contains an endpoint of x, and thus x belongs to no $(P_{\tilde{t}}, Q_{\tilde{t}})$-tuple.

Hence at most two $\mathbf{PQ}$-tuples contain x and are maximal (P_t, Q_t)-tuples for $t=\tilde{t}$, and thus at most two $\mathbf{PQ}$-tuples contain x and are maximal (P_t, Q_t)-tuples for all t in $\mathcal{I}$. □

LEMMA 2.5. *There are at most $O(p+q+r)$ $\mathbf{PQR}$-tuples.*

Proof. In order to count the number of distinct (P_t, Q_t, R_t)-tuples, we charge each maximal (P_t, Q_t, R_t)-tuple to a critical event. We then show that each critical event is charged at most a constant number of times. It then follows from Lemma 2.2 that there are $O(p+q+r)$ distinct maximal (P_t, Q_t, R_t)-tuples. A maximal (P_t, Q_t, R_t)-tuple consists of at most two supports in $\mathbf{P}$, at most three supports in $\mathbf{Q}$, and at most two supports in $\mathbf{R}$, and thus contains at most $(2^2-1)(2^3-1)(2^2-1)$ distinct subsets with at least one support in each of $\mathbf{P}$, $\mathbf{Q}$, and $\mathbf{R}$. Each maximal (P_t, Q_t, R_t)-tuple thus contains at most a constant number of distinct (P_t, Q_t, R_t)-tuples, which implies the result.

Let s be a maximal (P_t, Q_t, R_t)-tuple and let $\mathcal{I}$ be any maximal connected subset of $\mathbb{R}/\pi\mathbb{Z}$ such that s is a maximal (P_t, Q_t, R_t)-tuple for all $t \in \mathcal{I}$. Let u be a maximal

[3] Such an interval could be open or closed, a single point, or an interval of positive length.

(P_t, Q_t)-tuple and u' a maximal (Q_t, R_t)-tuple such that the union of u and u' is s and such that u and u' share at least one support in $\mathbf{Q}$.

First, suppose that $\mathcal{I} = \mathbb{R}/\pi\mathbb{Z}$. Then u is a maximal (P_t, Q_t)-tuple for all $t \in \mathbb{R}/\pi\mathbb{Z}$. Thus each support in u intersects Π_t for all $t \in \mathbb{R}/\pi\mathbb{Z}$ and thus intersects l_e; moreover, each support in u intersects Π_t only on l_e for all $t \in \mathbb{R}/\pi\mathbb{Z}$ except possibly for one value of t. Since $\mathbf{P}$ and $\mathbf{Q}$ have nonempty interior, $P_t \cup Q_t$ is not reduced to a point for all t in some interval of positive length. For all t in such an interval, since u is maximal, the union of the supports in u intersects Π_t in at least two distinct points. These at least two distinct points lie on l_e for some values of t by the above argument. Thus, for these values of t, l_e is the only line in Π_t whose set of supports contains u. Hence u is the set of supports of l_e. The same property holds for v, and thus s is also the set of supports of l_e. We can thus assume in the following that $\mathcal{I} \neq \mathbb{R}/\pi\mathbb{Z}$ and only count the maximal (P_t, Q_t, R_t)-tuples that are not the set of supports of l_e.

Interval $\mathcal{I}$ is thus a nonempty interval of $\mathbb{R}/\pi\mathbb{Z}$; it can be open or closed, a single point, or an interval of positive length. Let w_0 and w_1 denote the endpoints of $\mathcal{I} \neq \mathcal{R}/\pi\mathbb{Z}$.

If s contains a vertex v, or an edge with endpoint v such that v lies in $\Pi_{w_i} \setminus l_e$ for $i = 0$ or 1, then we charge s to the V-critical event (w_i, v). Otherwise, we charge s to an F-critical event (w_i, m) where m is a line in Π_{w_i} whose set of supports contains u or u'. Such a V-critical or F-critical event exists by Lemma 2.3.

We now prove that each critical event is charged by at most a constant number of distinct maximal (P_t, Q_t, R_t)-tuples. As mentioned before, that will imply the result.

Consider a V-critical event (t^*, v) that is charged by a maximal (P_t, Q_t, R_t)-tuple s. By the charging scheme, s contains a support x that is v or an edge with endpoint v, and s is a maximal (P_t, Q_t, R_t)-tuple for all t in at least one of three intervals, $\{t^*\}$ and two open intervals having t^* as endpoint; denote these intervals by $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3$.

By Lemma 2.4, at most two $\mathbf{PQ}$-tuples contain x and are maximal (P_t, Q_t)-tuples for all t in $\mathcal{I}_i$. Moreover, each of these $\mathbf{PQ}$-tuples contains at most two supports in $\mathbf{Q}$, and each of these supports belongs to at most two $\mathbf{QR}$-tuples that are maximal (Q_t, R_t)-tuples for all t in $\mathcal{I}_i$. Thus at most eight $\mathbf{PQR}$-tuples contain x and are maximal (P_t, Q_t, R_t)-tuples for all t in $\mathcal{I}_i$, for each $i = 1, \dots, 3$. Hence any V-critical event (t^*, v) is charged by at most 24 distinct maximal (P_t, Q_t, R_t)-tuples.

Consider now an F-critical event (t^*, m) that is charged by a maximal (P_t, Q_t, R_t)-tuple s, and define as before u and u'. By the charging scheme, the set of supports of m contains u or u' (or both); suppose without loss of generality that it contains u. The set of supports of m contains at most two supports in $\mathbf{P}$ and at most two supports in $\mathbf{Q}$. Since u contains at least one support in $\mathbf{P}$ and at least one support in $\mathbf{Q}$, there are at most 3^2 choices for u.

By the charging scheme, s is a maximal (P_t, Q_t, R_t)-tuple for all t in at least one of three intervals, $\{t^*\}$ and two open intervals having t^* as endpoint; denote these intervals by $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3$. It follows from Lemma 2.4 that, for each support x of $\mathbf{Q}$ in u, at most two $\mathbf{QR}$-tuples contain x and are maximal (Q_t, R_t)-tuples for all t in $\mathcal{I}_i$. There are at most 3^2 choices for u (as shown above), 2 for x, 3 for i, and 2 for the $\mathbf{QR}$-tuples containing x. Hence any F-critical event (t^*, m) is charged by at most $2^2 \times 3^3$ distinct maximal (P_t, Q_t, R_t)-tuples.

Therefore each critical event is charged by at most a constant number of distinct maximal (P_t, Q_t, R_t)-tuples, which concludes the proof. □

COROLLARY 2.6. *There are at most $O(p + q)$* $\mathbf{PQ}$*-tuples.*

Proof. Replace $\mathbf{R}$ by a copy of $\mathbf{Q}$ in Lemma 2.5. Any $\mathbf{PQ}$-tuple is also a $\mathbf{PQQ}$-

tuple, and there are at most $O(p+q+q) = O(p+q)$ of these. □

PROPOSITION 2.7. *There are $O(p+q+r)$ isolated generic tangent lines.*

Proof. A generic tangent line is transversal to e and to the edges and vertices of a **PQR**-tuple, by definition and Lemma 2.1. An isolated generic tangent line is thus an isolated transversal with respect to a set of edges and vertices that consists of a **PQR**-tuple and either edge e or one or both of its endpoints. The number of such sets is four times the number of **PQR**-tuples, which is in $O(p+q+r)$ by Lemma 2.5. The result follows since each such set consists of at most eight edges and vertices (at most two supports from each of the four polytopes) and thus admits at most eight isolated transversals [9]. □

2.4. Proof of Lemma 2.3. Recall that u_e denotes the set of supports of l_e in **P** and **Q**, and that Lemma 2.3 states the following.

> *Let t^* be the endpoint of a maximal interval throughout which $u \neq u_e$ is a maximal (P_t, Q_t)-tuple. Then t^* is a critical value. Moreover, there exists a V-critical event (t^*, v) or an F-critical event (t^*, m) such that u contains v or an edge with endpoint v, or u is contained in the set of supports of m.*

We can assume that u *contains no vertex v and no edge with endpoint v such that v lies on* $\Pi_{t^*} \setminus l_e$ because otherwise (t^*, v) is a V-critical event such that u contains v or an edge with endpoint v, which concludes the proof.

We prove a series of lemmas that yields Lemma 2.3. Indeed, we prove the existence of a line m in Π_{t^*} whose set of supports contains u (Lemma 2.10) such that (i) m lies in a plane $\Psi \neq \Pi_{t^*}$ containing a face of **P** (Lemma 2.11) and (ii) m is tangent in Ψ to polygon $\mathbf{Q} \cap \Psi$ or $\mathbf{P} \cap \Psi$, at some point not on l_e (Lemma 2.12). This proves that Π_{t^*} contains a line m whose set of supports contains u and such that (t^*, m) is an F-critical event, which concludes the proof.

By hypothesis, for any sufficiently small open neighborhood $\mathcal{N}$ of t^* whose endpoints are denoted by t_0 and t_1, u is not a maximal (P_t, Q_t)-tuple for some $t \in \mathcal{N}$ and u is a maximal (P_t, Q_t)-tuple for $t = t^*$ or for all $t \in (t^*, t_1)$ (or by symmetry for all $t \in (t_0, t^*)$).

We only consider in the following supports in **P** and in **Q**; polytope **R** plays no role. We start by proving two preliminary lemmas.

LEMMA 2.8. *Each support in u intersects Π_t in exactly one point (possibly on l_e) for all t in any sufficiently small open neighborhood $\mathcal{N}$ of t^*.*

Moreover, the union of all supports in u intersects Π_t in at least two distinct points for all $t \neq t^*$ in $\mathcal{N}$. This property also holds for $t = t^*$ if u is a maximal (P_{t^*}, Q_{t^*})-tuple.

Proof. Since u is a (P_t, Q_t)-tuple for some t in every open neighborhood of t^*, each support in u intersects Π_t for some t in every open neighborhood of t^*. It thus follows from the assumption that u contains no vertex v and no edge with endpoint v, such that v lies on $\Pi_{t^*} \setminus l_e$, that each support in u intersects Π_t for all t in any sufficiently small open neighborhood $\mathcal{N}$ of t^*. It follows that each support in u either lies in l_e or intersects Π_t in exactly one point for all $t \in \mathcal{N}$. However, no edge of u lies in l_e because otherwise if x denotes such an edge of, say, **P**, then any line tangent to P_t in Π_t and intersecting x contains an endpoint of x which is a vertex of **P**; thus, by definition, u does not contain x but one of its endpoints. Hence each support of u intersects Π_t in exactly one point for all $t \in \mathcal{N}$.

We now prove that the union of the supports in u intersects Π_t in at least two distinct points for any $t \in \mathcal{N}$ such that u is a maximal (P_t, Q_t)-tuple. Suppose for a contradiction that the union of the supports in u intersects Π_t in one single point v for some $t \in \mathcal{N}$ such that u is a maximal (P_t, Q_t)-tuple. Then polygons P_t and Q_t are both reduced to point v because otherwise u is not maximal (otherwise, a line in Π_t tangent to P_t and Q_t at v can be rotated about v until it becomes tangent to P_t or Q_t at some other points). Thus $v = P_t = Q_t$ is a vertex of $\mathbf{P}$ and of $\mathbf{Q}$ because the polytopes have nonempty interior. Hence $u = \{v\}$ because each support in u contains v. It follows that v lies on l_e since each support in u intersects Π_t for all $t \in \mathcal{N}$. Moreover, since P_t and Q_t are both reduced to point $v = l_e \cap \mathbf{P} = l_e \cap \mathbf{Q}$, the set u_e of supports of l_e is u, contradicting the hypotheses of Lemma 2.3.

Thus, if u is a maximal (P_t, Q_t)-tuple for all $t \in (t^*, t_1)$, the union of the supports in u intersects Π_t in at least two distinct points for all $t \in (t^*, t_1)$ and thus for all $t \neq t^*$ in any sufficiently small open neighborhood of t^*. Also, if u is a maximal (P_t, Q_t)-tuple for $t = t^*$, the union of the supports in u intersects Π_t in at least two distinct points for $t = t^*$ and thus for all t in any sufficiently small open neighborhood of t^*. ◻

Lemma 2.9. *If u is a maximal (P_{t^*}, Q_{t^*})-tuple, then u consists of at least three supports.*

Proof. Note that it follows from Lemma 2.8 that u contains at least two supports. Suppose for a contradiction that u consists of only two supports. By Lemma 2.8, they intersect Π_t in exactly two distinct points for all t in any sufficiently small open neighborhood $\mathcal{N}$ of t^*. Thus there exists for all $t \in \mathcal{N}$ a unique line m_t in Π_t whose set of supports contains u; moreover, m_t is continuous in terms of t. Since u is a maximal (P_{t^*}, Q_{t^*})-tuple, the set of supports of m_{t^*} is u. Thus, for all t in any sufficiently small $\mathcal{N}$, the set of supports of m_t is u. Thus the set of supports of m_t is invariant for $t \in \mathcal{N}$ and since m_{t^*} is tangent to P_{t^*} and Q_{t^*}, line m_t is tangent to P_t and Q_t for all $t \in \mathcal{N}$.

Hence, for all $t \in \mathcal{N}$, line m_t, whose set of supports is u, is tangent to P_t and Q_t in Π_t. Thus u is a maximal (P_t, Q_t)-tuple for all $t \in \mathcal{N}$. Moreover, m_t is the unique line in Π_t whose set of supports contains u, and thus u is a maximal (P_t, Q_t)-tuple for all $t \in \mathcal{N}$, contradicting the hypotheses of the lemma. ◻

Lemma 2.10. *There exists a line m in Π_{t^*} whose set of supports contains u that is tangent to P_{t^*} and Q_{t^*} along an edge of one of them, say of P_{t^*}.*

Proof. Consider first the case where u is a maximal (P_{t^*}, Q_{t^*})-tuple. There exists in Π_{t^*} a line m tangent to P_{t^*} and Q_{t^*} whose set of supports is u. By Lemma 2.9, the set u of supports of m contains at least three supports, and hence at least two supports in $\mathbf{P}$ (or in $\mathbf{Q}$). Furthermore, the supports of m in one polytope intersect Π_{t^*} in distinct points (by definition of supports). Thus m intersects P_{t^*} (or Q_{t^*}) in at least two distinct points and is tangent to P_{t^*} and Q_{t^*}. The result follows since P_{t^*} (and Q_{t^*}) is convex.

Consider now the case where u is a maximal (P_t, Q_t)-tuple for all $t \in (t^*, t_1)$. Then, for all $t \in (t^*, t_1)$, there exists a line in Π_t tangent to P_t and Q_t and whose set of supports is u. Moreover, by Lemma 2.8, this line is unique for each $t \in (t^*, t_1)$ and varies continuously in terms of $t \in (t^*, t_1)$. When t tends to t^*, the line tends to a line m_{t^*} in Π_{t^*} which is tangent to P_{t^*} and Q_{t^*} and whose set of supports contains u. If its set of supports strictly contains u, then m_{t*} is tangent to P_{t^*} and Q_{t^*} along an edge of one of them because the polygons are convex, and hence we can choose $m = m_{t^*}$ to complete the proof. Otherwise, u is a (P_{t^*}, Q_{t^*})-tuple.

We can suppose that u is a nonmaximal (P_{t^*}, Q_{t^*})-tuple since we already treated the case where u is maximal. There exists in Π_{t^*} a line tangent to P_{t^*} and Q_{t^*} whose

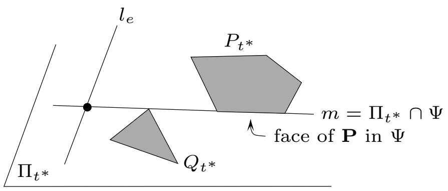


FIG. 2.6. *Line m is tangent to* **P** *along a face in plane* $\Psi \neq \Pi_{t^*}$.

set of supports is u. Since u is nonmaximal, this line is tangent to P_{t^*} and Q_{t^*} at a shared vertex and can be rotated about this vertex in Π_{t^*} until it becomes tangent to P_{t^*} and Q_{t^*} at some other points, which must occur because u is nonmaximal; let m denote the resulting line. The set of supports of m contains u, and m is tangent to P_{t^*} and Q_{t^*} along an edge of one of them because the polygons are convex. □

LEMMA 2.11. *Line m lies in a plane $\Psi \neq \Pi_{t^*}$ containing a face of* **P**.

Proof. By Lemma 2.10, m contains an edge of P_{t^*}; see Figure 2.6. This edge either intersects the relative interior of some face of **P**, in which case we take Ψ to be the plane containing that face, or this edge is an edge of **P**, in which case we take Ψ to be a plane, different from Π_{t^*}, containing one of the two faces of **P** incident to that edge. □

Let m_t be the line $\Psi \cap \Pi_t$ for all t in any sufficiently small open neighborhood $\mathcal{N}$ of t^*; line m_t is well defined since $\Psi \cap \Pi_{t^*}$ is line m by Lemmas 2.10 and 2.11.

LEMMA 2.12. *Line m is tangent to $\mathbf{P} \cap \Psi$ or to $\mathbf{Q} \cap \Psi$ at some point not on l_e.*

Proof. We assume for a contradiction that line m does not satisfy the lemma; i.e., m is not tangent to $\mathbf{P} \cap \Psi$ or to $\mathbf{Q} \cap \Psi$ at any point other than on l_e. We prove that the set of supports of m is u and is a maximal (P_t, Q_t)-tuple for all t in any sufficiently small neighborhood of t^*, contradicting the hypotheses of Lemma 2.3 and thus proving Lemma 2.12.

Since m is tangent to **Q** (by Lemma 2.10), m is tangent to $\mathbf{Q} \cap \Psi$ only on l_e (see Figure 2.7(a)), or m properly intersects $\mathbf{Q} \cap \Psi$ which is then a face or an edge of **Q** (see Figure 2.7(b)).[4] Similarly m is tangent to $\mathbf{P} \cap \Psi$ only on l_e, or m properly intersects it; however, $\mathbf{P} \cap \Psi$ is necessarily a face of **P** by Lemma 2.11.

Lemmas 2.13 and 2.14 which follow imply that the set of supports of m_t is invariant and equal to u for all t in any sufficiently small open neighborhood $\mathcal{N}$ of t^*. Moreover, since m_t varies continuously with t and $m = m_{t^*}$ is tangent to P_{t^*} and Q_{t^*} (by Lemma 2.10), line m_t is tangent to P_t and Q_t for all $t \in \mathcal{N}$. Hence u is a (P_t, Q_t)-tuple for all $t \in \mathcal{N}$. We now prove that u is a maximal (P_t, Q_t)-tuple for all $t \in \mathcal{N}$.

As we have seen before, $m = m_{t^*}$ is tangent to **P** in at least two points (by Lemma 2.10); thus m_{t^*} intersects its supports in at least two distinct points. Moreover, the set of supports of m_{t^*} is u. Thus there is a unique line in Π_{t^*} whose set of supports contains u. Hence u is a maximal (P_{t^*}, Q_{t^*})-tuple.

By Lemma 2.8, m_t is the unique line in Π_t whose set of supports contains u for all $t \neq t^*$ in $\mathcal{N}$. Thus u is a maximal (P_t, Q_t)-tuple for all $t \neq t^*$ in $\mathcal{N}$.

[4] Note that in these two situations, two edges of two distinct polytopes are then coplanar (in the first case an edge of **Q** and e are coplanar, and in the latter case a face of **P** is coplanar with a face or an edge of **Q**). Hence proving this lemma is straightforward under some general position assumption that excludes such situations.

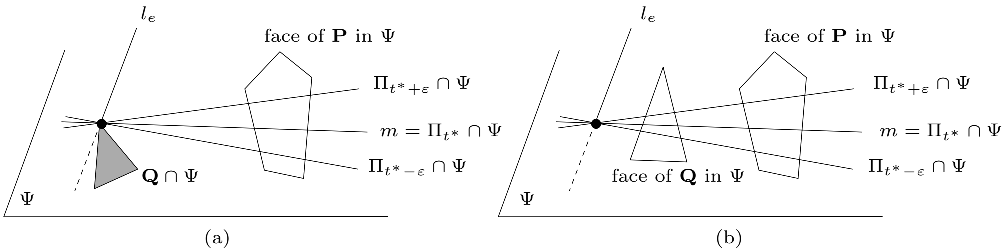

l_e
face of **P** in Ψ
$\Pi_{t^*+\varepsilon} \cap \Psi$
$m = \Pi_{t^*} \cap \Psi$
$\Pi_{t^*-\varepsilon} \cap \Psi$
$\mathbf{Q} \cap \Psi$
Ψ
(a)
l_e
face of **P** in Ψ
$\Pi_{t^*+\varepsilon} \cap \Psi$
$m = \Pi_{t^*} \cap \Psi$
$\Pi_{t^*-\varepsilon} \cap \Psi$
face of **Q** in Ψ
Ψ
(b)

FIG. 2.7. *m is tangent to $\mathbf{P}$ along a face in Ψ and* (a) *to $\mathbf{Q} \cap \Psi$ only on l_e or* (b) *to $\mathbf{Q}$ along a face in Ψ.*

Hence u is a maximal (P_t, Q_t)-tuple for all $t \in \mathcal{N}$, contradicting the hypotheses of Lemma 2.3 and thus concluding the proof of Lemma 2.12. □

LEMMA 2.13. *The set of supports of m_t is u for some t in any sufficiently small open neighborhood $\mathcal{N}$ of t^*.*

Proof. We first prove that the supports in u are supports of m_t for all $t \in \mathcal{N}$. A support vertex in u lies on l_e by Lemma 2.8 and thus lies in Π_t for all t. A support vertex in u also lies on m by Lemma 2.10 and thus lies in plane Ψ by Lemma 2.11. Hence, for all $t \in \mathcal{N}$, the support vertices in u lie on m_t and thus are supports of m_t.

In order to prove that the support edges in u are supports of m_t, it is sufficient (by Lemma 2.10) to prove that the support edges of m are supports of m_t. The support edges of m in $\mathbf{P}$ lie in plane Ψ (see Figure 2.7(b)) because Ψ contains m and a face of $\mathbf{P}$ (indeed if m intersects an edge of $\mathbf{P}$ not in Ψ, then m contains one of its endpoints, and thus the edge is not a support). Thus all the support edges of m lie in Ψ and m contains none of their endpoints (by definition). Since m_t lies in Ψ for all t and $m_{t^*} = m$, line m_t intersects all the support edges of m and contains none of their endpoints for all t in any sufficiently small open neighborhood $\mathcal{N}$ of t^*. Hence the support edges of m in $\mathbf{P}$ are supports of m_t for all $t \in \mathcal{N}$.

Consider the case where $\mathbf{Q} \cap \Psi$ is a face or an edge of $\mathbf{Q}$. Similarly as for $\mathbf{P}$, the support edges of m in $\mathbf{Q}$ lie in plane Ψ and thus are supports of m_t for all $t \in \mathcal{N}$.

Consider now the case where m is tangent to $\mathbf{Q} \cap \Psi$ only on l_e at, say, point v (see Figure 2.7(a)). Then v lies in Ψ (since $m \subset \Psi$ by Lemma 2.11) and also lies in Π_t for all t (since $l_e \subset \Pi_t$ for all t). Hence m_t contains v for all $t \in \mathcal{N}$. Moreover, m_t is tangent to $\mathbf{Q} \cap \Psi$ only at v for all t in any sufficiently small open neighborhood $\mathcal{N}$ of t^*. Hence the set of supports of m_t in $\mathbf{Q}$ is invariant for all $t \in \mathcal{N}$.

We have so far proved that the set of supports of m_t contains u for all $t \in \mathcal{N}$.

We now prove that the set of supports of m_t is u for some $t \in \mathcal{N}$. Consider first the case where u is a maximal (P_{t^*}, Q_{t^*})-tuple. Then, by Lemma 2.8, the union of the supports in u intersects Π_{t^*} in at least two distinct points; thus $m_{t^*} = m$ is the only line in Π_{t^*} whose set of supports contains u. Moreover, since u is a (P_{t^*}, Q_{t^*})-tuple, there exists a line in Π_{t^*} whose set of supports is u. Hence the set of supports of m_{t^*} is u.

Consider now the case where u is a maximal (P_t, Q_t)-tuple for all $t \in (t^*, t_1)$. By Lemma 2.8, for all $t \in (t^*, t_1)$, the union of the supports in u intersects Π_t in at least two distinct points; thus m_t is the only line in Π_t whose set of supports contains u. For all $t \in (t^*, t_1)$, since u is a (P_t, Q_t)-tuple there exists a line in Π_t whose set of supports is u. Hence the set of supports of m_t is u for all $t \in (t^*, t_1)$. □

Lemma 2.14. *The set of supports of m_t is invariant for t ranging in any sufficiently small open neighborhood $\mathcal{N}$ of t^*.*

Proof. First if $m = l_e$, then $m_t = l_e$ for all $t \in \mathcal{N}$ because Ψ contains $m = l_e$ (by Lemma 2.11), and Π_t contains l_e for all t (by definition). Thus the set of supports of m_t is invariant for all $t \in \mathcal{N}$. We now assume that $m \neq l_e$.

Line m is tangent to polygon P_{t^*} along an edge by Lemma 2.10. Thus m is tangent to $\mathbf{P}$ in at least two points. Hence, since $\mathbf{P} \cap \Psi$ is a face of $\mathbf{P}$ and m lies in Ψ, either m properly intersects $\mathbf{P} \cap \Psi$ or m is tangent to $\mathbf{P} \cap \Psi$ along one of its edges. In the latter case, the edge does not lie in l_e since $m \neq l_e$; thus m is tangent to $\mathbf{P} \cap \Psi$ at some point not on l_e, contradicting our assumptions. Hence m properly intersects the face of $\mathbf{P}$ in Ψ.

It follows that, if m contains a vertex of $\mathbf{P}$, then this vertex is an endpoint of a support edge of m_t for all t in any sufficiently small open neighborhood of t^* (indeed m_t lies in Ψ and tends to m when t tends to t^*). By Lemma 2.13, the set of supports of m_t is u for some t in any sufficiently small open neighborhood of t^*. Hence, if m contains a vertex of $\mathbf{P}$, this vertex is an endpoint of a support edge in u. By assumption u contains no edge with endpoint on $\Pi_{t^*} \setminus l_e$; thus m contains no vertex of $\mathbf{P}$ except possibly on l_e (since m lies in Π_{t^*}). It thus follows that the set of supports of m_t in $\mathbf{P}$ is invariant for t ranging in any sufficiently small open neighborhood of t^* (since $m_t \subset \Psi$ tends to m when t tends to t^* and all supports of m lie in Ψ).

Now consider the case where m properly intersects $\mathbf{Q} \cap \Psi$ which is a face or an edge of $\mathbf{Q}$. Similarly as for $\mathbf{P}$, m contains no vertex of $\mathbf{Q}$ except possibly on l_e, and thus the set of supports of m_t in $\mathbf{Q}$ is invariant for t ranging in any sufficiently small open neighborhood of t^*.

Finally, consider the case where m is tangent to $\mathbf{Q} \cap \Psi$ only on l_e. Then, as in the proof of Lemma 2.13, the set of supports of m_t in $\mathbf{Q}$ is invariant for all t ranging in any sufficiently small open neighborhood of t^*, which concludes the proof. □

2.5. Nongeneric tangent lines. We count here the number of nongeneric tangent lines. Note that, as mentioned before, there are no such lines under some adequate general position assumption.

Proposition 2.15. *There are at most $O(p + q + r)$ isolated nongeneric tangent lines except possibly those that lie in planes that contain e and are tangent to all four polytopes.*

Proof. An isolated nongeneric tangent line lies in plane Π_t for some t and contains (at least) two distinct points, each of which is a vertex of $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{R}$, or $\mathbf{S}$, or a point of tangency between the line and one of the polygons P_t, Q_t, and R_t; indeed, otherwise the line can be moved in Π_t while keeping the same supports.

We count first the isolated nongeneric tangent lines that contain two distinct points of tangency with two of the polygons P_t, Q_t, and R_t in Π_t for some t. Consider such a line ℓ tangent to, say, P_t and Q_t in Π_t. Line ℓ is nongeneric and thus properly intersects a face of $\mathbf{S}$ or a face or an edge of $\mathbf{R}$ lying in Π_t. If ℓ properly intersects a face of $\mathbf{S}$ or a face or an edge of $\mathbf{R}$ lying in Π_t but not entirely contained in l_e, then Π_t is one of the at most four planes tangent to $\mathbf{R}$ or $\mathbf{S}$. There are $O(p + q)$ lines tangent to P_t and Q_t in two distinct points in each of these planes and thus $O(p+q)$ such lines in total. Otherwise, Π_t intersects each of $\mathbf{R}$ and $\mathbf{S}$ in an edge contained in l_e. The supports of ℓ are thus the union of a $\mathbf{PQ}$-tuple and, in each of $\mathbf{R}$ and $\mathbf{S}$, the edge lying in l_e or one (or both) of its endpoints. It follows that at most a constant number of such isolated nongeneric tangent lines contain a given $\mathbf{PQ}$-tuple in their set of supports. Hence the number of such lines is at most the number of $\mathbf{PQ}$-tuples, which is in $O(p+q)$ by Corollary 2.6. It follows that there are at most $O(p+q+r)$ isolated nongeneric tangent lines that contain two distinct points of tangency with two of the polygons P_t, Q_t, and R_t in Π_t for some t. We obtain similarly that there are at most $O(p+q+r)$ isolated nongeneric tangent lines that contain two distinct points of tangency with only one of the polygons P_t, Q_t, and R_t.

We now count the isolated nongeneric tangent lines that contain a unique vertex of $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{R}$, or $\mathbf{S}$ and a unique point of tangency with the polygons P_t, Q_t, and R_t in Π_t for some t. Each vertex v of $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{R}$, or $\mathbf{S}$ that does not lie on l_e is contained in a unique plane Π_t, and there are, in that plane, at most six lines through v and tangent to P_t, Q_t, or R_t. There are thus $O(p+q+r)$ such lines in total. Consider now a line ℓ through a vertex v on l_e and tangent to P_t at $w \neq v$ in Π_t for some t. We can suppose that each of Q_t and R_t is either tangent to ℓ at w or is properly intersected by ℓ; indeed otherwise ℓ is tangent to two polygons in two distinct points. If Q_t (or R_t) is a face of $\mathbf{Q}$ (resp., $\mathbf{R}$) or an edge not contained in l_e, then Π_t is one of the at most two planes tangent to $\mathbf{Q}$ (resp., $\mathbf{R}$) and, in each of these planes, there are at most two lines through v and tangent to P_t. If Q_t (or R_t) is tangent to ℓ at w such that the support edges of ℓ in $\mathbf{P}$ and in $\mathbf{Q}$ (resp., $\mathbf{R}$) are not collinear, then ℓ goes through a vertex of $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{R}$, or $\mathbf{S}$ that lies on l_e, and through a vertex of the intersection of two of these polytopes. There are at most eight vertices of $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{S}$ on l_e and $O(p+q+r)$ vertices on the intersection of two of these polytopes. There are thus $O(p+q+r)$ such lines in total. Otherwise, Q_t (and R_t) is an edge contained in l_e or is tangent to ℓ at w such that the support edges of ℓ in $\mathbf{P}$ and in $\mathbf{Q}$ (resp., $\mathbf{R}$) are collinear; then ℓ is not isolated.

We finally bound the number of isolated nongeneric tangent lines that contain no point of tangency with the polygons P_t, Q_t, and R_t in Π_t for any t (and thus contain at least two vertices of $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{S}$). Consider such a line ℓ that lies in plane Π_t for some t. Line ℓ is tangent to $\mathbf{P}$, $\mathbf{Q}$, and $\mathbf{R}$ and thus properly intersects P_t, Q_t, and R_t in plane Π_t which is tangent to $\mathbf{P}$, $\mathbf{Q}$, and $\mathbf{R}$. If plane Π_t is not tangent to $\mathbf{S}$, ℓ goes through an endpoint of e (since ℓ is tangent to $\mathbf{S}$), and there are $O(p+q+r)$ such lines ℓ that go through an endpoint of e and at least another vertex of $\mathbf{P}$, $\mathbf{Q}$, or $\mathbf{R}$. If plane Π_t is tangent to $\mathbf{S}$, line ℓ lies in a plane Π_t tangent to $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{S}$, which concludes the proof. □

Note that there can be $\Omega(n^2)$ isolated nongeneric tangent lines that lie in a plane tangent to all four polytopes. Consider, for instance, four polytopes that admit a common tangent plane containing edge e, an edge e' of $\mathbf{P}$, and two faces of $\mathbf{Q}$ and $\mathbf{R}$ of linear complexity such that all the lines through a vertex of each face intersect e and e'. All these lines are isolated nongeneric tangent lines.

2.6. Proof of the main lemma. Proposition 2.7, which handles the isolated generic tangent lines, and Proposition 2.15, which handles the isolated nongeneric tangent lines, directly yield the Main Lemma.

3. Upper bounds. We prove in this section the upper bounds of Theorems 1.1, 1.2, and 1.3. The lower bounds are proved in section 4. Consider k pairwise distinct polytopes $\mathbf{P}_1, \dots, \mathbf{P}_k$ with $n_1, \dots, n_k$ edges, respectively, and n edges in total.

LEMMA 3.1. *For any edge e of $\mathbf{P}_i$, there are $O(n_j + n_l + n_m)$ sets of open edges, chosen from $\mathbf{P}_i$, $\mathbf{P}_j$, $\mathbf{P}_l$, and $\mathbf{P}_m$, that admit an isolated transversal that intersects e and is tangent to these four polytopes.*

Proof. Any isolated transversal to a set of edges is isolated with respect to the set of all its supports. It is thus sufficient to bound the number of sets of open edges, chosen from $\mathbf{P}_i$, $\mathbf{P}_j$, $\mathbf{P}_l$, and $\mathbf{P}_m$, that are intersected by an isolated line that intersects e and is tangent to these four polytopes. The Main Lemma states that there are $O(n_j + n_l + n_m)$ isolated lines intersecting e and tangent to $\mathbf{P}_i$, $\mathbf{P}_j$, $\mathbf{P}_l$, and $\mathbf{P}_m$, excluding those that lie in planes that contain e and are tangent to all four polytopes. Any of these $O(n_j + n_l + n_m)$ isolated lines intersects at most two open edges in any polytope. Thus there are $O(n_j + n_l + n_m)$ sets of open edges (chosen from $\mathbf{P}_i$, $\mathbf{P}_j$, $\mathbf{P}_l$, and $\mathbf{P}_m$) that are intersected by one of these isolated lines. Now consider any isolated line that lies in a plane that contains e and is tangent to all four polytopes. This plane contains all the open edges that are intersected by the isolated line. Thus these edges (and any subset of them) admit no isolated transversal. □

LEMMA 3.2. *A minimal set of open edges and vertices that admit an isolated transversal consists of* (i) *two vertices,* (ii) *one vertex and one or two edges, or* (iii) *two, three, or four edges.*

Proof. Consider a minimal set of open edges and vertices that admit an isolated transversal. The elements are necessarily distinct because the set is minimal. If the set contains two vertices, it contains no other element since the two vertices admit a unique transversal.

Suppose now that the set contains one vertex. None of the open edges contains the vertex because otherwise such an edge would be redundant. Thus, the vertex and any segment define either a line, and thus admit an isolated transversal, or they define a plane. If none of the other edges intersects that plane in a unique point, the vertex and all open edges admit zero or infinitely many common transversals, a contradiction. Thus there exists an edge that intersects the plane in a unique point. Hence, the vertex and two open edges admit a unique transversal, and the minimal set contains no other element.

Suppose finally that the set contains only open edges. The characterization of the transversals to a set of line segments [9] shows that either two, three, or four of these line segments admit at most two transversals, or that the set of common transversals to all the open line segments can be parameterized by an open set of parameters in $\mathbb{R}^2$, $\mathbb{R}$, or $\mathbb{R}/\pi\mathbb{Z}$. In the latter case, the edges admit no isolated transversal, a contradiction. Hence, the minimal set of edges consists of two, three, or four edges. (Note that two or three edges may admit an isolated transversal if that transversal contains one or two of the edges.) □

We can now prove the upper bound of Theorem 1.3.

PROPOSITION 3.3. *There are $O(n^2k^2)$ minimal sets of open edges and vertices, chosen from some polytopes, that admit an isolated transversal that is tangent to these polytopes.*

Proof. We bound the number of minimal sets depending of their type according to Lemma 3.2. First, there are $O(n^2)$ pairs of vertices, pairs of edges, and sets of one vertex and one edge. Hence, at most $O(n^2)$ such pairs admit an isolated transversal.

Consider a minimal set of one vertex and two open edges, chosen from some polytopes, that admit an isolated transversal that is tangent to these polytopes. The open edges do not contain the vertex because otherwise they admit no isolated transversal. Thus the vertex and each edge define a plane. For each of the $O(n^2)$ planes defined by a vertex and an open edge not containing it, there are $O(k)$ lines in that plane that are tangent to one of the polytopes at some point other than the vertex. Hence there are $O(n^2k)$ sets of one vertex and two edges, chosen from some polytopes, that admit an isolated transversal that is tangent to these polytopes.

It is straightforward to show that three open edges admit an isolated transversal only if the line containing one of the edges intersects the other two edges. Since any line intersects at most two open edges in any of the k polytopes, there are $O(nk^2)$ sets of three open edges that admit an isolated transversal.

Consider now the case of four edges, chosen from at most three polytopes, that admit an isolated transversal that is tangent to these polytopes. The two edges chosen from the same polytope belong to the same face, and the isolated transversal lies in the plane containing that face. Each of the other two open edges intersects that plane in one point, because otherwise the four open edges admit zero or infinitely many transversals. For each of the $O(n)$ planes containing a face of one of the polytopes, and each of the $O(n)$ edges intersecting that plane in exactly one point, there are at most $2k$ lines in that plane that contain this point and are tangent to one of the k polytopes at some other point. Hence there are $O(n^2k)$ sets of four open edges, chosen from at most three polytopes, that admit an isolated transversal that is tangent to these polytopes.

We finally bound the number of sets of four edges, no two chosen from the same polytope. By Lemma 3.1 and by summing over all n edges e of the polytopes, the number T of sets of four open edges, chosen from four polytopes, that admit an isolated transversal that is tangent to these four polytopes satisfies

$$T \leqslant n \sum_{j<l<m} C\,(n_j + n_l + n_m),$$

where C is some constant. Since each n_i, $1 \leqslant i \leqslant k$, appears $\binom{k-1}{2}$ times in the sum, it follows that

$$T \leqslant C\,n \sum_{1\leqslant i\leqslant k} n_i \binom{k-1}{2} = C\,n^2 \binom{k-1}{2},$$

and thus T is in $O(n^2k^2)$ as claimed. □

The above result implies the following upper bounds and in particular those of Theorem 1.1.

Proposition 3.4. *There are $O(n^2k^2)$ connected components of maximal free line segments tangent to at least four of the polytopes. This bound also holds for connected components of possibly occluded lines tangent to at least four of the polytopes. Furthermore, the same bound holds for isolated such segments or lines.*

Proof. We prove the proposition for possibly occluded lines tangent to at least four of the polytopes; the proof is similar for maximal free line segments. By Proposition 3.3, there are $O(n^2k^2)$ minimal sets of open edges and vertices, chosen from some polytopes, that admit an isolated transversal that is tangent to these polytopes. The bound on the number of connected components thus follows from the fact that any connected component of lines tangent to four polytopes contains an isolated line. Indeed, any nonisolated line can be moved while keeping the same set of supports until (at the limit) the line intersects a new edge or vertex. During the motion, the line remains tangent to all four polytopes since it keeps the same supports (except at the limit); if the line has more than one degree of freedom, this can be repeated until the line becomes isolated. □

We now prove the upper bound of Theorem 1.2. We start with two preliminary lemmas.

Lemma 3.5. *Four possibly intersecting convex polygons in $\mathbb{R}^2$ admit at most a constant number of connected components of line transversals.*

Proof. Consider the usual geometric transform where a line in $\mathbb{R}^2$ with equation $y = ax + b$ is mapped to the point $(-a, b)$ in the dual space (see, e.g., [24, section

8.2.1]). The transversals to a convex polygon are mapped to a region bounded from above by a convex x-monotone curve and from below by a concave x-monotone curve; such a region is called a stabbing region, and the curves are referred to as the upper and lower boundaries of the stabbing region. The transversals to four polygons are mapped to the intersection of four stabbing regions. There exists no transversal of a given slope if and only if the lower boundary of a stabbing region lies above the upper boundary of another stabbing region at that slope. Two such boundaries intersect in at most two points, and thus the transversals to four polygons form at most a constant number of connected components of transversals. $\square$

As in section 2, let $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{S}$ be four polytopes in $\mathbb{R}^3$, with p, q, r, and $s \geqslant 1$ edges, respectively, and let e be a closed edge of $\mathbf{S}$.

Lemma 3.6. *There are $O(p+q+r)$ connected components of lines intersecting e and tangent to $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{S}$.*

Proof. As in the proof of Proposition 3.4, any connected component of lines intersecting e and tangent to $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{S}$ contains an isolated line. The Main Lemma thus yields that there are $O(p+q+r)$ connected components of lines intersecting e and tangent to $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{S}$ except for the components that contain only isolated lines that lie in planes that contain e and are tangent to all four polytopes.

We show that there are at most a constant number of connected components of lines intersecting e and tangent to $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{S}$ that lie in planes that contain e and are tangent to all four polytopes. There may be infinitely many such planes that intersect $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{S}$ only on l_e, but all the lines tangent to the four polytopes in all these planes belong to the same connected component. Besides these planes there are at most two planes containing e and tangent to all four polytopes. In any such plane, the lines tangent to the four polytopes are the transversals to the four polygons that are the faces, edges, or vertices of $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{S}$ lying in the plane. Lemma 3.5 thus yields the result. $\square$

We can now prove the upper bound of Theorem 1.2.

Proposition 3.7. *Given three polytopes with n edges in total and one polytope with m edges, there are $O(mn)$ connected components of lines tangent to the four polytopes.*

Proof. Let $\mathbf{S}$ denote the polytope with m edges. First, if $\mathbf{S}$ consists of a single point, it is straightforward to show that there are $O(n)$ connected components of lines tangent to the four polytopes. Otherwise, by summing over all the edges of $\mathbf{S}$, Proposition 3.6 yields that the number of connected components of lines tangent to the four polytopes is $O(mn)$. $\square$

4. Lower bounds.

We provide in this section the lower bound examples needed for Theorems 1.1, 1.2, and 1.3. The following proposition proves the lower bound of Theorem 1.2.

Lemma 4.1. *There exist four disjoint polytopes of complexity n such that the number of common tangent lines is finite and $\Omega(n^2)$. There also exist two polytopes of complexity n and two polytopes of complexity m such that the number of common tangent lines is finite and $\Omega(mn)$.*

Proof. We consider four planar regular polygons P, Q, R, and S, each with n vertices, embedded in $\mathbb{R}^3$. P is centered at the origin and parallel to the yz-plane, Q is obtained from P by a rotation of angle $\frac{\pi}{n}$ about the x-axis, and R and S are obtained from P and Q, respectively, by a translation of length 1 in the positive x-direction (see Figure 4.1). We transform the polygons P and Q into the polytopes $\mathbf{P}$ and $\mathbf{Q}$ by adding a vertex at coordinates $(\varepsilon, 0, 0)$. Similarly, we transform the polygons R and

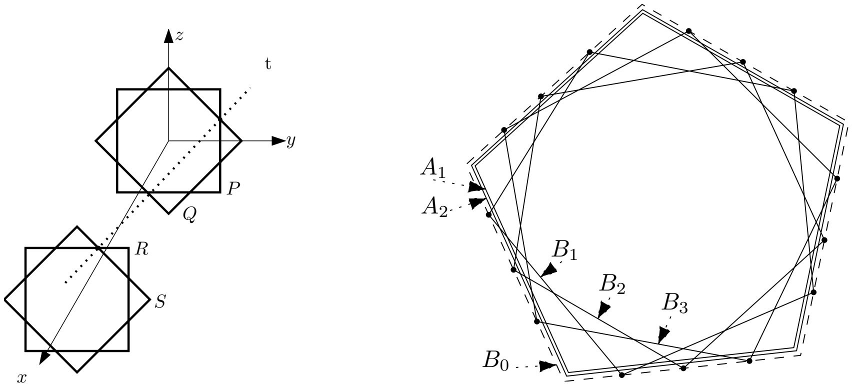


FIG. 4.1. *Lower bound examples for Lemmas* 4.1 *and* 4.2.

S into the polytopes $\mathbf{R}$ and $\mathbf{S}$ by adding a vertex at coordinates $(1+\varepsilon, 0, 0)$.

For ε sufficiently small, the lines tangent to $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{S}$ are the lines through a vertex of $P \cap Q$ and a vertex of $R \cap S$. Since $P \cap Q$ and $R \cap S$ have $2n$ vertices each, there are $4n^2$ tangent lines. Now, moving $\mathbf{P}$ and $\mathbf{S}$ by 2ε in the x direction ensures the disjointness of the polytopes while preserving the existence of the tangents if ε is small enough.

Replacing R and S in the above construction by regular polygons each with m vertices yields the $\Omega(mn)$ lower bound in the case of two polytopes of complexity n and two polytopes of complexity m. □

We now prove the lower bounds of Theorems 1.1 and 1.3. The following proposition directly yields these bounds since the number of isolated tangents to any four of the polytopes is less than or equal to the number of sets of open edges and vertices in at most four polytopes that admit an isolated transversal that is tangent to these polytopes.

LEMMA 4.2. *There exist k disjoint polytopes of total complexity n such that the number of maximal free line segments tangent to four of them is finite and $\Omega(n^2k^2)$. Moreover, these segments lie in pairwise distinct lines.*

Proof. The lower bound example is similar to the one with four polyhedra. For simplicity suppose that n and k are such that $\frac{n}{k}$ and $\frac{k}{4}$ are integers. We first take an $\frac{n}{k}$-regular polygon A_1 in the plane $x = 0$. Next we consider a copy, B_0, of A_1 scaled by a factor of $(1+\varepsilon)$, and on each edge of B_0 we place $\frac{k}{4}$ points. Polygon B_i, $1 \leqslant i \leqslant \frac{k}{4}$, is constructed by taking the ith point on each edge of B_0. If ε is small enough, the intersection points of A_1 and B_i are outside the other polygons B_j for $1 \leqslant j \leqslant \frac{k}{4}$ and $i \neq j$. Now the A_i, for $2 \leqslant i \leqslant \frac{k}{4}$, are constructed as copies of A_1 scaled by a factor $1 + \frac{i}{k}\varepsilon$ (see Figure 4.1). For the moment, all polygons lie in plane $x = 0$. We now construct four families of $\frac{k}{4}$ polygons each:

- P_i is a copy of A_i translated by $i\epsilon$ in the negative x direction.
- Q_i is a copy of B_i translated by $i\epsilon$ in the positive x direction.
- R_i is a copy of B_i translated by $1 - i\epsilon$ in the positive x direction.
- S_i is a copy of A_i translated by $1 + i\epsilon$ in the positive x direction.

Any choice of four polygons, one in each family P_i, Q_j, R_l, and S_m, reproduces the quadratic example of Lemma 4.1 with polygons of size $\frac{n}{k}$ and thus with total number of tangents greater than $\left(\frac{k}{4}\right)^4 4 \left(\frac{n}{k}\right)^2 = \frac{n^2k^2}{4}$. Furthermore, the lines tangent to P_i, Q_j, R_l, and S_m are occluded only by $P_{i'}$ and $S_{m'}$ for $i' > i$ and $m' > m$, that is, beyond the portion of the tangents containing the contact points. The k polygons can be transformed into k convex polyhedra as in Lemma 4.1. $\square$

5. Algorithm.

Using the sweep-plane algorithm outlined in section 2.1, we can compute in $O(n^2k^2 \log n)$ time all minimal sets of open edges and vertices, chosen from some of the polytopes, that admit a possibly occluded isolated transversal that is tangent to these polytopes. Now, for some of these lines, the segment joining the contact points with the polytopes is free. We can use standard, but complicated, ray-shooting data structures in order to determine which of these $O(n^2k^2)$ segments are free; this can be done in $O(\log^2 n)$ time per query using $O((nk)^{2+\epsilon})$ preprocessing time and storage [3].

We present in this section a solution that uses $O(n^2k^2 \log n)$ time and $O(nk^2)$ space. We adapt the algorithm outlined in section 2.1 to directly compute the minimal sets of edges and vertices admitting an isolated line transversal that contains a free segment tangent to their respective polytopes. Our algorithm has better time and space complexities than the previously mentioned approach and is readily implementable. Moreover, the space complexity drops to $O(nk)$ if no occlusion is taken into account. Precisely, we prove the following theorem which is more powerful, though more technical, than Theorem 1.4 and directly yields it.

THEOREM 5.1. *Given k polytopes in $\mathbb{R}^3$ with n edges in total, we can compute in $O(n^2k^2 \log n)$ time and $O(nk)$ space all the minimal sets of open edges and vertices, chosen from some of the polytopes, that admit an isolated, possibly occluded, line transversal tangent to these polytopes. We can also compute, in $O(n^2k^2 \log n)$ time and $O(nk^2)$ space, all the minimal sets of open edges and vertices that admit an isolated line transversal containing a maximal free segment that is tangent to these polytopes. Furthermore, the algorithm reports which of the transversals contain such a free line segment.*

For ease of presentation, we describe a simplified version of the algorithm in which we assume that the polytopes are in generic position; see section 5.2 for details. Using the same techniques as in section 2, it is straightforward though tedious to generalize the algorithm for arbitrary situations. We also detail the algorithm only for the case of minimal sets of four edges, with no two chosen from the same polytope; the other sets of at most four edges and vertices can be computed similarly.

5.1. Algorithm overview and data structures.

The input to our algorithm is a set of possibly intersecting polytopes structured in a standard way so that classic incidence queries can be performed in constant time (see, for instance, [7, section 9.1]).

We consider each polytope edge, e, in turn and sweep a plane around it between its two incident faces. During the sweep we create and maintain the following objects.

Combinatorial polygons. The sweep plane intersects each polytope in a (possibly empty) convex polygon whose vertices correspond to polytope edges. For each of these polygons, we maintain the set of vertices, each represented by its corresponding polytope edge, in a data structure that admits logarithmic-time vertex insertion, deletion, and look-up operations, as well as ray-shooting queries. This can be done with a balanced binary search tree (see [21, section 7.9.1]).

Combinatorial bitangents. The algorithm keeps track of the lines contained in the sweep plane and tangent to two polygons. The polytopes properly intersected by such a bitangent between its two supports are its *blockers*. A bitangent is represented by (pointers to) its two supports and a set of its blockers, ordered by polytope index, stored in a balanced binary search tree.

Polytope edges. We associate with each polytope edge a list of pointers to the combinatorial bitangents it supports in the current sweep plane.

Critical events. The sweep stops at critical events at which time combinatorial polygons and bitangents are updated. In addition to the V- and F-critical events defined in section 2.3, we introduce the following two new types of events at which the set of blockers of some combinatorial bitangents may change. A *T-critical* event occurs whenever three bitangents, supported by a **PQR**-tuple, become aligned (see Figure 5.1(b)). An *I-critical* event occurs when the sweep plane contains a point of intersection between an edge and a face of two (distinct) polytopes (see Figure 5.2).

Each event is represented by a data structure providing pointers to the primitives that define it: a vertex for a V-event, a bitangent and a face for an F-event, three bitangents for a T-event, and a face and an edge for an I-event. In addition, for a T-event, we store a bit of information specifying which of the line transversals to l_e and the three support edges defines the T-event. Note that the critical value of each critical event can be computed in constant time from the information associated with the event; it thus does not need to be explicitly stored.

Finally, critical events are sorted in the order in which they appear during the sweep and stored in an *event queue* supporting insertion and deletion in logarithmic time.

5.2. Generic position assumption. Our generic position assumption is that *the ordered set of events does not change under any arbitrarily small perturbation of the input polytopes.* This assumption corresponds to the following: (i) the events are generic, and (ii) no two events occur in the same sweep plane, except for F- and I-critical events induced by the same pair of edge and face. The genericity of the events is ensured by (but not characterized by) the following geometric conditions:

V-critical events: no vertex lies on a line containing another edge.

F-critical events: no two edges in two distinct polytopes are coplanar.

I-critical events: if an edge intersects a face of another polytope, it does so properly and not on a line containing another edge.

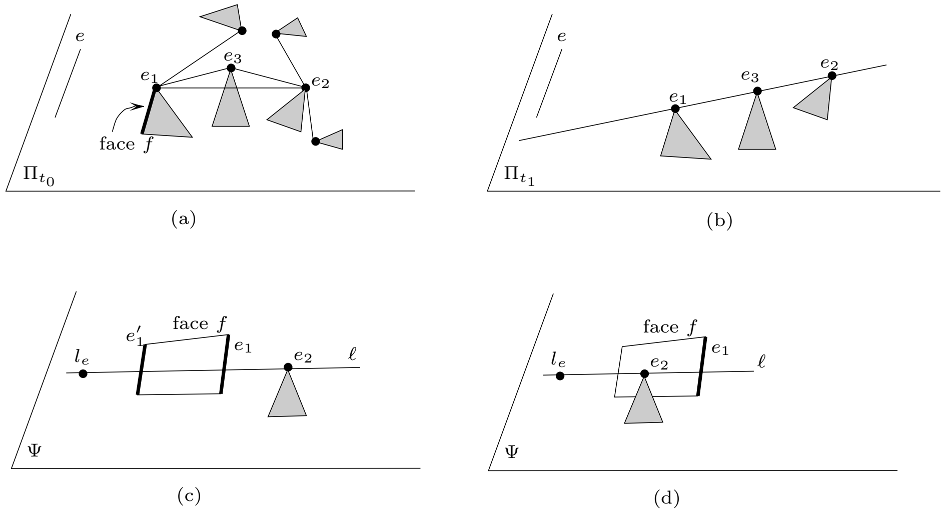
T-critical events: any four lines containing polytope edges admit zero or two transversals.

5.3. Initialization. For each new sweep, we initialize the event queue and construct the combinatorial polygons and combinatorial bitangents as follows.

Combinatorial polygons. Computing the combinatorial polygons in the initial sweep plane can easily be done in $O(n)$ time.

Combinatorial bitangents. The bitangent lines to two polygons P and Q in the initial sweep plane through a given vertex of P can be computed by a binary search on Q in $O(\log n)$ time. The blockers of a given bitangent can be found using one ray-shooting query per combinatorial polygon, for a total time of $O(k \log n)$. Altogether, the $O(nk)$ combinatorial bitangents can thus be computed in $O(nk^2 \log n)$ time.

Event queue. There are $O(n)$ V-critical events and $O(nk)$ I-critical events, since an edge intersects a polytope in at most two faces. The $O(nk)$ edge-face intersection points are computed and stored once before the beginning of the first sweep; this computation can be done by using brute force in $O(n^2)$ time and with $O(nk)$ space, since it is done once for all the sweeps. For each sweep, all the V- and I-critical events can then be inserted in $O(nk \log n)$ time. For each of the $O(nk)$ combinatorial bitangents, we also insert F- and T-critical events in $O(k \log n)$ time as explained in



(a)

(b)

(c)

(d)

FIG. 5.1. (a) *The sweep plane in which the combinatorial bitangent with support edges* e_1 *and* e_2 *is created.* (b) *The sweep plane at a T-critical event induced by the three bitangents with support edges in* e_1*,* e_2*, and* e_3*.* (c)–(d) *A line* ℓ *that defines an F-critical event.* (d) *The F-event defined by* ℓ *occurs simultaneously with an I-critical event.*

section 5.4 (Lemma 5.2). In total, initializing the event queue takes $O(nk^2 \log n)$ time per sweep.

Thus, initializing all the combinatorial polygons, bitangents, and the event queue can be done in $O(nk^2 \log n)$ time per sweep plus $O(n^2)$ time overhead for a total of $O(n^2k^2 \log n)$ time as announced in Theorem 5.1.

5.4. Updating the event queue. Every time a new combinatorial bitangent is created, we compute and insert into the queue new F- and T-events as described below. Let e_1 and e_2 denote the two support edges of a new combinatorial bitangent. Let Π_{t_0} denote the critical plane at which the new combinatorial bitangent is created.

New T-critical events. See Figure 5.1(a)–(b). Consider all the bitangents having e_1 as support edge and compute the set of support edges (distinct from e_1 and e_2) of all these bitangents. Compute the intersection of this set with the similar set for e_2; this can be done in $O(k \log k)$ time by ordering the edges by their indices. For each edge e_3 in that set, insert a T-event for each line transversal to l_e, e_1, e_2, and e_3 if the transversal is tangent to the three polytopes containing e_1, e_2, and e_3; this test can be done in constant time. Each of the at most k insertions into the event queue takes $O(\log n)$. Thus computing and inserting the new T-critical events takes $O(k \log n)$ time per new bitangent.

New F-critical events. Consider in turn each of the four faces incident to one of the two support edges. Let e_1 and f denote the considered edge and face. We compute a candidate F-event, in constant time, as follows. Compute the line ℓ (if any) that lies in the plane Ψ containing f and goes through l_e and e_2 (see Figure 5.1(c)). If ℓ is tangent to the polytope containing e_2, ℓ defines an F-event. We reject this F-event if ℓ does not intersect e_1 (in such a case, the edge e_1 does not intersect the sweep plane at the F-event, and thus the combinatorial bitangent to e_1 and e_2 would have been deleted at some V-event before the F-event). We also discard this F-event if it occurs at the critical value t_0 where the (considered) bitangent is created (that is, Π_{t_0} contains ℓ); we discard such F-events because when a bitangent is created at an

F-event, we do not reinsert the same F-event into the queue. We thus retain at most four F-events, at most one for each of the four faces incident to one of the two support edges. If no F-event is retained, the bitangent will be deleted at a V-critical event and no new F-critical event is created. If more than one F-event is retained, we need only keep the first one, since, as we shall see in section 5.5.2, the combinatorial bitangent will be deleted at the first of these events.

Again, let f denote the face incident to edge e_1 that induces that F-critical event. If the other support edge, e_2, intersects face f (see Figure 5.1(d)), then this event will be treated as an I-critical event and again we create no new F-event. Otherwise, we insert the F-event into the queue in $O(\log n)$ time. We thus get the following lemma.

Lemma 5.2. *Each time a combinatorial bitangent is created, the event queue can be updated in $O(k \log n)$ time.*

5.5. Processing events.

5.5.1. V-critical events. Let v denote the vertex that induces a V-critical event. As the sweep plane reaches v, all edges incident to v start or cease to be swept; we call the former *starting* edges and the latter *terminating* edges. Let $\mathbf{Q}$ denote the polytope to which v belongs and let Π_{t_0} be the sweep plane containing v. When processing a V-event, we perform the following operations.

Create and delete combinatorial bitangents. Suppose first that the critical plane through v properly intersects $\mathbf{Q}$. Consider in turn each combinatorial bitangent supported by a terminating edge, e_t, incident to v and let h denote the other support edge of this bitangent. We check all starting edges incident to v to find the edge e_s such that the line in $\Pi_{t_0+\epsilon}$ through e_s and h is tangent to $\mathbf{Q}$ for $\epsilon > 0$ arbitrarily small. We create a new combinatorial bitangent and delete the old one; in fact, we simply replace e_t by e_s in the combinatorial bitangent, create a pointer from edge e_s to the bitangent, and update the event queue. After handling the last bitangent supported by edge e_t, delete all the pointers from e_t to the bitangents.

The critical plane through v contains $O(k)$ bitangents through v; thus, by continuity, at most $O(k)$ combinatorial bitangents are deleted and created. Each deletion and creation takes linear time in the degree of v plus $O(k \log n)$ time for updating the event queue (Lemma 5.2). Hence, since the sum of the degrees of the vertices is $O(n)$, this step takes $O(nk^2 \log n)$ time in total for all nonextremal V-events.

Suppose now that the critical plane through v is tangent to Q and that all edges incident to v are starting. For each edge not incident to v, we can decide in constant time whether it supports a bitangent through v in the critical plane through v. If so, we check, for each edge incident to v, if the line in plane $\Pi_{t_0+\epsilon}$ that goes through these two edges is tangent to $\mathbf{Q}$ for $\epsilon > 0$ arbitrarily small. If so, we create a new combinatorial bitangent. By continuity, $O(k)$ bitangents are created in total time $O(n + kd)$, where d is the degree of v. For each of these newly created bitangents, we compute its set of blockers in (brute force) $O(n)$ time and update the event queue in $O(k \log n)$ time (Lemma 5.2). This takes $O(nk \log n)$ time per event; hence $O(nk^2 \log n)$ time per sweep since there are at most two sweep planes tangent to any polytope.

Finally, if all edges incident to v are terminating, we delete all the $O(k)$ bitangents supported by these edges; for each bitangent, deleting its blockers and the pointer from the edge not incident to v can be done in $O(k)$ time. Hence, this takes $O(k^2)$ time per critical event and $O(k^3)$ time per sweep.

Update the combinatorial polygon associated with $\mathbf{Q}$. This takes $O(\log n)$ time per polytope edge incident to v, thus $O(n \log n)$ time in total for all V-events.

Hence, processing all V-events takes $O(nk^2 \log n)$ time per sweep.

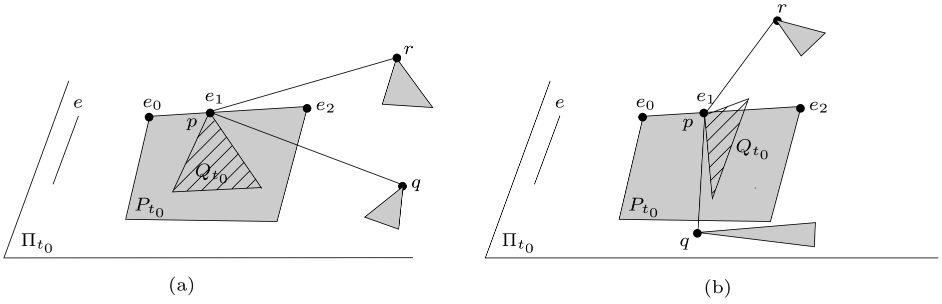


FIG. 5.2. *I-critical event.*

5.5.2. F-critical events. We process an F-critical event as follows. Let b and f denote the bitangent and face associated with the event. Let e_1 and e_2 denote the two support edges of b such that e_1 is the edge that belongs to f (see Figure 5.1(c)–(d)). By construction of F-events (see section 5.4), e_2 does not intersect face f (see Figure 5.1(c)); thus the bitangent b is deleted and a new combinatorial bitangent is created.

Bitangent b is removed from the lists of bitangents supported by e_1 and e_2 in $O(k)$ time. The support edges of the new bitangent are e_2 and the edge $e_1' \neq e_1$ of f that is intersected by the line in the plane Ψ (containing f) through l_e and e_2 (see Figure 5.1(c)). This edge e_1' is also one of the two edges adjacent to e_1 in its combinatorial polygon. Edge e_1' can thus be computed in $O(\log n)$ time. As usual, the new bitangent is added to the lists of bitangents supported by e_1' and e_2. We then compute all the blockers of this new bitangent by performing one ray-shooting query per combinatorial polygon, for a total time of $O(k \log n)$. We finally update the event queue in $O(k \log n)$ time (Lemma 5.2).

There are $O(k)$ F-events associated to each polytope face, thus $O(nk)$ F-events per sweep. Hence, the total time complexity for processing all F-events is $O(nk^2 \log n)$ per sweep.

5.5.3. I-critical events. An I-event is associated with a face f of some polytope $\mathbf{P}$ and an edge e_1 of some other polytope $\mathbf{Q}$. Let p denote the point of intersection between f and e_1. The sweep plane, Π_{t_0}, that contains p intersects the two polytopes $\mathbf{P}$ and $\mathbf{Q}$ in two polygons P_{t_0} and Q_{t_0}. See Figure 5.2. Point p lies on an edge of P_{t_0}; the two endpoints of this edge are the intersection of two edges of $\mathbf{P}$, say e_0 and e_2. These two polytope edges can be computed in $O(\log n)$ time using the combinatorial polygon associated with $\mathbf{P}$.

Create or delete combinatorial bitangents. If the two polygons P_{t_0} and Q_{t_0} are tangent at p (see Figure 5.2(a)), the two combinatorial bitangents whose pairs of support edges are (e_0, e_1) and (e_1, e_2) are either created or deleted at the I-event. If these bitangents appear in the list of bitangents having edge e_1 as support, we remove them from the list and delete them; this can be done by brute force in $O(k)$ time. Otherwise we create these two combinatorial bitangents. We compute their set of occluders in $O(k \log n)$ time by intersecting the bitangents with all the polytopes using their associated combinatorial polygons. Finally, we update the event queue in $O(k \log n)$ time.

Update sets of blockers. Now consider each of the $O(k)$ bitangents having e_1 as a support edge except for the two bitangents that might have just been created. We update its set of blockers as follows. First, note that only polytope **P** may have to be added to, or removed from, the set of blockers. Two situations occur: either the geometric bitangent segment joining the two support edges in Π_{t_0} properly intersects polygon P_{t_0} or it does not. In the first case (e.g., segment *pq* in Figure 5.2), polytope **P** was and remains a blocker of the bitangent. In the second case (e.g., segment *pr* in Figure 5.2), **P** has to be either removed from or added to the set of blockers. This can be done in $O(k \log k)$ time by searching for **P** in the set (recall that polytopes are ordered by their index in a binary search tree).

Processing an I-event thus takes $O(k \log n)$ time. Since any polytope edge intersects any other polytope in at most two points, there are $O(nk)$ I-events which can be processed in $O(nk^2 \log n)$ time in total per sweep.

5.5.4. T-critical events. Suppose that on the line transversal to e_1, e_2, e_3, and l_e (the one associated to the T-event) edges e_1, e_2, e_3 are met in that order at points p_1, p_2, p_3. Let $\mathbf{Q}_i$ be the polytope containing e_i, $1 \leqslant i \leqslant 3$.

Update sets of blockers. Update the occluder set for the bitangent with support edges e_1 and e_3 by either removing $\mathbf{Q}_2$ (if it appears in the set) or adding $\mathbf{Q}_2$ (if it does not appear in the set); this can be done in $O(\log n)$ time.

Output. First determine if the segment p_1p_3 is unoccluded by checking if the set of blockers of the bitangent with support edges e_1 and e_3 is empty or reduced to $\mathbf{Q}_2$. If so and if the segment intersects the reference edge e, then it is a free segment transversal to the four edges e, e_1, e_2, e_3. In order to report each such transversal exactly once, we report it only if the reference edge e is smaller than e_2 for some global ordering of all edges. This can be done in constant time.

There are $O(nk^2)$ T-critical events per sweep (see the proof of Proposition 3.3); thus all the T-events can be processed in $O(nk^2 \log n)$ time per sweep.

5.6. Complexity. Note first that we assume a model of computation in which bounded-degree algebraic polynomials may be evaluated in constant time. See [18] for a detailed description of the predicates concerning line transversals that are used in this algorithm.

In this model of computation, we have described a $\Theta(n^2k^2 \log n)$-time algorithm for computing all the minimal sets of edges, with no two chosen from the same polytope, that admit an isolated line transversal containing a free segment that is tangent to all these polytopes. As mentioned earlier, the sweep-plane algorithm can be easily modified to report all types of minimal support sets.

The space used by the algorithm is $\Theta(nk^2)$ in the worst case. To see this, first notice that storing the combinatorial polygons and the V-, F- and I-critical events uses $O(nk)$ space. There are also $O(nk)$ combinatorial bitangents in any sweep plane. Storing the combinatorial bitangents thus requires $\Theta(nk^2)$ space since, in the worst case, $\Theta(nk)$ of them may be intersected by $\Theta(k)$ polytopes. Furthermore, there may be $\Theta(nk^2)$ T-events in the queue since each of the $\Theta(nk)$ bitangents may share a support with $\Theta(k)$ other bitangents. This yields the bounds of Theorem 1.4 for computing minimal free segments.

Notice that, with a slight modification to the algorithm and no increase in the time complexity, we can reduce the storage requirement of the T-events to $O(nk)$. To do this we maintain the bitangents sorted by polar angle around each vertex of the combinatorial polygons, which can easily be done since the cyclic ordering changes only at T-critical events or when a bitangent is created or deleted. Since two bitangents become aligned only when they are neighbors in this cyclic ordering, we need only maintain the T-events for pairs of consecutive bitangents, and there can be only $O(nk)$ of these at any one time.

Finally, the bounds of Theorem 1.4 that concern the computation of potentially occluded isolated lines tangent to polytopes are obtained by noticing that we need not maintain the sets of blockers of the bitangents, which reduces the space requirements for the combinatorial bitangents to $O(nk)$.

6. Conclusion. We have presented a tight bound on the *number* of (connected components of) lines and maximal free line segments that are tangent to at least four among k possibly intersecting polytopes in arbitrary position. A problem that we leave open is to prove that the same bound holds for the *combinatorial complexity* of the set of all maximal free line segments among k polytopes.

We have also shown how to compute in near-optimal worst-case time all the *minimal* free line segments that are isolated transversals to their set of supports and tangent to the corresponding polytopes. We believe that our algorithm can also be made to report all connected sets of minimal free segments that are transversal to the same set of edges. A problem that we have not solved, however, is to compute in the same time and space complexities, respectively, the polytopes supporting the endpoints of the corresponding *maximal* free line segments.

REFERENCES

[1] P. K. Agarwal, *On stabbing lines for convex polyhedra in 3D*, Comput. Geom., 4 (1994), pp. 177–189.

[2] P. K. Agarwal, B. Aronov, V. Koltun, and M. Sharir, *Lines avoiding unit balls in three dimensions*, Discrete Comput. Geom., 34 (2005), pp. 231–250.

[3] P. K. Agarwal and M. Sharir, *Ray shooting amidst convex polyhedra and polyhedral terrains* in three dimensions, SIAM J. Comput., 25 (1996), pp. 100–116.

[4] M. de Berg, H. Everett, and L. J. Guibas, *The union of moving polygonal pseudodiscs—Combinatorial bounds and applications*, Comput. Geom., 11 (1998), pp. 69–81.

[5] M. de Berg, D. Halperin, M. Overmars, and M. van Kreveld, *Sparse arrangements and the number of views of polyhedral scenes*, Internat. J. Comput. Geom. Appl., 7 (1997), pp. 175–195.

[6] M. Bern, D. P. Dobkin, D. Eppstein, and R. Grossman, *Visibility with a moving point of view*, Algorithmica, 11 (1994), pp. 360–378.

[7] J. D. Boissonnat and M. Yvinec, *Algorithmic Geometry*, Cambridge University Press, Cambridge, UK, 1998.

[8] H. Brönnimann, O. Devillers, V. Dujmović, H. Everett, M. Glisse, X. Goaoc, S. Lazard, H.-S. Na, and S. Whitesides, *On the number of lines tangent to four convex polyhedra*, in Proceedings of the 14th Annual Canadian Conference on Computational Geometry, Lethbridge, Canada, 2002, pp. 113–117.

[9] H. Brönnimann, H. Everett, S. Lazard, F. Sottile, and S. Whitesides, *Transversals to line segments in three-dimensional space*, Discrete Comput. Geom., 34 (2005), pp. 381–390.

[10] F. S. Cho and D. Forsyth, *Interactive ray tracing with the visibility complex*, Computers and Graphics, 23 (1999), pp. 703–717.

[11] R. Cole and M. Sharir, *Visibility problems for polyhedral terrains*, J. Symbolic Comput., 7 (1989), pp. 11–30.

[12] O. Devillers, V. Dujmović, H. Everett, X. Goaoc, S. Lazard, H.-S. Na, and S. Petitjean, *The expected number of 3D visibility events is linear*, SIAM J. Comput., 32 (2003), pp. 1586–1620.

[13] D. P. Dobkin and D. G. Kirkpatrick, *Fast detection of polyhedral intersection*, Theoret. Comput. Sci., 27 (1983), pp. 241–253.

[14] F. Durand, *A Multidisciplinary Survey of Visibility*, ACM SIGGRAPH Course Notes: Visibility, Problems, Techniques, and Applications, 2000.

[15] F. Durand, G. Drettakis, and C. Puech, *The visibility skeleton: A powerful and efficient multi-purpose global visibility tool*, in Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, 1997, pp. 89–100.

[16] F. Durand, G. Drettakis, and C. Puech, *The* 3*D visibility complex*, ACM Trans. Graphics, 21 (2002), pp. 176–206.

[17] A. Efrat, L. J. Guibas, O. A. Hall-Holt, and L. Zhang, *On incremental rendering of silhouette maps of a polyhedral scene*, in Proceedings of the Eleventh ACM-SIAM Symposium on Discrete Algorithms, San Francisco, 2000, pp. 910–917.

[18] H. Everett, S. Lazard, W. Lenhart, J. Redburn, and L. Zhang, *Predicates for line transversals in* 3*D*, in Proceedings of the 18th Annual Canadian Conference on Computational Geometry, 2006, pp. 43–46.

[19] X. Goaoc, *Structures de visibilité globales: Taille, calcul et dégénérescences*, Ph.D. thesis, Université Nancy 2, Nancy, France, 2004.

[20] D. Halperin and M. Sharir, *New bounds for lower envelopes in three dimensions, with applications to visbility in terrains*, Discrete Comput. Geom., 12 (1994), pp. 313–326.

[21] J. O'Rourke, *Computational Geometry in C*, 2nd ed., Cambridge University Press, Cambridge, UK, 1998.

[22] M. Pellegrini, *On lines missing polyhedral sets in* 3*-space*, Discrete Comput. Geom., 12 (1994), pp. 203–221.

[23] M. Pocchiola and G. Vegter, *The visibility complex*, Internat. J. Comput. Geom. Appl., 6 (1996), pp. 279–308.

[24] M. Sharir and P. K. Agarwal, *Davenport-Schinzel Sequences and Their Geometric Applications*, Cambridge University Press, Cambridge, UK, 1995.

[25] J. Stolfi, *Oriented Projective Geometry: A Framework for Geometric Computations*, Academic Press, New York, 1991.

SIAM J. COMPUT.
Vol. 37, No. 2, pp. 552–583

ONE-WAY COMMUNICATION COMPLEXITY AND THE NEČIPORUK LOWER BOUND ON FORMULA SIZE[*]

HARTMUT KLAUCK[†]

Abstract. In this paper the Nečiporuk method for proving lower bounds on the size of Boolean formulas is reformulated in terms of one-way communication complexity. We investigate the settings of probabilistic formulas, nondeterministic formulas, and quantum formulas. In all cases we can use results about one-way communication complexity to prove lower bounds on formula size. The main results regarding formula size are as follows: We show a polynomial size gap between probabilistic/quantum and deterministic formulas, a near-quadratic gap between the sizes of nondeterministic formulas with limited access to nondeterministic bits and nondeterministic formulas with access to slightly more such bits, and a near-quadratic lower bound on quantum formula size. Furthermore we give a polynomial separation between the sizes of quantum formulas with and without multiple read random inputs. The lower bound methods for quantum and probabilistic formulas employ a variant of the Nečiporuk bound in terms of the Vapnik–Chervonenkis dimension. To establish our lower bounds we show optimal separations between one-way and two-way protocols for limited nondeterministic and quantum communication complexity, and we show that zero-error quantum one-way communication complexity asymptotically equals deterministic one-way communication complexity for total functions.

Key words. formula size, communication complexity, quantum computing, limited nondeterminism, lower bounds, computational complexity

AMS subject classifications. 68Q17, 68Q10, 81P68, 03D15

DOI. 10.1137/S009753970140004X

1. Introduction.

One of the most important goals of complexity theory is to prove lower bounds on the size of Boolean circuits computing some explicit functions. Currently only linear lower bounds for this complexity measure are known. It is well known that superlinear lower bounds are provable, however, if we restrict the circuits to fan-out 1, i.e., if we consider Boolean formulas. The best known technique for providing these is due to Nečiporuk [32]; see also the survey by Boppana and Sipser [7]. It applies to Boolean formulas with arbitrary gates of fan-in 2. For other methods applying to circuits over a less general basis of gates, see again [7]. The largest lower bounds provable with Nečiporuk's method are of the order $\Theta(n^2/\log n)$.

The complexity measure of formula size is not only interesting because formulas are restricted circuits, which are easier to handle in lower bounds, but also because the logarithm of the formula size is asymptotically equivalent to the circuit depth.

It has become customary to consider randomized algorithms as a standard model of computation. While randomization can be eliminated quite efficiently using the nonuniformity of circuits, randomized circuits are sometimes simpler to describe and

[*] Received by the editors December 20, 2001; accepted for publication (in revised form) January 8, 2007; published electronically June 5, 2007. The results in this paper appeared in *On the size of probabilistic formulae*, in Proceedings of the 8th International Symposium on Algorithms and Computation, Singapore, 1997, Lect. Notes Comput. Sci. 1350, Springer, 1997, pp. 243–252; *Lower bounds for computation with limited nondeterminism*, in Proceedings of the 13th IEEE Conference on Computational Complexity, Buffalo, NY, 1998, pp. 141–153; and *On quantum and probabilistic communication: Las Vegas and one-way protocols*, in Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, Portland, OR, 2000, pp. 644–651.

http://www.siam.org/journals/sicomp/37-2/40004.html

[†] Institut für Informatik, Goethe-Universität Frankfurt, Robert-Mayer Str. 11–15, 60054 Frankfurt, Germany (klauck@thj.informatik.uni-frankfurt.de). The work of this author was supported by DFG Project KL 1470/1.

more concise than deterministic circuits. It is natural to ask whether we can prove lower bounds for the size of randomized formulas.

More generally, we like to consider different modes of computation other than randomization. First we are interested in nondeterministic formulas. It turns out that general nondeterministic formulas are as powerful as nondeterministic circuits and thus are intractable for lower bounds with current techniques. But this construction relies heavily on a large consumption of nondeterministic bits guessed by the simulating formula; in other words, such a simulation drastically increases the length of *proofs* involved in nondeterministic computation. So we can ask whether the size of formulas with a limited number of nondeterministic guesses can be lower bounded, in the spirit of research on limited nondeterminism (for a survey of this topic see [14]).

Finally, we are interested in quantum computing. The model of quantum formulas was introduced by Yao in [41]. He gave a superlinear lower bound for quantum formulas computing the MAJORITY function. Later Roychowdhury and Vatan [37] proved that the classical Nečiporuk bound divided by $\log n$ applies to quantum formulas and showed a lower bound of the order $\Omega(n^2/\log^2 n)$ for an explicit function. They also showed that quantum formulas can actually be simulated quite efficiently by classical Boolean circuits.

The outline of this paper is the following. First we observe that the Nečiporuk method can be defined in terms of one-way communication complexity. While this observation is not relevant for deterministic computations, it becomes useful if we consider other modes of computation. First we consider probabilistic formulas. We derive a variation of the Nečiporuk bound in terms of randomized communication complexity and, using results due to Kremer, Nisan, and Ron [26], a combinatorial variant involving the Vapnik–Chervonenkis (VC) dimension. Applying this lower bound we show a near-quadratic lower bound for probabilistic formula size (Corollary 3.7).

We also exhibit a function for which probabilistic formulas are smaller by a factor of $\sqrt{n}$ than deterministic formulas and even Las Vegas (zero-error) formulas (Corollary 3.13). This is shown to be the maximal such gap provable under the condition that the lower bound for deterministic formulas is given by the Nečiporuk method. Furthermore we observe that the standard Nečiporuk bound asymptotically also works for Las Vegas formulas.

We then introduce communication complexity type Nečiporuk methods for nondeterministic formulas and for quantum formulas. To apply these generalizations we have to provide lower bounds for one-way communication complexity with limited nondeterminism and for quantum one-way communication complexity. Since the communication problems we investigate are asymmetric (i.e., Bob receives much fewer inputs than Alice), our results show optimal separations between one- and two-round communication complexity for limited nondeterministic and for quantum communication complexity. Such separations have been known previously for deterministic and probabilistic protocols; see [26, 36]. In the quantum case such a separation is given in the equivalent scenario of quantum random access codes in the work of Ambainis et al. [2]. In the case of limited nondeterminism such a separation was unknown prior to this work.

In the nondeterministic case we give a specific combinatorial argument for the communication lower bound (Theorem 5.5). In the quantum case we give a general lower bound method based on the VC dimension (Theorem 5.9), which can also be extended to the case where the players share prior entanglement, as an application of the ideas in [2]. The generalization to protocols with entanglement has also been observed by Nayak in his thesis [31]. Furthermore we show that exact and Las Vegas quantum one-way communication complexity are never much smaller than deterministic one-way communication complexity for total functions (Theorems 5.11 and 5.12), generalizing a theorem of Hromkovič and Schnitger [19].

Then we are ready to give Nečiporuk-type lower bound methods for nondeterministic formulas and quantum formulas. In the nondeterministic case we show that for an explicit function there is a threshold on the amount of nondeterminism needed for efficient formulas; i.e., a near-quadratic size gap occurs between formulas allowed to make a certain amount of nondeterministic guesses and formulas allowed a logarithmic factor more. The threshold is polynomial in the input length (Theorem 6.4).

For quantum formulas (in Corollary 6.11) we show a lower bound of $\Omega(n^2/\log n)$, improving by a logarithmic factor on the best previously known bound due to Roychowdhuri and Vatan [37]. More importantly, our bound also applies to a more general model of quantum formulas, which are, e.g., allowed to access multiple read random variables. This feature makes these generalized quantum formulas a proper generalization of both quantum formulas and probabilistic formulas. It turns out that we can give a $\Omega(\sqrt{n}/\log n)$ separation between formulas with multiple read random variables and without this option, even if the former are classical and the latter are quantum (Corollary 6.6). Thus quantum formulas as defined by Yao are not capable of efficiently simulating classical probabilistic formulas. We show that the VC-dimension variant of the Nečiporuk bound holds for generalized quantum formulas and the standard Nečiporuk bound holds for generalized quantum Las Vegas formulas (Theorem 6.10).

The organization of the paper is as follows: In section 2 we describe some preliminaries regarding the VC dimension, classical communication complexity, and Boolean circuits. In section 3 we expose the basic lower bound approach and apply the idea to probabilistic formulas. In section 4 we give more background on quantum computing and information theory. In section 5 we give the lower bounds for nondeterministic and quantum one-way communication complexity. In section 6 we derive our results for nondeterministic and quantum formulas and apply those bounds. In section 7 we give some conclusions.

2. Preliminaries.

2.1. The VC dimension.

We start with a useful combinatorial concept [39], the Vapnik–Chervonenkis dimension. This will be employed to derive lower bounds for one-way communication complexity and then to give generalizations of the Nečiporuk lower bound on formula size.

Definition 2.1. *A set S is shattered by a set of Boolean functions $\mathcal{F}$, if for all $R \subseteq S$ there is a function $f \in \mathcal{F}$, so that for all $x \in S$: $f(x) = 1 \iff x \in R$.*

The size of the largest set shattered by $\mathcal{F}$ is called the VC dimension $VC(\mathcal{F})$ of $\mathcal{F}$.

The following fact [39] will be useful.

Fact 2.2. *Let $\mathcal{F}$ be a set of Boolean functions $f : X \to \{0,1\}$. Then*

$$2^{VC(\mathcal{F})} \leq |\mathcal{F}| \leq (|X|+1)^{VC(\mathcal{F})}.$$

2.2. One-way communication complexity.

We now define the model of one-way communication complexity, first described by Yao [40]. Our discussion of this model will be informal; for a more formal treatment of the material and additional background information we refer to the excellent monograph by Kushilevitz and Nisan [27].

DEFINITION 2.3. *Let $f : X \times Y \to \{0,1\}$ be a function. Two players Alice and Bob with unrestricted computational power receive inputs $x \in X, y \in Y$ to the function.*

Alice sends a binary encoded message to Bob, who then computes the function value. The complexity of a protocol is the worst case length of the message sent (over all inputs).

The deterministic one-way communication complexity of f, denoted $D(f)$, is the complexity of an optimal deterministic protocol computing f.

In the case Bob sends one message and Alice announces the result, we use the notation $D^B(f)$.

The communication matrix of a function f is the matrix M, with $M(x,y) = f(x,y)$ for all inputs x, y.

We will consider different modes of acceptance for communication protocols. Let us begin with nondeterminism.

DEFINITION 2.4. *In a nondeterministic one-way protocol for a Boolean function $f : X \times Y \to \{0,1\}$ Alice first guesses nondeterministically a sequence of s bits. Then she sends a message to Bob, depending on the sequence and her own input. Bob computes the function value. Note that the guessed sequence is known only to Alice. In such a protocol an input is accepted, if there is at least one sequence of s bits, which leads to the output "1" when guessed by Alice. All other inputs are defined as rejected. f is computed by the protocol, if all input pairs are accepted/rejected correctly by the nondeterministic protocol.*

The complexity of a nondeterministic one-way protocol with s nondeterministic bits is the length of the longest message used.

The nondeterministic communication complexity $N(f)$ is the complexity of an optimal one-way protocol for f using arbitrarily many nondeterministic bits.

$N_s(f)$ denotes the complexity of an optimal nondeterministic protocol for f, which uses at most s private nondeterministic bits for every input.

Note that if we do not restrict the number of nondeterministic bits, then nondeterministic protocols with more than one round of communication can be simulated without loss: Alice guesses a dialogue and sends this dialogue if it is consistent with her input; Bob checks the same with his input and outputs 1 if this is implied by the dialogue.

While nondeterministic communication is a theoretically motivated model, probabilistic communication is the most powerful realistic model of communication besides quantum mechanical models.

DEFINITION 2.5. *In a probabilistic protocol with private random coins Alice and Bob each possess a source of independent random bits that can be used to obtain an arbitrary number of random bits under the uniform distribution. The players are allowed to access that source and communicate depending on their inputs and the random bits they read. We distinguish the following modes of acceptance:*

1. *In a Las Vegas protocol the players are not allowed to err. They may, however, give up without an output with some probability ϵ. The complexity of a one-way protocol is the worst case length of a message used by the protocol; the Las Vegas complexity of a function f is the complexity of an optimal Las Vegas protocol computing f and is denoted $R_{0,\epsilon}(f)$.*
2. *In a probabilistic protocol with bounded error ϵ the output has to be correct with probability at least $1 - \epsilon$. The complexity of a protocol is the worst case length of the message sent (over all inputs and the random guesses); the complexity of a function is the complexity of an optimal protocol computing*

that function and is denoted $R_\epsilon(f)$. For $\epsilon = 1/3$ the notation is abbreviated to $R(f)$.

3. *A bounded error protocol has one-sided error, if inputs with $f(x_A, x_B) = 0$ are rejected with certainty.*

We also consider probabilistic communication with public randomness. Here the players have access to a shared source of random bits without communicating. This means that both players can read the ith bit produced by the random source and thus establish a shared random bit string. Complexity in this model is denoted R^{pub}, with acceptance defined as above.

The difference between probabilistic communication complexity with public and with private random bits is actually only an additive $O(\log n)$ as shown by Newman [33] via an argument based on the nonuniformity of the model.

The following communication problems are frequently considered in the literature about communication complexity.

Definition 2.6 (disjointness problem). $DISJ_n(x_1 \ldots x_n, y_1 \ldots y_n) = 1 \iff \forall i : \neg x_i \vee \neg y_i$. *The function accepts, if the two sets described by the inputs are disjoint.*

Index function:

$IX_{2^n}(x_1 \ldots x_{2^n}, y_1 \ldots y_n) = 1 \iff x_y = 1.$

The deterministic one-way communication complexity of a function can be characterized as follows. Let $row(f)$ be the number of different rows in the communication matrix of f. Note that in the communication matrix the rows are associated to the inputs of the sender, Alice.

Fact 2.7. $D(f) = \lceil \log row(f) \rceil$.

It is relatively easy to estimate the deterministic one-way communication complexity using this fact. As an example consider the index function; note that obviously $D^B(IX_n) = \log n$. It is easy to see with Fact 2.7 that $D(IX_n) = n$, since there are 2^n different rows in the communication matrix of IX_n. Kremer, Nisan, and Ron [26] show that also $R^{pub}(IX_n) = \Omega(n)$ holds. A bound with a tight constant factor has been obtained by Ambainis et al. [2] using information theory. Similar results were also given by Katz and Trevisan [20].

A general lower bound method for probabilistic one-way communication complexity is shown in [26].

We consider the VC dimension for functions as follows.

Definition 2.8. *For a function $f : X \times Y \to \{0,1\}$ let $\mathcal{F} = \{g | \exists x \in X : \forall y \in Y : g(y) = f(x,y)\}$. Then define $VC(f) = VC(\mathcal{F})$.*

Fact 2.9. $R^{pub}(f) = \Omega(VC(f))$.

In section 5.2 we will generalize this result to quantum one-way protocols.

With the above definition $\lceil \log |\mathcal{F}| \rceil = D(f)$. Then $VC(f) \leq D(f) \leq \lceil \log(|Y| + 1) \cdot VC(f) \rceil$ due to Fact 2.2.

Las Vegas communication can be quadratically more efficient than deterministic communication in many-round protocols for total functions [27]. For one-way protocols the situation is different as shown by Hromkovič and Schnitger [19].

Fact 2.10. *For all total functions f: $R^{pub}_{0,1/2}(f) \geq D(f)/2$.*

We will also generalize this result to quantum communication in section 5.2.

2.3. Circuits and formulas. We now define the models of Boolean circuits and formulas. Note that we do not consider questions of uniformity of families of such circuits. For the definition of a Boolean circuit we refer to [7]. We consider circuits with fan-in 2. While it is well known that almost all $f : \{0,1\}^n \to \{0,1\}$ need circuit size $\Theta(2^n/n)$ (see, e.g., [7]), superlinear lower bounds for explicit functions are known only for restricted models of circuits.

DEFINITION 2.11. *A (deterministic) Boolean formula is a Boolean circuit with fan-in* 2 *and fan-out* 1. *The Boolean inputs may be read arbitrarily often, the gates are arbitrary, and constants* $0, 1$ *may be read.*

The size (or length) of a deterministic Boolean formula is the number of its nonconstant leaves.

It is possible to show that for Boolean functions the logarithm of the formula size is linearly related to the optimal circuit depth (see [7]).

Probabilistic formulas have been considered in [38, 6, 12] with the purpose of constructing efficient (deterministic) monotone formulas for the majority function in a probabilistic manner.

The standard model of a probabilistic formula is a probability distribution on deterministic formulas. Since such a distribution can give some positive probability to all formulas of the given size, this is not a compact representation of a Boolean function. Hence we consider the following model of probabilistic formulas: "Fair" probabilistic formulas are formulas that read input variables plus additional random variables. The model mentioned before will be called "strong" probabilistic formulas.

DEFINITION 2.12. *A fair probabilistic formula is a Boolean formula, which works on input variables and additional random variables* $r_1, \ldots, r_m$; *a strong probabilistic formula is a probability distribution* F *on deterministic Boolean formulas. Fair (resp., strong) probabilistic formulas* F *compute a Boolean function* f *with bounded error, if*

$$\Pr[F(x) \neq f(x)] \leq 1/3.$$

Fair (resp., strong) probabilistic formulas F *are one-sided error formulas for* f *(i.e., have one-sided error), if*

$$\Pr[F(x) = 0 | f(x) = 1] \leq 1/2 \quad \textit{and} \quad \Pr[F(x) = 1 | f(x) = 0] = 0.$$

A Las Vegas formula consists of two Boolean formulas. One formula computes the output; the other (verifying) formula indicates whether the computation of the first can be trusted or not. Both work on the same inputs. There are four different outputs, of which two are interpreted as "?" (the verifying formula rejects), and the other as 0 *(resp.,* 1*). A Las Vegas formula* F *computes* f, *if the outputs* 0 *and* 1 *are always correct, and*

$$\Pr[F(x) = ?] \leq 1/2.$$

The size of a fair probabilistic formula is the number of its nonconstant leaves; the size of a strong probabilistic formula is the expected size of a deterministic formula according to F.

It is easy to see that one can decrease the error probability to arbitrarily small constants, while increasing the size by a constant factor; therefore, we will sometimes allow different error probabilities.

A strong probabilistic formula F can be transformed into a deterministic formula. For one-sided error formulas this increases the size by a factor of $O(n)$: Choose $O(n)$ formulas randomly according to F and connect them by an OR gate. An application of the Chernov inequality proves that the error probability is so small that no errors are possible anymore. Strong formulas with bounded (two-sided) error are derandomized by picking $O(n)$ formulas and connecting them by an approximative majority function. That function outputs 1 on n Boolean variables if at least $2n/3$ have the value 1 and outputs 0 if at most $n/3$ variables have the value 1. An approximative majority function can be computed by a deterministic formula of size $O(n^2)$; see [38, 6]. Thus the size increases by a factor of $O(n^2)$.

Let us remark that strong probabilistic formulas may have sublinear length; this is impossible for fair probabilistic formulas depending on all inputs. As an example, the approximative majority function may be computed by a strong probabilistic formula through picking a random input and outputting its value.

We will later also consider nondeterministic formulas.

DEFINITION 2.13. *A nondeterministic formula with s nondeterministic bits is a formula with additional input variables $a_1, \ldots, a_s$. The formula accepts an input x, if there is a setting of the variables a, so that (a, x) is accepted.*

3. The general lower bound method and probabilistic formulas. There are some well known results giving lower bounds for the length of Boolean formulas. The method of Nečiporuk [32, 7] remains the one giving the largest lower bounds among those methods working for formulas in which all fan-in 2 functions are allowed as gates. For other methods see [7] and [3]; a characterization for formula size with gates AND, OR, NOT using the communication complexity of a certain game is also known (see [27]). For such formulas the largest known lower bound is a near-cubic bound due to Håstad [15].

Let us first give the standard definition of the Nečiporuk bound.

Let f be a function on the n variables in $X = \{x_1, \ldots, x_n\}$. For a subset $S \subseteq X$ let a subfunction on S be a function induced by f by assigning Boolean values to the variables in $X - S$. The set of all subfunctions on S is called the set of S subfunctions of f.

FACT 3.1 (Nečiporuk). *Let f be a Boolean function on n variables. Let $S_1, \ldots, S_k$ be a partition of the variables and s_i the number of S_i subfunctions on f. Then every deterministic Boolean formulas for f has size at least*

$$(1/4) \sum_{i=1}^{k} \log s_i.$$

It is easy to see that the Nečiporuk function $(1/4) \sum_{i=1}^{k} \log s_i$ is never larger than $n^2/\log n$.

DEFINITION 3.2. *The function "indirect storage access" (ISA) is defined as follows: There are three blocks of inputs U, X, Y, with $|U| = \log n - \log\log n$, $|X| = |Y| = n$. U addresses a block of length* $\log n$ *in X, which addresses a bit in Y. This bit is the output; thus $ISA(U, X, Y) = Y_{X_U}$.*

The following is proved, e.g., in [7, 42].

FACT 3.3. *Every deterministic formula for ISA has size $\Omega(n^2/\log n)$.*

There is a deterministic formula for ISA with size $O(n^2/\log n)$.

1We are now going to generalize the Nečiporuk method to probabilistic formulas and later to nondeterministic and quantum formulas. We will use a simple connection to one-way communication complexity and use the guidance obtained by this connection to give lower bounds from lower bounds in communication complexity. In the case of probabilistic formulas we will employ the VC dimension to give lower bounds. Informally speaking we will replace the log of the size of the set of subfunctions by the VC dimension of that set and get a lower bound for probabilistic formulas.

Our lower bounds are valid in the model of strong probabilistic formulas. Corollary 3.7 shows that even strong probabilistic formulas with a two-sided error do not help to decrease the size of formulas for ISA. All upper bounds will be given for fair formulas.

We are going to show that the (standard) Nečiporuk is at most a factor of $O(\sqrt{n})$ larger than the probabilistic formula size for total functions. Thus the maximal gap we can show using the best known general lower bound method is limited.

On the other hand we describe a Boolean function, for which fair probabilistic formulas with a one-sided error are a factor $\Theta(\sqrt{n})$ smaller than Las Vegas formulas, as well as a similar gap between one-sided error formulas and two-sided error formulas. The lower bound on Las Vegas formulas uses the new observation that the standard Nečiporuk bound asymptotically also works for Las Vegas formulas.

3.1. Lower bounds for probabilistic formulas. We now derive a Nečiporuk-type bound with one-way communication.

DEFINITION 3.4. *Let f be a Boolean function of n Boolean inputs, and let $y_1 \dots y_k$ be a partition of the input variables.*

We consider k communication problems for $i = 1, \dots, k$. Player Bob receives all inputs in y_i; player Alice receives all other inputs. The deterministic one-way communication complexity of f under this partition of inputs is called $D(f_i)$. The public coin bounded error one-way communication complexity of f under this partition of inputs is called $R^{pub}(f_i)$.

The probabilistic Nečiporuk function is $(1/4)\sum_i R^{pub}(f_i)$.

It is easy to see that $(1/4)\sum_i D(f_i)$ coincides with the standard Nečiporuk function and is therefore a lower bound for deterministic formula size due to Fact 3.1.

THEOREM 3.5. *The probabilistic Nečiporuk function is a lower bound for the size of strong probabilistic formulas with a bounded error.*

Proof. We will show for every partition $y_1, \dots, y_k$ of the inputs how a strong probabilistic formula F can be simulated in the k communication games. Let F_i be the distribution over deterministic formulas on variables in y_i induced by picking a deterministic formula as in F and restricting to the subformula with all leaves labeled by variables in y_i and containing all paths from these to the root. We want to simulate the formula in game i so that the probabilistic one-way communication is bounded by the expected number of leaves in F_i.

We are given a probabilistic formula F. The players now pick a deterministic formula F' induced by F with their public random bits; player Alice knows all of the inputs except those in y_i. This also fixes a subformula F'_i drawn from F_i. Actually the players have access only to an arbitrarily large public random string, so the distributions F_i may be approximated only within arbitrary precision. This alters success probabilities by arbitrarily small values. We disregard these small changes in probability.

Let V_i contain the vertices in F'_i, which have two predecessors in F'_i, and let P_i contain all paths, which start in V_i or at a leaf, and which end in V_i or at the root, but contain no further vertices from V_i. It suffices, if Alice sends two bits for each such path, which shows whether the last gate of the path computes $0, 1, g$, or $\neg g$, for the function g computed by the first gate of the path. Then Bob can evaluate the formula alone.

There are at most $2|V_i| + 1$ paths as described, since the fan-in of the formula is 2. Thus the overall communication is $4|V_i| + 2$. The set of leaves L_i with variables from y_i has $|V_i| + 1$ elements, and thus

$$R^{pub}(f_i) \leq 4|V_i| + 2 < 4|L_i|,$$

and $1/4\sum_i R^{pub}(f_i)$ is a lower bound for the length $E[\sum_i |L_i|] = \sum_i E[|L_i|]$ of the probabilistic formula. □

Let $VC(f_i)$ denote the VC dimension of the communication problem f_i. We call $\sum_i VC(f_i)$ the VC–Nečiporuk function.

COROLLARY 3.6. *The VC–Nečiporuk function is an asymptotical lower bound for the length of strong probabilistic formulas with a bounded error.*

The standard Nečiporuk function is an asymptotical lower bound for the length of strong Las Vegas formulas for total functions.

Proof. Using Fact 2.9 the VC dimension is an asymptotical lower bound for the probabilistic public coin bounded error one-way communication complexity.

As in the proof of Theorem 3.5 we may simulate a Las Vegas formula by Las Vegas public coin one-way protocols. Using Fact 2.10 public coin Las Vegas one-way protocols for total functions can be only a constant factor more efficient than optimal deterministic one-way protocols. □

According to Fact 3.3 the deterministic formula length of the ISA function from definition 3.2 is $\Theta(n^2/\log n)$. We now employ our method to show a lower bound of the same order for strong bounded error probabilistic formulas. Thus ISA is an explicit function for which strong probabilism does not allow us to decrease the formula size significantly.

COROLLARY 3.7. *Every strong probabilistic formula for the ISA function (with a bounded error) has length $\Omega(n^2/\log n)$.*

Proof. ISA has inputs Y, X, U and computes Y_{X_U}. First we define a partition. We partition the inputs in X into $n/\log n$ blocks containing $\log n$ bits each; all other inputs are in one additional block. In a communication game Alice thus receives all inputs but those in one block of X. Let S denote the set of possible values of the variables in that block. This set is shattered: Let $R \subseteq S$ and $R = \{r_1, \dots, r_m\}$. Then set the pointer U to the block of inputs belonging to Bob, and set $Y_i = 1 \iff i \in R$.

Thus the VC dimension of f_i is at least $|S| = n$. Since there are $n/\log n$ communication games, the result follows. □

The next result would be trivial for deterministic or for fair probabilistic formulas, but strong probabilistic formulas can compute functions depending on all inputs in sublinear size. Consider, e.g., the approximate majority function. This partial function can be computed by a strong probabilistic formula of length 1 by picking a random input variable. For total functions on the other hand we have the following.

COROLLARY 3.8. *Every strong probabilistic formula which computes a total function depending on n variables has length $\Omega(n)$.*

Proof. We partition the inputs into n blocks containing one variable each. In a communication game Alice thus receives $n-1$ variables, and Bob receives 1 variable. Since the function depends on both Alice's and Bob's inputs, the deterministic communication complexity is at least 1. If the probabilistic one-way communication were 0, the error would be 1/2, and thus the protocol would not compute correctly. □

Fact 2.2 shows that for a function $f : X \times Y \to \{0,1\}$ it is true that $D(f) \leq \lceil VC(f) \cdot \log(|Y|+1) \rceil$. This leads to the following.

THEOREM 3.9. *For all total functions $f : \{0,1\}^n \to \{0,1\}$ having a strong probabilistic formula of length s and for all partitions of the inputs of f:*

$$\frac{\sum D(f_i)}{s} = O(\sqrt{n}).$$

Proof. Obviously $D(f_i) \leq n$ for all i. Since a partition of the inputs can contain at most $\sqrt{n}$ blocks with more than $\sqrt{n}$ variables, these contribute at most $n\sqrt{n}$ to the Nečiporuk function $\sum D(f_i)$. All smaller blocks satisfy $D(f_i) \leq \lceil \sqrt{n} \cdot VC(f_i) \rceil$.

Thus overall $\sum D(f_i) \leq O(\sqrt{n}(n + \sum VC(f_i))) = O(\sqrt{n}s)$, with Corollary 3.8 and Theorem 3.5. □

If a total function has an efficient (say, linear length) probabilistic formula, then the Nečiporuk method does not give near-quadratic lower bounds for the deterministic formula size.

3.2. A function for which probabilism helps. We now describe a function for which one-sided error probabilism helps as much as we can possibly show under the constraint that the lower bound for deterministic formulas is given using the Nečiporuk method. We find such a complexity gap even between strong Las Vegas formulas and fair one-sided error formulas.

DEFINITION 3.10. *The matrix product function MP receives two $n \times n$-matrices $T^{(1)}, T^{(2)}$ over $\mathbb{Z}_2$ as input and accepts if and only if their product is not the all zero matrix.*

THEOREM 3.11. *The MP function can be computed by a fair one-sided error formula of length $O(n^2)$.*

Proof. We use a fingerprinting technique similar to the one used in matrix product verification [30] but adapted to be computable by a formula. First we construct a vector as a fingerprint for each matrix using some random input variables. Then we multiply the fingerprints and obtain a bit. This bit is always zero if the matrix product is zero; otherwise, it is 1 with probability 1/4. Thus we obtain a one-sided error formula.

Let $r^{(1)}, r^{(2)}$ be random strings of n bits each. The fingerprints are defined as

$$F^{(1)}[k] = \bigoplus_{i=1}^{n} r^{(1)}[i]T^{(1)}[i,k] \text{ and } F^{(2)}[k] = \bigoplus_{j=1}^{n} T^{(2)}[k,j]r^{(2)}[j].$$

Then let

$$b = \bigoplus_{k=1}^{n} F^{(1)}[k] \wedge F^{(2)}[k].$$

Obviously b can be computed by a formula of linear length.

Assume $T^{(1)}T^{(2)} = 0$. Then $b = r^{(1)}T^{(1)}T^{(2)}r^{(2)} = 0$ for all $r^{(1)}$ and $r^{(2)}$.

If on the other hand $T^{(1)}T^{(2)} \neq 0$, then i, j exist such that $\bigoplus_k T^{(1)}[i,k]T^{(2)}[k,j] = 1$. Fix all random bits except $r^{(1)}[i]$ and $r^{(2)}[j]$ arbitrarily. Note that

$$b = \bigoplus_{i,j=1}^{n} \left(r^{(1)}[i]r^{(2)}[j] \cdot \bigoplus_{k=1}^{n} T^{(1)}[i,k]T^{(2)}[k,j] \right).$$

Regardless of how the values of sums for other i, j look, one of the values of $r^{(1)}[i]$ and $r^{(2)}[j]$ yields the result $b = 1$; this happens with probability 1/4. □

THEOREM 3.12. *For the MP function a lower bound of $\Omega(n^3)$ holds for the length of strong Las Vegas formulas.*

Proof. We use the Nečiporuk method. First the partition of the inputs has to be defined. There are n blocks b_j with the bits $T^{(2)}(i,j)$ for $i = 1, \ldots, n$ plus one block for the remaining inputs. Then Alice receives all inputs except n bits in column j of the second matrix, i.e., $T^{(2)}(\cdot, j)$, which go to Bob. We show that MP has now one-way communication complexity $\Omega(n^2)$. The Nečiporuk method then gives us a lower bound of $\Omega(n^3)$ for the length of deterministic and strong Las Vegas formulas. Without loss of generality (w.l.o.g.) assume Bob has the bits $T^{(2)}(i,1)$.

We construct a set of assignments to the input variables of Alice. Let U be a subspace of $\mathbb{Z}_2^n$ and T_U be a matrix, with $T_U x = 0 \iff x \in U$. For every U we choose T_U as $T^{(1)}$ and $T^{(2)}(i,j) = 0$ for all i and for $j \geq 2$. If there are $2^{\Omega(n^2)}$ pairwise different subspaces, then we get that many different inputs. But these inputs correspond to different rows in the communication matrix, since all $T^{(1)}$ have different kernels. Thus with Corollary 3.6 the Las Vegas one-way communication is $\Omega(n^2)$.

To see that there are $2^{\Omega(n^2)}$ pairwise different subspaces of $\mathbb{Z}_2^n$ we count the subspaces with dimension at most $n/2$. There are 2^n vectors. There are $\binom{2^n}{n/2}$ possibilities to choose a set of $n/2$ pairwise different vectors. Each such set generates a subspace of dimension at most $n/2$. Each such subspace is generated by at most $\binom{2^{n/2}}{n/2}$ sets of $n/2$ pairwise different vectors from the subspace. Hence this number is an upper bound on the number of times a subspace is counted, and there are at least

$$\frac{\binom{2^n}{n/2}}{\binom{2^{n/2}}{n/2}} \geq 2^{\Omega(n^2)}$$

pairwise different subspaces of $\mathbb{Z}_2^n$. □

COROLLARY 3.13. *There is a function that can be computed by a fair one-sided error formula of length $O(N)$, while every strong Las Vegas formula needs length $\Omega(N^{3/2})$ for this task; i.e., there is a size gap of $\Omega(N^{1/2})$ between Las Vegas and one-sided error formulas.*

There is also a size gap of $\Omega(N^{1/2})$ between one-sided error formulas and (two-sided) bounded error probabilistic formulas.

Proof. The first statement is proved in the previous theorems. For the second statement we consider the following function with four matrices as input. The function is the parity of the MP function on the first two matrices and the complement of MP on the other two matrices.

A fair probabilistic formula can compute the function obviously with length $O(n^2)$ following the construction in Theorem 3.11. Assume we have a one-sided error formula, then fix the first two input matrices once in a way so that their product is the 0 matrix and then so that their product is something else. In this way one gets one-sided error formulas for both MP and its complement. Then one can use both formulas on the same input and combine their results to get a Las Vegas formula, which leads to the desired lower bound with Theorem 3.12.

For the construction of a Las Vegas formula let F be the one-sided error formula for MP and G be the one-sided error formula for $\neg MP$. Then F and $\neg G$ are formulas for MP, so that F never erroneously accepts and is correct with probability 1/2, and $\neg G$ never erroneously rejects and is correct with probability 1/2. Assuming the function value is 0, then F rejects. With probability 1/2, $\neg G$ also rejects; otherwise, we may give up. Assuming the function value is 1, then $\neg G$ accepts. With probability 1/2, F also accepts; otherwise, we may give up. The other way around, if both formulas accept or both reject we can safely use this result, and this result comes up with probability 1/2; the only other possible result is that F rejects and $\neg G$ accepts, in which case we have to give up. □

The formula described in the proof of Theorem 3.11 has the interesting property that each input is read exactly once, while the random inputs are read often. MP cannot be computed by a deterministic formula reading the inputs only once, since this contradicts the size bound of Theorem 3.12. Later we will show that MP cannot be computed substantially more efficiently by a fair probabilistic formula reading its random inputs only once than by deterministic formulas. This follows from a lower bound for the size of such formulas given by the Nečiporuk function divided by $\log n$ (Corollary 6.7). Hence for the MP function read-once random inputs are of little use.

4. Background on quantum computing and information.

In this section we define more technical notions and describe results we will need. We start with information theory, then define the model of quantum formulas, and give results from quantum information theory. We also discuss programmable quantum gates. These results are used in the following section to give lower bounds for one-way communication complexity. Then we proceed to apply these to derive more formula size bounds.

4.1. Information theory.

We now define a few notions from classical information theory; see, e.g., [10].

DEFINITION 4.1. *Let X be a random variable with values in $S = \{x_1, \dots, x_n\}$.*

The entropy of X is $H(X) = -\sum_{x \in S} \Pr(X = x) \log \Pr(X = x)$.

The entropy of X given an event E is $H(X|E) = -\sum_{x \in S} \Pr(X = x|E) \log \Pr(X = x|E)$.

The conditional entropy of X given a random variable Y is $H(X|Y) = \sum_y \Pr(Y = y) H(X|Y = y)$, where the sum is over the values of Y. Note that $H(X|Y) = H(XY) - H(Y)$.

The information between X and Y is $I(X : Y) = H(X) - H(X|Y)$.

The conditional information between X and Y, given Z, is $I(X : Y|Z) = H(XZ) + H(YZ) - H(Z) - H(XYZ)$.

For $\alpha \in [0, 1]$ we define $H(\alpha) = -\alpha \log \alpha - (1 - \alpha) \log(1 - \alpha)$.

All of the above definitions use the convention $0 \log 0 = 0$.

The following result is a simplified version of Fano's inequality; see [10].

FACT 4.2. *If X, Y are Boolean random variables with $\Pr(X \neq Y) \leq \epsilon$, then $I(X : Y) \geq H(X) - H(\epsilon)$.*

Proof. Let $Z = 1 \iff X = Y$ and $Z = 0 \iff X \neq Y$. Then $H(X|Y) = H(XY) - H(Y) = H(ZY) - H(Y) \leq H(Z) \leq H(\epsilon)$. □

The next lemma is similar in the sense of a "Las Vegas variant."

LEMMA 4.3. *Let X be a random variable with a finite range of values S, and let Y be a random variable with range $S \cup \{x_?\}$, so that $\Pr(Y = x|X = x) \geq 1 - \epsilon$ for all $x \in S$, $\Pr(Y = x|X \neq x) = 0$ for all $x \neq x_?$, and $\Pr(Y = x_?|X = x) \leq \epsilon$ for all $x \in S$. Then $I(X : Y) \geq (1 - \epsilon) H(X)$.*

Proof. $I(X : Y) = H(X) - H(X|Y)$. Let $\delta = \Pr(Y = x_?) \leq \epsilon$, $\epsilon_x = \Pr(Y = x_?|X = x) \leq \epsilon$, $p_x = \Pr(X = x)$.

$$\begin{aligned} H(X|Y) &\leq (1 - \delta) H(X|Y \neq x_?) + \delta H(X|Y = x_?) \\ &= \delta H(X|Y = x_?) \\ &= -\delta \sum_x \Pr(X = x|Y = x_?) \log(\Pr(X = x|Y = x_?)) \\ &= -\delta \sum_x (\epsilon_x p_x / \delta) \log(\epsilon_x p_x / \delta) \\ &\leq -\epsilon \sum_x p_x \log p_x + \delta \sum_x (\epsilon_x p_x / \delta) \log(\delta / \epsilon_x) \\ &\leq \epsilon H(X) + \delta \log \sum_x p_x \text{ with Jensen's inequality} \\ &\leq \epsilon H(X). \qquad \square \end{aligned}$$

4.2. Quantum computation. We refer to [35] for a thorough introduction into the field. Let us briefly mention that pure quantum states are unit vectors in a Hilbert space written $|\psi\rangle$, inner products are denoted $\langle\psi|\phi\rangle$, and the standard norm is $\| \, |\psi\rangle \, \| = \sqrt{\langle\psi|\psi\rangle}$. Outer products $|\psi\rangle\langle\phi|$ are matrix valued.

In the space $\mathbb{C}^4$ we will consider not only the standard basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ but also the Bell basis consisting of

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), \quad |\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle),$$

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle), \quad |\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle).$$

The dynamics of a discrete time quantum system is described by unitary operations. We give some examples of such operations. A very useful operation is the Hadamard transform:

$$H_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Then $H_n = \underbrace{H_2 \otimes \cdots \otimes H_2}_{n}$ is the n-wise tensor product of H_2.

The CNOT operation is defined by $CNOT : |x, y\rangle \to |x, x \oplus y\rangle$ on Boolean values x, y.

Furthermore measurements are fundamental operations. Measuring as well as tracing out subsystems leads to probabilistic mixtures of pure states.

Definition 4.4. *An ensemble of pure states is a set $\{(p_i, |\phi_i\rangle) | 1 \le i \le k\}$. Here the p_i are the probabilities of the pure states $|\phi_i\rangle$. Such an ensemble is called a mixed state.*

The density matrix of a pure state $|\phi\rangle$ is the matrix $|\phi\rangle\langle\phi|$; the density matrix of a mixed state $\{(p_i, |\phi_i\rangle) | 1 \le i \le k\}$ is

$$\sum_{i=1}^{k} p_i |\phi_i\rangle\langle\phi_i|.$$

A density matrix is always Hermitian, positive semidefinite, and has trace 1. Thus a density matrix has nonnegative eigenvalues that sum to 1. The results of all measurements of a mixed state are determined by the density matrix.

A pure state in a Hilbert space $H = H_A \otimes H_B$ cannot in general be expressed as a tensor product of pure states in the subsystems.

Definition 4.5. *A mixed state $\{(p_i, |\phi_i\rangle) | 1 \le i \le k\}$ in a Hilbert space $H_1 \otimes H_2$ is called separable if it has the same density matrix as a mixed state $\{(q_i, |\psi_i^1\rangle \otimes |\psi_i^2\rangle) | i = 1, \ldots, k'\}$ for pure states $|\psi_i^1\rangle$ from H_1 and $|\psi_i^2\rangle$ from H_2 with $\sum_i q_i = 1$ and $q_i \ge 0$. Otherwise, the state is called entangled.*

Consider, e.g., the state $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ in $\mathbb{C}^2 \otimes \mathbb{C}^2$. The state is entangled and is usually called an EPR pair. This name refers to Einstein, Podolsky, and Rosen, who first considered such states [13].

Linear transformations on density matrices are called superoperators. Not all superoperators are physically allowed.

Definition 4.6. *A superoperator T is positive if it sends positive semidefinite Hermitian matrices to positive semidefinite Hermitian matrices. A superoperator is trace preserving if it maps matrices with trace 1 to matrices with trace 1.*

A superoperator T is completely positive if every superoperator $T \otimes I_F$ is positive, where I_F is the identity superoperator on a finite dimensional extensional F of the underlying Hilbert space.

A superoperator is physically allowed iff it is completely positive and trace preserving.

The following theorem characterizes physically allowed superoperators in terms of unitary operation, adding qubits, and tracing out [35].

FACT 4.7. *The following statements are equivalent:*

1. *A superoperator T sending density matrices over a Hilbert space H_1 to density matrices over a Hilbert space H_2 is physically allowed.*
2. *There is a Hilbert space H_3 with $\dim(H_3) \leq \dim(H_1)$ and a unitary map U, so that for all density matrices ρ over H_1:*

$$T\rho = trace_{H_1 \otimes H_3}[U(\rho \otimes |0_{H_3 \otimes H_2}\rangle\langle 0_{H_3 \otimes H_2}|)U^\dagger].$$

4.3. Quantum information theory. In this section we describe notions and results from quantum information theory.

DEFINITION 4.8. *The von Neumann entropy of a density matrix ρ_X is $S(X) = S(\rho_X) = -trace(\rho_X \log \rho_X)$.*

The conditional von Neumann entropy $S(X|Y)$ of a bipartite system with density matrix ρ_{XY} is defined as $S(XY) - S(Y)$, where the state ρ_Y of the Y system is the result of a partial trace over X.

The von Neumann information between two parts of a bipartite system in a state ρ_{XY} is $S(X:Y) = S(X) + S(Y) - S(XY)$ (ρ_X and ρ_Y are the results of partial traces).

The conditional von Neumann information of a system in state ρ_{XYZ} is $S(X : Y|Z) = S(XZ) + S(YZ) - S(Z) - S(XYZ)$.

Let $\mathcal{E} = \{(p_i, \rho_i)|i = 1, \dots, k\}$ be an ensemble of density matrices. The Holevo information of the ensemble is $\chi(\mathcal{E}) = S(\sum_i p_i\rho_i) - \sum_i p_i S(\rho_i)$.

The von Neumann entropy of a density matrix depends on the eigenvalues only, so it is invariant under unitary transformations. If the underlying Hilbert space has dimension d, then the von Neumann entropy of a density matrix is bounded by $\log d$. A fundamental result is the so-called Holevo bound [16], which states an upper bound on the amount of classical information in a quantum state.

FACT 4.9. *Let X be a classical random variable with $\Pr(X = x) = p_x$. Assume for each x that a quantum state with density matrix ρ_x is prepared; i.e., there is an ensemble $\mathcal{E} = \{(p_x, \rho_x)|x = 0, \dots, k\}$. Let $\rho_{XZ} = \sum_{x=0}^{k} p_x|x\rangle\langle x| \otimes \rho_x$. Let Y be a classical random variable which indicates the result of a measurement on the quantum state with density matrix $\rho_Z = \sum_x p_x\rho_x$. Then*

$$I(X : Y) \leq \chi(\mathcal{E}) = S(X : Z).$$

We will also need the following lemma.

LEMMA 4.10. *Let $\mathcal{E} = \{(p_x, \sigma_x)|x = 0, \dots, k\}$ be an ensemble of density matrices, and let $\sigma = \sum_x p_x\sigma_x$ be the density matrix of the mixed state of the ensemble. Assume that there is an observable with possible measurement results x and "?", so that for all x measuring the observable on σ_x yields x with probability at least $1 - \epsilon$, the result "?" with probability at most ϵ, and a result $x' \neq x$ with probability 0; then*

$$S(\sigma) \geq \sum_x p_x S(\sigma_x) + (1 - \epsilon)H(X), \quad \textit{i.e.,}\ \chi(\mathcal{E}) \geq (1 - \epsilon)H(X).$$

Proof. The proof proceeds similar to the information theoretic arguments in [2]. States x of a classical random variable X are coded as quantum states σ_x, where x and σ_x have probability p_x. The density matrix of the overall mixed state is σ and has von Neumann entropy $S(\sigma)$. σ corresponds to the "code" of a random x.

According to Holevo's theorem (Fact 4.9) the information on X one can access by measuring σ with result Y is bounded by $I(X:Y) \leq S(\sigma) - \sum_x p_x S(\sigma_x)$. But there is such a measurement as assumed in the lemma, and with Lemma 4.3 $I(X:Y) \geq (1-\epsilon)H(X)$. Thus the lemma follows. □

Not all of the relations that are valid in classical information theory hold in quantum information theory. The following fact states a notable exception, the so-called Araki–Lieb inequality and one of its consequences; see [35].

FACT 4.11. $S(XY) \geq |S(X) - S(Y)|$.

$S(X:Y|Z) \leq 2S(X)$.

The reason for this behavior is entanglement.

LEMMA 4.12. *If σ_{XY} is separable, then $S(XY) \geq S(X)$ and $S(X:Y) \leq S(X)$.*

4.4. The quantum communication model. Now we define quantum one-way protocols.

DEFINITION 4.13. *In a two-player quantum one-way protocol players Alice and Bob each possess a private set of qubits. Some of the qubits are initialized to the Boolean inputs of the players; all other qubits are in some fixed basis state $|0\rangle$.*

Alice then performs some quantum operation on her qubits and sends a set of these qubits to Bob. The latter action changes the possession of qubits rather than the global state. We can assume that Alice sends the same number of qubits for all inputs. After Bob has received the qubits he can perform any quantum operation on the qubits in his possession, and afterwards he announces the result of the computation. The complexity of a protocol is the number of qubits sent.

In an exact quantum protocol the result has to be correct with certainty. $Q_E(f)$ is the minimal complexity of an exact quantum protocol for a function f.

In a bounded error protocol the output has to be correct with probability $1-\epsilon$ (for $1/2 > \epsilon > 0$). The bounded error quantum one-way communication complexity of a function f is $Q_\epsilon(f)$, the minimal complexity of a bounded error quantum one-way protocol for f, and we set $Q(f) = Q_{1/3}(f)$.

Quantum Las Vegas protocols are defined in a manner similar to their probabilistic counterparts; the complexity measure notation is $Q_{0,\epsilon}(f)$.

Cleve and Buhrman [9] *consider a different model of quantum communication: Before the start of the protocol Alice and Bob own a set of qubits whose state may be entangled but must be independent of the inputs. Then as above a quantum communication protocol is used. We use the superscript* pub *to denote the complexity in this model.*

It is possible to simulate the model with entangled qubits by allowing first an arbitrary finite communication independent of the inputs, followed by an ordinary protocol. By measuring distributed EPR pairs it is possible to simulate classical public randomness. The technique of superdense coding of [5] allows one in the model with prior entanglement to send n bits of classical information with $\lceil n/2 \rceil$ qubits.

4.5. Quantum circuits and formulas. Besides quantum Turing machines quantum circuits [11] are a universal model of quantum computation (see [41]) and are generally easier to handle in descriptions of quantum algorithms. A more general model of quantum circuits in which superoperator gates work on density matrices is described in [1]. We begin with the basic model.

DEFINITION 4.14. *A unitary quantum gate with k inputs and k outputs is specified by a unitary operator $U : \mathbb{C}^{2^k} \to \mathbb{C}^{2^k}$.*

A quantum circuit consists of unitary quantum gates with $O(1)$ inputs and outputs each, plus a set of inputs to the circuits, which are connected to an acyclic directed graph, in which the inputs are sources. Sources are labeled by Boolean constants or by input variables. Edges correspond to qubits; the circuit uses as many qubits as it has sources. One designated qubit is the output qubit. A quantum circuit computes a unitary transformation on the source qubits in the obvious way. In the end the output qubit is measured in the standard basis.

The size of a quantum circuit is the number of its gates; the depth is the length of the longest path from an input to the output.

A quantum circuit computes a function with a bounded error if it gives the right output with probability at least $2/3$ for all inputs.

A quantum circuit computes a Boolean function with a one-sided error if it has a bounded error and furthermore never erroneously accepts.

A pair of quantum circuits computes a Boolean function f in the Las Vegas sense, if the first is a one-sided error circuit for f and the second is a one-sided error circuit for $\neg f$.

A quantum circuit computes a function exactly if it makes no error.

The definition of Las Vegas circuits is motivated by the fact that we can easily verify the computation of a pair of one-sided error circuits for f and $\neg f$ as in the classical case; see the proof of Corollary 3.13.

We are interested in restricted types of circuits, namely, quantum formulas [41].

DEFINITION 4.15. *A quantum formula is a quantum circuit with the following additional property: For each source there is at most one path connecting it to the output. The length or size of a quantum formula is the number of its sources.*

Apart from the Boolean input variables a quantum formula is allowed to read Boolean constants only. There is only one final measurement. We also call the model from [41] *pure* quantum formulas. Compare also the definitions in [37].

In [1] a more general model of quantum circuits is studied, in which superoperators work on density matrices.

DEFINITION 4.16. *A superoperator gate g of order (k, l) is a trace-preserving, completely positive map from the density matrices on k qubits to the density matrices on l qubits.*

A quantum superoperator circuit is a directed acyclic graph with inner vertices marked by superoperator gates with fitting fan-in and fan-out. The sources are marked with input variables or Boolean constants. One gate is designated as the output.

A function is computed as follows. In the beginning the sources are each assigned a density matrix corresponding to the Boolean values determined by the input or by a constant. The Boolean value 0 corresponds to $|0\rangle\langle 0|$, 1 to $|1\rangle\langle 1|$. The overall state of the qubits involved is the tensor product of these density matrices.

Then the gates are applied in an arbitrary topological order. Applying a gate means applying the superoperator composed of the gate's superoperator on the chosen qubits for the gate and the identity superoperator on the remaining qubits.

In the end the state of the output qubit is supposed to be a classical probability distribution on $|0\rangle$ and $|1\rangle$.

The following fact from [1] allows one to apply gates in an arbitrary topological ordering.

FACT 4.17. *Let C be a quantum superoperator circuit, and let C_1 and C_2 be two sets of gates working on different sets of qubits. Then for all density matrices ρ on* *the qubits in the circuit the result of C_1 applied to the result of C_2 on ρ is the same as the result of C_2 applied to the result of C_1 on ρ.*

Let two arbitrary topological orderings of the gates in a quantum superoperator circuit be given. The result of applying the gates in one ordering is the same as the result of applying the gates in the other ordering for any input density matrix.

One more aspect is interesting in the definition of quantum formulas: We want to allow quantum formulas to access multiple read random inputs, just as fair probabilistic formulas. This makes it possible to simulate the latter model. Instead of random variables we allow the quantum formulas to read an arbitrary nonentangled state. A pure state on k qubits is called nonentangled if it is the tensor product of k states on one qubit each. A mixed state is nonentangled if it can be expressed as a probabilistic ensemble of nonentangled pure states. Note that a classical random variable read k times can be modeled as $|1^k\rangle$ with probability $1/2$ and $|0^k\rangle$ with probability $1/2$.

We restrict our definition to gates with fan-in 2; the set of quantum gates with fan-in 2 is known to be universal [4].

Definition 4.18. *A generalized quantum formula is a quantum superoperator circuit with fan-out 1/fan-in 2 gates together with a fixed nonentangled mixed state. The sources of the circuit are either labeled by input variables or may access a qubit of the state. Each qubit of this state may be accessed only by one gate.*

As proved in [1] Fact 4.7 implies that quantum superoperator circuits with constant fan-in are asymptotically as efficient as quantum circuits with constant fan-in. The same holds for quantum formulas. The essential difference between pure and generalized quantum formulas is the availability of multiple read random bits.

4.6. Programmable quantum gates. For simulations of quantum mechanical formulas by communication protocols we will need a programmable quantum gate. Such a gate allows Alice to communicate a unitary operation as a program stored in some qubits to Bob, who then applies this operation to some of his qubits.

Formally we have to look for a unitary operator G, with

$$G(|d\rangle \otimes |P_U\rangle) = U(|d\rangle) \otimes |P'_U\rangle.$$

Here $|P_U\rangle$ is the "code" of a unitary operator U and $|P'_U\rangle$ some leftover of the code.

The bad news is that such a programmable gate does not exist, as proved in [34]. Note that in the classical case such gates are easy to construct.

Fact 4.19. *If N different unitary operators (pairwise different by more than a global phase) can be implemented by a programmable quantum gate, then the gate needs a program of length $\log N$.*

Since there are infinitely many unitary operators on just one qubit there is no programmable qubit with finite program length implementing them all. The proof uses the fact that the gate works deterministically, and actually a probabilistic solution to the problem exists.

We now sketch a construction of Nielsen and Chuang [34]. For the sake of simplicity we describe just the construction for unitary operations on one qubit.

The program of a unitary operator U is

$$|P_U\rangle = \frac{1}{\sqrt{2}}(|0\rangle U|0\rangle + |1\rangle U|1\rangle).$$

The gate receives as input $|d\rangle \otimes |P_U\rangle$. The gate then measures the first and second qubits in the basis $\{|\Phi^+\rangle, |\Phi^-\rangle, |\Psi^+\rangle, |\Psi^-\rangle\}$. Then the third qubit is used as a result.

For a state $|d\rangle = a|0\rangle + b|1\rangle$ the input to the gate is

$$[a|0\rangle + b|1\rangle]\frac{|0\rangle U|0\rangle + |1\rangle U|1\rangle}{\sqrt{2}} = \frac{1}{2}\left[|\Phi^+\rangle(aU|0\rangle + bU|1\rangle) + |\Phi^-\rangle(aU|0\rangle - bU|1\rangle)\right.$$
$$\left. + |\Psi^+\rangle(aU|1\rangle + bU|0\rangle) + |\Psi^-\rangle(aU|1\rangle - bU|0\rangle)\right].$$

Thus the measurement produces the correct state with probability 1/4, and moreover the result of the measurement indicates whether the computation was done correctly. Also, given this measurement result we know exactly which unitary "error" operation has been applied before the desired operation. We now state Nielsen and Chuang's result.

FACT 4.20. *There is a probabilistic programmable quantum gate with m input qubits for the state plus $2m$ input qubits for the program, which implements every unitary operation on m qubits and succeeds with probability $1/2^{2m}$. The result of a measurement done by the gate indicates whether the computation was done correctly and which unitary error operation has been performed.*

Also note that it is easy to construct an approximate programmable quantum gate in the following sense. For any error parameter ϵ we may discretize the set of superoperators to a finite set so that for each superoperator T there is an operator T' from the finite set, such that for each density matrix ρ we have that $T\rho$ is ϵ-close to $T'\rho$. Then we can construct a gate that receives the classical description of one of these finitely many superoperators as a program.

5. One-way communication complexity: The nondeterministic and the quantum cases.

5.1. A lower bound for limited nondeterminism. In this section we investigate nondeterministic one-way communication with a limited number of nondeterministic bits. Analogous problems for many-round communication complexity have been addressed in [18], but in this section we again consider asymmetric problems, for which the one-way restriction is essential.

It is easy to see that if player Bob has m input bits, then m nondeterministic bits are the maximum player Alice needs. Since the nondeterministic communication complexity without any limitation on the number of available nondeterministic bits is at most m, Alice can just guess the communication and send it to Bob in case it is correct with respect to her input and leads to acceptance. Bob can then check the same for his input. Thus an optimal protocol can be simulated.

For the application to lower bounds on formula size we are again interested in functions with an asymmetric input partition; i.e., Alice receives much more inputs than Bob. For nontrivial results thus the number of nondeterministic bits must be smaller than the number of Bob's inputs.

A second observation is that using s nondeterministic bits can reduce the communication complexity from the deterministic one-way communication complexity d to $d/2^s$ in the best case. If s is sublogarithmic, strong lower bounds follow already from the deterministic lower bounds, e.g., $N_{\epsilon \log n}(\neg EQ) \geq n^{1-\epsilon}$, while $N_{\log n}(\neg EQ) = O(\log n)$. On the other hand, we have the following.

LEMMA 5.1.

$$N_s(f) = c \Rightarrow N_c(f) \leq c.$$

Proof. In a protocol with communication c at most 2^c different messages can be sent (for all inputs). To guess such a message c nondeterministic bits are sufficient. □

Hence it is unnecessary for a nondeterministic protocol to use more nondeterministic bits than communication. We are interested in determining how large the difference between nondeterministic one-way communication complexity with s nondeterministic bits and unrestricted nondeterministic communication complexity may be. Therefore we consider the maximal such gap as a function G.

Corollary 5.2. *Let $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ be a Boolean function and $G : \mathbb{N} \to \mathbb{N}$ a monotone increasing function, with $N(f) = c$ and $N_s(f) = G(c)$ for some s.*

Then $N_{G^{-1}(n)}(f) \leq c$ and hence $s \leq G^{-1}(n)$, where $G^{-1}(x) = \min\{y | G(y) \geq x\}$.

Proof. $G(c) \leq n$ and hence $c \leq G^{-1}(n)$. □

The range of values of s for which a gap G between $N(f)$ and $N_s(f)$ is possible is thus limited. If, e.g., an exponential difference $G(x) = 2^x$ holds, then $s \leq \log n$. If $G(x) = r \cdot x$, then $s \leq n/r$.

We now show a gap between nondeterministic one-way communication complexity with s nondeterministic bits and unlimited nondeterministic communication complexity. First we define the family of functions exhibiting this gap. Denote by $\mathcal{P}(a, b)$ the set of size b subsets of a size a universe.

Definition 5.3. *Let $DI_{n,s}$ be the following Boolean function for $1 \leq s \leq n$:*

$$\begin{aligned} DI_{n,s}(x_1, \dots, x_n; x_{n+1}) = 1 \iff & \forall i : x_i \in \mathcal{P}(n^3, s) \\ & \wedge \exists i : |\{j | j \neq i; x_i \cap x_j \neq \emptyset\}| \geq s. \end{aligned}$$

Note that the function has $\Theta(sn \log n)$ input bits in a standard encoding. We consider the partition of inputs in which Bob receives the set x_{n+1} and Alice all other sets. The upper bounds in the following lemma are trivial, since Bob receives only $O(s \log n)$ input bits.

Lemma 5.4.

$$\begin{aligned} N_{O(s \log n)}(DI_{n,s}) &= O(s \log n). \\ D^B(DI_{n,s}) &= O(s \log n). \end{aligned}$$

The lower bound we present now results in a near optimal difference between nondeterministic (one-way) communication and limited nondeterministic one-way communication. Limited nondeterministic one-way communication has also been studied subsequently to this work in [17]. There a tradeoff between the consumption of nondeterministic bits and the one-way communication is demonstrated (i.e., with more nondeterminism the communication gradually decreases). Here we describe a fundamentally different phenomenon of a threshold type: Nondeterministic bits do not help much, until a certain number of them are available, when quite quickly the optimal complexity is attained. For more results of this type see [22].

Theorem 5.5. *There is a constant $\epsilon > 0$, so that for $s \leq n$*

$$N_{\epsilon s}(DI_{n,s}) = \Omega(ns \log n).$$

Proof. We have to show that all nondeterministic one-way protocols computing $DI_{n,s}$ with ϵs nondeterministic bits need much communication.

A nondeterministic one-way protocol with ϵs nondeterministic bits and communication c induces a cover of the communication matrix with $2^{\epsilon s}$ Boolean matrices having the following properties: Each 1-entry of the communication matrix is a 1-entry in at least one of the Boolean matrices, no 0-entry of the communication matrix is a 1-entry in any of the Boolean matrices, and furthermore the set of rows appearing in those matrices has size at most 2^c. This set of matrices is obtained by fixing the nondeterministic bits and taking the communication matrices of the resulting deterministic protocols. Note that the rows of the communication matrices of the deterministic protocols correspond to messages. We will deduce the lower bound from the weaker property that *each* of the Boolean matrices covering the communication matrix uses at most 2^c different rows. This can be used to show that the lower bound holds even for protocols with limited, but public, nondeterminism.

We start by constructing a submatrix of the communication matrix with some useful properties and then show the theorem for this "easier" problem.

Partition the universe $\{1, \dots, n^3\}$ in n disjoint sets $U_1, \dots, U_n$, with $|U_i| = n^2 = m$. Then choose vectors of n size s subsets of the universe, so that the ith subset is from U_i. Thus the n subsets of a vector are pairwise disjoint. Now the protocol has to determine whether the set of Bob intersects nontrivially with s of the sets given to Alice.

We restrict the set of inputs further. There are $\binom{m}{s}$ different size s subsets of U_i. We choose a set of such subsets so that each pair of them has no more than $s/2$ common elements. To do so we start with any subset and remove all subsets having more than $s/2$ common elements with any already chosen subset. This process can be repeated until all size s subsets are either chosen or discarded. We end with a set of size s subsets of U_i, whose elements have pairwise no more than $s/2$ common elements. In every step at most $\binom{s}{s/2}\binom{m}{s/2}$ subsets are discarded; thus we choose at least

$$\frac{\binom{m}{s}}{\binom{s}{s/2}\binom{m}{s/2}} \geq \left(\frac{m}{s}\right)^{s/2} / 2^{3s/2} \tag{5.1}$$

sets.

As described we draw Alice's inputs as vectors of sets, where the set at position i is drawn from the set of subsets of U_i we have just constructed. These inputs are identified with the rows of the submatrix of the communication matrix. The columns of the submatrix are restricted to elements of $U_1 \cup \{\top\} \times \cdots \times U_n \cup \{\top\}$, for which s positions are occupied; i.e., $n - s$ positions carry the extra symbol $\top$ which stands for "no element." Call the constructed submatrix M.

Now assume there is a protocol computing the restricted problem with communication matrix M. Fixing the nondeterministic bits gives us a deterministic protocol. If a nondeterministic protocol uses r nondeterministic bits, then there are 2^r such deterministic protocols, and at least one of them accepts a fraction of $1/2^r$ of the ones in M, where $r = \epsilon s$. We now show that such a matrix must have many different rows, which corresponds to a large amount of communication.

Each row of M (being a vector of zeros and ones) also corresponds to a vector of n sets, the associated input of Alice. A position i is called a *difference position* for a pair of such sequences if they have different sets at position i. According to our construction these sets have no more than $s/2$ elements in common.

We say a set of rows has k difference positions if there are k positions $i_1, \dots, i_k$, so that for each i_l there are two rows in the set for which i_l is a difference position.

We now show that each row of M' containing "many" ones does not "fit" on many rows of M, i.e., contains ones these do not have. Since M' has a one-sided error, the rows of M' are either sparse or cover only a few rows of M. Observe that each row of M has exactly $\binom{n}{s}s^s$ ones.

LEMMA 5.6. *Let z be a row of M', appearing several times in M'. The rows of M, in whose place in M the row z appears in M', may have δn difference positions. Then z contains at most $2\binom{n}{s}s^s/2^{\delta s/6}$ ones.*

Proof. Several rows of M having δn difference positions are given, and the ones of z occur in all of these rows. Let C be the set of $\binom{n}{s}s^s$ columns/sets being the ones in the first such row. All other columns are forbidden and may not be ones in z.

A column in C is chosen randomly by choosing s out of n positions and then one of s elements for each position. Let $k = \delta s$. We have to show an upper bound on the number of ones in z, and we analyze this number as the probability of getting a one when choosing a column in C. The probability of getting a one is at most the probability that the chosen positions have a nontrivial intersection with less than $k/2$ sets U_i at difference positions i (call this event E) plus the probability of getting a one under the condition of event $\overline{E}$, following the general formula $\mathrm{Prob}(A) \leq \mathrm{Prob}(A|E) + \mathrm{Prob}(\overline{E})$.

We first count the columns in C, which have a nontrivial intersection with at most $k/2$ of the sets U_i at difference positions i. Consider the slightly different experiment in which s times independently one of n positions is chosen; hence positions may be chosen more than one time. Now expected $\delta s = k$ difference positions are chosen. Applying Chernov's inequality yields that, with probability at most

$$e^{-\frac{1}{4\cdot 2}\cdot k} \leq 2^{-\delta s/6},$$

at most $k/2$ difference positions occur. When choosing a random column in C instead, this probability is even smaller, since now positions are chosen without repetitions. Thus the columns in C, which "hit" less than $k/2$ difference positions, contribute at most $2^{-\delta s/6}\binom{n}{s}s^s$ ones to z.

Now consider the columns/sets in C, which intersect at least $k/2$ of the U_i at difference positions i. Such a column/set fits on all of the rows, if the element at each position not bearing a $\top$ lies in the intersection of all sets in the rows at position i. At each difference position there are two rows, which hold different sets at that position, and those sets have no more than $s/2$ common elements.

Fix an arbitrary set of positions such that at least $k/2$ difference positions are included. The next step of choosing a column in C consists of choosing one of s elements for each position. But if a position is a difference position, then at most $s/2$ elements satisfy the condition of lying in the sets held by all of the rows at that position. Thus the probability of fitting on all of the rows is at most $2^{-k/2}$, and at most $\binom{n}{s}s^s/2^{k/2}$ such columns can be a one in z.

Overall only a fraction of $2^{-\delta s/6+1}$ of all columns in C can be ones in z. □

At least one-half of all ones in M' lie in rows containing at least $\geq \binom{n}{s}s^s/2^{r+1}$ ones. Lemma 5.6 tells us that such a row fits only on a set of rows of M having no more than δn difference positions, where $r + 1 = \delta s/6 - 1$. Hence such a row can cover at most all of the ones in $\binom{m}{s}^{\delta n}$ rows of M and therefore only $\binom{m}{s}^{\delta n}\binom{n}{s}s^s$ ones.

According to (5.1) at least $(m/s)^{sn/2}\binom{n}{s}s^s/(2^{3sn/2}2^{r+1})$ ones are covered by such rows; hence

$$\frac{(m/s)^{sn/2}\binom{n}{s}s^s}{\binom{m}{s}^{\delta n}\binom{n}{s}s^s 2^{3sn/2}2^{r+1}} \geq \frac{(m/s)^{sn/2}}{(em/s)^{6\epsilon sn+12n}2^{3sn/2}2^{\epsilon s+1}}$$
$$= 2^{\Omega(sn\log n)}$$

rows are necessary (for $\epsilon = 1/20$ and $n \geq s \geq 400$). □

5.2. Quantum one-way communication. Our first goal in this section is to prove that the VC-dimension lower bound for randomized one-way protocols (Fact 2.9) can be extended to the quantum case. To achieve this we first prove a linear lower bound on the bounded error quantum communication complexity of the index function IX_n and then describe a reduction from the index function IX_d to any function with VC dimension d, thus transferring the lower bound. It is easy to see that $VC(IX_n) = n$, and thus the bounded error probabilistic one-way communication complexity is large for that function.

The problem of *random access quantum coding* has been considered in [2]. In a n, m, ϵ-random access quantum code all Boolean n-bit words x have to be mapped to states of m qubits each, so that for $i = 1, \ldots, n$ there is an observable, so that measuring the quantum code with that observable yields the bit x_i with probability $1 - \epsilon$. The quantum code is allowed to be a mixed state. The following is a result from [2].

FACT 5.7. *For every n, m, ϵ-random access quantum coding $m \geq (1 - H(\epsilon))n$.*

It is easy to see that the problem of random access quantum coding is equivalent to the construction of a quantum one-way protocol for the index function. If there is such a protocol, then the messages can serve as mixed state codes, and if there is such a code, the codewords can be used as messages. We can thus deduce a lower bound for IX_n in the model of one-way quantum communication complexity without prior entanglement.

We now give a proof that can also be adapted to the case of allowed prior entanglement. The proof follows Nayak's idea, who also obtained the generalization to the case of prior entanglement in his thesis [31].

THEOREM 5.8. *$Q_\epsilon(IX_n) \geq (1 - H(\epsilon))n$.*

$Q_\epsilon^{pub}(IX_n) \geq (1 - H(\epsilon))n/2$.

Proof. Let M be the register containing the message sent by Alice, and let X be a register holding a uniformly random input to Alice. Then σ_{XM} denotes the state of Alice's qubits directly before the message is sent. σ_M is the state of a random message. Now every bit is decodable with probability $1 - \epsilon$, and thus $S(X_i : M) \geq 1 - H(\epsilon)$ for all i. To see this consider $S(X_i : M)$ as the Holevo information of the following ensemble:

$$\sigma_{i,0} = \sum_{x:x_i=0} \frac{1}{2^{n-1}} \sigma_M^x$$

with probability 1/2 and

$$\sigma_{i,1} = \sum_{x:x_i=1} \frac{1}{2^{n-1}} \sigma_M^x$$

with probability 1/2, where σ_M^x is the density matrix of the message on input x. The information obtainable on x_i by measuring σ_M must be at $1 - H(\epsilon)$ due to Fano's inequality (Fact 4.2), and thus the Holevo information of the ensemble is at least $1 - H(\epsilon)$; hence $S(X_i : M) \geq 1 - H(\epsilon)$.

But then $S(X : M) \geq (1 - H(\epsilon)n$ (since all X_i are mutually independent). $S(X : M) \leq S(M)$ using Lemma 4.12, since X and M are not entangled. Thus the number of qubits in M is at least $(1 - H(\epsilon))n$.

Now we analyze the complexity of IX_n in the one-way communication model with entanglement.

The density matrix of the state induced by a uniformly random input on X, the message M, and the qubits E_A, E_B containing the prior entanglement in the possession of Alice and Bob is $\sigma_{XME_AE_B}$. Here E_A contains those qubits of the entangled state Alice keeps; note that some of the entangled qubits will usually belong to M. Tracing out X and E_A we receive a state σ_{ME_B}, which is accessible to Bob. Now every bit of the string in X is decodable; thus $S(X_i : ME_B) \geq 1 - H(\epsilon)$ for all i as before. But then also $S(X : ME_B) \geq (1 - H(\epsilon)n$, since all of the X_i are mutually independent.

$S(X : ME_B) = S(X : E_B) + S(X : M|E_B) \leq 2S(M)$ by an application of the Araki–Lieb inequality; see Fact 4.11. Note that $S(X : E_B) = 0$. So the number of qubits in M must be at least $(1 - H(\epsilon))n/2$. □

Note that the lower bound shows that two-round deterministic communication complexity can be exponentially smaller than one-way quantum communication complexity. For a more general quantum communication round hierarchy see [25].

THEOREM 5.9. *For all functions f : $Q_\epsilon(f) \geq (1 - H(\epsilon))VC(f)$ and $Q^{pub}_\epsilon(f) \geq (1 - H(\epsilon))VC(f)/2$.*

Proof. We now describe a reduction from the index function to f. Assume $VC(f) = d$; i.e., there is a set $S = \{s_1, \ldots, s_d\}$ of inputs for Bob, which is shattered by the set of functions $f(x, .)$. The reduction then goes from IX_d to f.

For each $R \subseteq S$ let c_R be the incidence vector of R (having length d). c_R is a possible input for Alice when computing the index function IX_d. For each R choose some x_R, which separates this subset from the rest of S, i.e., so that $f(x_R, y) = 1$ for all $y \in R$ and $f(x_R, y) = 0$ for all $y \in S - R$.

Assume a protocol for f is given. To compute the index function the players do the following. Alice maps c_R to x_R. Bob's inputs i are mapped to the s_i. Then $f(x_R, s_i) = 1 \iff s_i \in R \iff c_R(i) = 1$.

In this manner a quantum protocol for f must implicitly compute IX_d. According to Theorem 5.8 the lower bounds follow. □

As an application of the previous theorem we get lower bounds for the disjointness problem in the model of quantum one-way communication complexity.

COROLLARY 5.10. *$Q_\epsilon(DISJ_n) \geq (1 - H(\epsilon))n$.*

$Q^{pub}_\epsilon(DISJ_n) \geq (1 - H(\epsilon))n/2$.

The first result has independently been obtained by Buhrman and de Wolf [8]. Note that the obtained lower bound method is not tight in general. There are functions for which an unbounded gap exists between the VC dimension and the quantum one-way communication complexity [24].

Now we turn to the exact and Las Vegas quantum one-way communication complexity. For classical one-way protocols it is known that Las Vegas communication complexity is at most a factor 1/2 better than deterministic communication for total functions; see Fact 2.10.

THEOREM 5.11. *For all total functions f:*

$Q_E(f) = D(f)$,

$Q_{0,\epsilon}(f) \geq (1 - \epsilon)D(f)$.

Proof. Let $row(f)$ be the number of different rows in the communication matrix of $f(x, y)$. According to Fact 2.7, $D(f) = \lceil \log row(f) \rceil$. We assume in the following that the communication matrix consists of pairwise different rows only.

We will show that any Las Vegas one-way protocol which gives up with probability at most $\epsilon \geq 0$ for some function f having $row(f) = R$ must use messages with von Neumann entropy at least $(1 - \epsilon) \log R$, when started on a uniformly random input. Inputs for Alice are identified with rows of the communication matrix. We then conclude that the Hilbert space of the messages must have dimension at least $R^{1-\epsilon}$, and hence at least $(1-\epsilon)\log R$ qubits have to be sent. This gives us the second lower bound of the theorem. The upper bound of the first statement is trivial; the lower bound of the first statement follows by taking $\epsilon = 0$.

We now describe a process in which rows of the communication matrix are chosen randomly bit per bit. Let p be the probability of having a 0 in column 1 (i.e., the number of 0s in column 1 divided by the number of rows). Then a 0 is chosen with probability p, a 1 with probability $1-p$. Afterwards the set of rows is partitioned into the set I_0 of rows starting with a 0 and the set I_1 of rows starting with a 1. When $x_1 = b$ is chosen, the process continues with I_b and the next column.

Let ρ_y be the density matrix of the following mixed state: The (possibly mixed) message corresponding to a row starting with y is chosen uniformly over all such rows.

The probability that a 0 is chosen after y is called p_y, and the number of different rows beginning with y is called row_y.

We want to show via induction that $S(\rho_y) \geq (1-\epsilon)\log row_y$. Surely $S(\rho_y) \geq 0$ for all y.

Recall that Bob can determine the function value for an arbitrary column with the correctness guarantee of the protocol.

Then with Lemma 4.10, $S(\rho_y) \geq p_y S(\rho_{y0}) + (1-p_y)S(\rho_{y1}) + (1-\epsilon)H(p_y)$, and via induction

$$\begin{aligned} S(\rho_y) &\geq p_y((1-\epsilon)\log row_{y0}) \\ &\quad + (1-p_y)((1-\epsilon)\log row_{y1}) + (1-\epsilon)H(p_y) \\ &= (1-\epsilon)[p_y \log(p_y row_y) \\ &\quad + (1-p_y)\log((1-p_y)row_y) + H(p_y)] \\ &= (1-\epsilon)\log row_y. \end{aligned}$$

We conclude that $S(\rho) \geq (1-\epsilon)\log row(f)$ for the density matrix ρ of a message to a uniformly random row. Hence the lower bound on the number of qubits holds. □

We now again consider the model with prior entanglement.

THEOREM 5.12. *For all total functions f:*
$Q_E^{pub}(f) = \lceil D(f)/2 \rceil$,
$Q_{0,\epsilon}^{pub}(f) \geq D(f)(1-\epsilon)/2$.

The upper bound follows from superdense coding [5]. Instead of the lower bounds of the theorem we prove a stronger statement. We consider an extended model of quantum one-way communication that will be useful later.

In a *nonstandard* one-way quantum protocol Alice and Bob are allowed to communicate in arbitrarily many rounds; i.e., they can exchange many messages. But Bob is not allowed to send Alice a message, so that the von Neumann information between Bob's input and all of Alice's qubits is larger than 0. The communication complexity of a protocol is the number of qubits sent by Alice in the worst case. The model is at least as powerful as the model with prior entanglement, since Bob may, e.g., generate some EPR pairs and send one qubit of each pair to Alice, and then Alice may send a message as in a protocol with prior entanglement.

LEMMA 5.13. *For all functions f a nonstandard quantum one-way protocol with a bounded error must communicate at least $(1-H(\epsilon))VC(f)/2$ qubits from Alice to Bob.*

For all total functions f a nonstandard quantum one-way protocol

1. *with exact acceptance must communicate at least $\lceil D(f)/2 \rceil$ qubits from Alice to Bob;*

2. *with Las Vegas acceptance and success probability* $1-\epsilon$ *must communicate at least* $(1-\epsilon)D(f)/2$ *qubits from Alice to Bob.*

Proof. In this proof we always call the qubits available to Alice P and the qubits available to Bob Q for simplicity disregarding that these registers change during the course of the protocol. We assume that the inputs are in registers X, Y and are never erased or changed in the protocol. Furthermore we assume that for all fixed values x, y of the inputs the remaining global state is pure.

For the first statement it is again sufficient to investigate the complexity of the index function.

Let σ_{XYPQ} be the state for random inputs in X, Y for Alice and Bob, with qubits P and Q in the possession of Alice and Bob, respectively. Since Bob determines the result, it must be true that in the end of the protocol $S(X_Y : YQ) \geq 1 - H(\epsilon)$, since the value X_Y can be determined from Bob's qubits with probability $1-\epsilon$. It is always true in the protocol that $S(XP : Y) = 0$. Let $\rho_P^{X=x,Y=y}$ be the density matrix of P for fixed inputs $X = x$ and $Y = y$. Then we have that for all x, y, y': $\rho_P^{X=x,Y=y} = \rho_P^{X=x,Y=y'}$.

$\rho_{PQ}^{X=x,Y=y}$ purifies $\rho_P^{X=x,Y=y}$. Then the following fact from [29] and [28] tells us that all y and corresponding states of Q are "equivalent" from the perspective of Alice.

FACT 5.14. *Assume* $|\phi_1\rangle$ *and* $|\phi_2\rangle$ *are pure states in a Hilbert space* $H \otimes K$, *so that* $\mathrm{Tr}_K|\phi_1\rangle\langle\phi_1| = \mathrm{Tr}_K|\phi_2\rangle\langle\phi_2|$.

Then there is a unitary transformation U *acting on* K, *so that* $I \otimes U|\phi_1\rangle = |\phi_2\rangle$ *(for the identity operator* I *on* H*).*

Thus there is a local unitary transformation applicable by Bob alone, so that $\rho_{PQ}^{X=x,Y=y}$ can be changed to $\rho_{PQ}^{X=x,Y=y'}$. Hence for all i we have $S(QY : X_i) \geq 1 - H(\epsilon)$, and thus $S(X : QY) \geq (1 - H(\epsilon))n$.

In the beginning $S(X : QY) = 0$. Then the protocol proceeds w.l.o.g. so that each player applies a unitary transformation on his qubits and then sends a qubit to the other player. Since the information cannot increase by local operations, it is sufficient to analyze what happens if qubits are sent. When Bob sends a qubit to Alice, $S(X : QY)$ is not increased. When Alice sends a qubit to Bob, then Q is augmented by a qubit M, and $S(X : QMY) \leq S(X : QY) + S(XQY : M) \leq S(X : QY) + 2S(M) \leq S(X : QY) + 2$ due to Fact 4.11. Thus the information can increase only when Alice sends a qubit and always by at most 2. The lower bound follows.

Now we turn to the second part. We consider the same situation as in the proof of Theorem 5.11. Let σ_P^{rc} denote the density matrix of the qubits P in Alice's possession under the condition that the input row is r and the input column is c. Clearly σ_{PQ}^{rc} (containing also Bob's qubits) is a purification of σ_P^{rc}. Again $\sigma_P^{rc} = \sigma_P^{rc'}$ for all r, c, c', and according to Fact 5.14 for all c and all corresponding states of Q, it is true that Bob can switch locally between them. Hence it is possible for Bob to compute the function for an arbitrary column.

The probability of choosing a 0 after a prefix y of a row is again called p_y, and the number of different rows beginning with y is called row_y. ρ_y contains the state of Bob's qubits at the end of the protocol if a random row starting with y is chosen uniformly (and some fixed column c is chosen). Surely $S(\rho_y) \geq 0$ for all y. Since Bob can change his column (and the corresponding state of Q) by a local unitary transformation, he is able to compute the function for an arbitrary column, always with the success probability of the protocol, at the end. With Lemma 4.10 $S(\rho_y) \geq p_y S(\rho_{y0}) + (1-p_y)S(\rho_{y1}) + (1-\epsilon)H(p_y)$.

At the end of the protocol thus $S(\sigma_Q^c) = S(\rho) \geq (1-\epsilon)\log row(f) + \sum_r \frac{1}{row(f)} S(\sigma_Q^{rc})$ for all c. Thus the Holevo information of the ensemble, in which $\rho_r = \sigma_Q^{rc}$ is chosen with probability $1/row(f)$, is at least $(1-\epsilon)\log row(f)$. Let σ_{RPQ} be the density matrix of rows, qubits of Alice and Bob. It follows that $S(R:Q) \geq (1-\epsilon)\log row(f)$, and as before at least half that many qubits have to be sent from Alice to Bob. □

6. More lower bounds on formula size.

6.1. Nondeterminism and formula size. Let us first mention that any nondeterministic circuit can easily be transformed into an equivalent nondeterministic formula without increasing size by more than a constant factor. To do so one simply guesses the values of all internal gates and then verifies that all of these guesses are correct and that the circuit accepts. This is a big AND over test involving $O(1)$ variables, which can be implemented by a Boolean formula in conjunctive normal form. Hence large lower bounds for nondeterministic formula size are very hard to prove, since even nonlinear lower bounds for the size of deterministic circuits computing some explicit functions are unknown. We now show that formulas with limited nondeterminism are more approachable. We start by introducing a variant of the Nečiporuk method, this time in terms of nondeterministic communication:

Definition 6.1. *Let f be a Boolean function with n input variables and $y_1 \dots y_k$ be a partition of the inputs in k blocks.*

Player Bob receives the inputs in y_i, and player Alice receives all other inputs. The nondeterministic one-way communication complexity of with s nondeterministic bits of f under this input partition is called $N_s(f_i)$. Define the s-nondeterministic Nečiporuk function as $1/4\sum_{i=1}^k N_s(f_i)$.

Lemma 6.2. *The s-nondeterministic Nečiporuk function is a lower bound for the length of nondeterministic Boolean formulas with s nondeterministic bits.*

The proof is analogous to the proof of Theorem 3.5. Again protocols simulate the formula in k communication games. This time Alice fixes the nondeterministic bits by herself, and no probability distribution on formulas is present.

We will apply the above methodology to the following language.

Definition 6.3. *Let $AD_{n,s}$ denote the following language (for $1 \leq s \leq n$):*

$$\begin{aligned} AD_{n,s} = \{(x_1, \dots, x_{n+1}) | \forall i : x_i \in \mathcal{P}(n^3, s),& \\ x_i \textit{ is written in sorted order}& \\ \wedge \exists i : |\{j | j \neq i; x_i \cap x_j \neq \emptyset\}| \geq s\}.& \end{aligned}$$

Theorem 6.4. *Every nondeterministic formula with s nondeterministic bits for $AD_{n,20s}$ has length at least $\Omega(n^2 s \log n)$.*

$AD_{n,s}$ can be computed by a nondeterministic formula of length $O(ns^2 \log n)$, which uses $O(s \log n)$ nondeterministic bits (for $s \geq \log n$).

Proof. For the lower bound we use the methodology we have just described. We consider the $n+1$ partitions of the inputs, in which Bob receives the set x_i and Alice all other sets. The function they have to compute now is the function $DI_{n,s}$ from Definition 5.3. In Theorem 5.5 a lower bound of $\Omega(ns \log n)$ is shown for this problem; hence the length of the formula is $\Omega(n \cdot ns \log n)$.

For the upper bound we proceed as follows: The formula guesses (in binary) a number i with $1 \leq i \leq n+1$ and pairs $(j_1, w_1), \dots, (j_s, w_s)$, where $1 \leq j_k \leq n+1$ and $1 \leq w_k \leq n^3$ for all $k = 1, \dots, s$. The number i indicates a set, and the pairs are witnesses that set i and set j_k intersect on element w_k.

The formula does the following tests. First there is a test of whether all sets consist of s sorted elements. For this ns comparisons of the form $x_i^j < x_i^{j+1}$ suffice, which can be realized with $O(\log^2 n)$ gates each. Since $s \geq \log n$ overall $O(ns^2 \log n)$ gates are enough.

The next test is whether $j_1 < \cdots < j_s$. This makes sure that witnesses for s different sets have been guessed. Also $i \neq j_k$ for all k must be tested.

Then the formula tests whether for all $1 \leq l \leq n+1$ the following holds: If $l = i$, then all guessed elements are in x_l; if $1 \leq l \leq n+1$ and $1 \leq k \leq s$, the formula also tests, whether $l = j_k$ implies, that $w_k \in x_l$.

All of these tests can be done simultaneously by a formula of length $O(ns^2 \log n)$. □

For $0 < \epsilon \leq 1/2$ let $s = n^{\frac{\epsilon}{1-\epsilon}}$; then the lower bound for limited nondeterministic formulas is $\Omega(N^{2-\epsilon}/\log^{1-\epsilon} N)$, with $N^\epsilon/\log^\epsilon N$ nondeterministic bits allowed. $O(N^\epsilon \log^{1-\epsilon} N)$ nondeterministic bits suffice to construct a formula having length $O(N^{1+\epsilon}/\log^\epsilon N)$. Hence the threshold for constructing an efficient formula is polynomially large, allowing an exponential number of computations on each input.

6.2. Quantum formulas. Now we derive the lower bound for generalized quantum formulas. Roychowdhury and Vatan [37] consider pure quantum formulas (recall these are quantum formulas which may not access multiply readable random bits). Their result is as follows.

FACT 6.5. *Every pure quantum formula computing a function f with a bounded error has length*

$$\Omega\left(\sum_i D(f_i)/\log D(f_i)\right)$$

for the Nečiporuk function $\sum_i D(f_i)$; see Fact 3.1 *and Definition* 3.4.

Furthermore in [37] it is shown that pure quantum formulas can be simulated efficiently by deterministic circuits.

Now we know from section 3.2 that the Boolean function MP with $O(n^2)$ inputs (the matrix product function) has fair probabilistic formulas of linear size $O(n^2)$, while the Nečiporuk bound is cubic (Theorems 3.11 and 3.12). Thus we get the following.

COROLLARY 6.6. *There is a Boolean function MP with N inputs, which can be computed by fair one-sided error formulas of length $O(N)$, while every pure quantum formula with a bounded error for MP has size $\Omega(N^{3/2}/\log N)$.*

We conclude that pure quantum formulas are not a proper generalization of classical formulas. A fair probabilistic formula can be simulated efficiently by a generalized quantum formula, on the other hand. We now derive a lower bound method for generalized quantum formulas. First we give again a lower bound in terms of one-way communication complexity, and then we show that the VC–Nečiporuk bound is a lower bound, too.

This implies with Theorem 3.9 that the maximal difference between the sizes of deterministic formulas and generalized bounded error quantum formulas provable with the Nečiporuk method is at most $O(\sqrt{n})$.

But first let us conclude the following corollary, which states that fair probabilistic formulas reading their random bits only once are sometimes inefficient.

COROLLARY 6.7. *The (standard) Nečiporuk function divided by* $\log n$ *is an asymptotical lower bound for the size for fair probabilistic formulas reading their random inputs only once.*

Proof. We have to show that pure quantum formulas can simulate these special probabilistic formulas. For each random input we use two qubits in the state $|00\rangle$. These are transformed into the state $|\Phi^+\rangle$ by a Hadamard gate. One of the qubits is never used again; then the other qubit has the density matrix of a random bit. Then the probabilistic formula can be simulated. For the simulation of gates unitary transformations on three qubits are used. These get the usual inputs of the gate simulated plus one empty qubit as input, which after the application of the gate carries the output. These gates are easily constructed unitarily. According to [4] each 3-qubits gate can be composed of $O(1)$ unitary gates on two qubits only. □

We will need the following observation [1].

FACT 6.8. *If the density matrix of two qubits in a circuit (with nonentangled inputs) is not the tensor product of their density matrices, then there is a gate so that both qubits are reachable on a path from that gate.*

Since the above situation is impossible in a formula, the inputs to a gate are never entangled.

The first lower bound is stated in terms of one-way communication complexity. It is interesting that actually randomized complexity suffices for a lower bound on quantum formulas.

THEOREM 6.9. *Let f be a Boolean function on n inputs and $y_1 \dots y_k$ a partition of the input variables in k blocks. Player Bob knows the inputs in y_i, and player Alice knows all other inputs. The randomized (private coin) one-way communication complexity of f (with a bounded error) under this input partition is called $R(f_i)$.*

Every generalized quantum formula for f with a bounded error has length

$$\Omega\left(\sum_i \frac{R(f_i)}{\log R(f_i)}\right).$$

Proof. For a given partition of the input we show how a generalized quantum formula F can be simulated in the k communication games, so that the randomized one-way communication in game i is bounded by a function of the number of leaves in a subtree F_i of F. F_i contains exactly the variables belonging to Bob as leaves, and its root is the root of F. Furthermore F_i contains all gates on paths from these leaves to the root. Note that the additional nonentangled mixed state which the formula may access is given to Alice.

F is a tree of fan-in 2 fan-out 1 superoperators (recall that superoperators are not necessarily reversible). "Wires" between the gates carry one qubit each. F_i is a formula that Bob wants to evaluate, the remaining parts of the formula F belong to Alice, and she can easily compute the density matrices for all qubits on any wire in her part of the formula by a classical computation, as well as the density matrices for the qubits crossing to Bob's formula F_i. Note that none of the qubits on wires crossing to F_i is entangled with another, so the state of these qubits is a probabilistic ensemble of pure nonentangled states. Hence Alice may fix a pure nonentangled state from this ensemble with a randomized choice.

In all communication games Bob evaluates the formula as far as possible without the help of Alice. By an argument as in other Nečiporuk methods (e.g., [7, 37] or the previous sections) it is sufficient to send a few bits from Alice to Bob to evaluate a path with the following property: All gates on the path have one input from Alice and one input from its predecessor, except for the first gate, which has one input from Alice and one (already known) input from Bob. With standard arguments the number of such paths is a lower bound on the number of leaves in the subformula; see section 3.1.

Hence we have to consider some path $g_1, \ldots, g_m$ in F, where g_1 has one input or a gate from Alice as predecessor and an input or gate from Bob as the other predecessor, and all gates g_i have the previous gate g_{i-1} and an input or gate from Alice's part of the formula as predecessors. The density matrix of Bob's input to g_1 is called ρ, and the density matrix of the other m inputs is called σ. The circuit computing σ works on different qubits than the circuit computing ρ.

Thus the density matrix of all inputs to the path is $\rho \otimes \sigma$; see Fact 6.8. The path maps $\rho \otimes \sigma$ with a superoperator T to a density matrix μ on one qubit; alternatively we may view σ as determining a superoperator T_σ on one qubit that has to be applied to ρ. Now Alice can compute this superoperator by herself, classically.

Bob knows ρ. Bob wants to know the state $T_\sigma \rho$. Since this operator works on a single qubit only, it can be described within a precision $1/poly(k)$ by a constant size matrix containing numbers of size $O(\log k)$ for any integer k. Thus Alice may communicate T_σ to Bob within this precision using $O(\log k)$ bits.

In this way Alice and Bob may evaluate the formula, and the error of the formula is changed only by $size_i/poly(k)$ compared to the error of the quantum formula, where $size_i$ denotes the number of gates in F_i. Thus choosing $k = poly(size_i)$ the communication is bounded $R(f_i) \leq O(size_i \log size_i)$. This implies $size_i \geq \Omega(R(f_i)/\log R(f_i))$. Summation over all i yields the theorem. □

The above construction loses a logarithmic factor, but in the combinatorial bounds we actually apply, we can avoid this, by using quantum communication and the programmable quantum gate from Fact 4.20.

THEOREM 6.10. *The VC–Nečiporuk function is an asymptotical lower bound for the length of generalized quantum formulas with a bounded error.*

The Nečiporuk function is an asymptotical lower bound for the length of generalized quantum Las Vegas formulas.

Proof. We proceed similar to the above construction, but Alice and Bob use quantum computers. Instead of communicating a superoperator in matrix form with some precision, we use the programmable quantum gate.

Alice and Bob cooperatively evaluate the formulas F_i in a communication game as before. As before, for certain paths Alice wants to help Bob to apply a superoperator T_σ on a state ρ of his. Using Fact 4.7 we can assume that this is a unitary operator on $O(1)$ qubits (one of them ρ, the others blank) followed by throwing away all but one of the qubits.

This time Alice sends to Bob the program corresponding to the unitary operation in T_σ. Bob feeds this program into the programmable quantum gate, which tries to apply the transformation, and if this is successful, the formula evaluation can continue after discarding the unnecessary qubits. This happens with probability $\Omega(1)$. If Alice could get some notification from Bob saying whether the gate has operated successfully, and if not, what kind of error occurred, then Alice could send him another program that both undoes the error and the previous operator and then makes another attempt to compute the desired operator.

Note that the error that resulted by an application of the programmable quantum gate is determined by the classical measurement outcome resulting in its application. Furthermore this error can be described by a unitary transformation itself. If the error function is E, the desired is unitary is U, and the state it has to be applied to is ρ, then Bob now holds $UE\rho E^\dagger U^\dagger$. Once Alice knows E (which is determined by Bob's measurement outcome), Alice can produce a program for $UE^\dagger U^\dagger$. If Bob applies this transformation successfully, they are done; otherwise, they can iterate. Note that only an expected number of $O(1)$ such iterations are necessary, and hence the expected quantum communication in this process is $O(1)$, too.

So the expected communication can be reduced to $O(size_i)$. But Alice needs some communication from Bob. Luckily this communication does not reveal any information about Bob's input: Bob's measurement outcomes are random numbers without correlation with his input.

So we consider the nonstandard one-way communication model from Lemma 5.13, in which Bob may talk to Alice but without revealing any information about his input. Using this model in the construction and letting Bob always ask explicitly for more programs reduces the communication in game i to $O(size_i)$ in the expected sense.

With Lemma 5.13 we get the lower bounds for a bounded error and Las Vegas communication. □

Now we can give a lower bound for ISA showing that even generalized quantum formulas compute the function not significantly more efficient than deterministic formulas.

Corollary 6.11. *Every generalized quantum formula which computes ISA with a bounded error has length $\Omega(n^2/\log n)$.*

Considering the matrix multiplication function MP, we get the following.

Corollary 6.12. *There is a function, which can be computed by a generalized quantum formula with a bounded error as well as by a fair probabilistic formula with a bounded error, with size $O(N)$. Every generalized quantum Las Vegas formula needs size $\Omega(N^{3/2})$ for this task. Hence there is a size gap of $\Omega(N^{1/2})$ between the Las Vegas formula length and the length of the bounded error formulas.*

Since the VC–Nečiporuk function is a lower bound for generalized quantum formulas, Theorem 3.9 implies that the maximal size gap between deterministic formulas and generalized quantum formulas with a bounded error provable by the (standard) Nečiporuk method is $O(\sqrt{n})$ for input length n. Such a gap actually already lies between generalized quantum Las Vegas formulas and fair probabilistic formulas with a bounded error.

7. Conclusions.

In this paper we have derived lower bounds for the sizes of probabilistic, nondeterministic, and quantum formulas. These lower bounds follow the general approach of reinterpreting the Nečiporuk bound in terms of one-way communication complexity. This is nontrivial in the case of quantum formulas, where we had to use a programmable quantum gate. Nevertheless we have obtained the same combinatorial lower bound for quantum and probabilistic formulas based on the VC dimension.

Using the lower bound methods we have derived a general $\sqrt{n}$ gap between the bounded error and Las Vegas formula size. Another result is a threshold phenomenon for the amount of nondeterminism needed to compute a function, which gives a near-quadratic size gap for a polynomial threshold on the number of nondeterministic bits.

To derive our results we needed lower bounds for one-way communication complexity. These results give gaps between two-round and one-way communication complexity in these models. Those gaps have been generalized to round hierarchies for larger number of rounds in [22] and [25] for the nondeterministic and the quantum cases, respectively. Furthermore we have shown that quantum Las Vegas one-way protocols for total functions are not much more efficient than deterministic one-way protocols. The lower bounds for quantum one-way communication complexity are also useful to give lower bounds for quantum automata and for establishing that only bounded error quantum finite automata can be exponentially smaller than deterministic finite automata [23]. A generalization of the VC-dimension bound on quantum one-way communication complexity is given in [24].

The following problems remain open:

1. Give a better separation between deterministic and probabilistic/quantum formula size (see [21] for a candidate function).
2. Separate the size complexities of generalized quantum and probabilistic formulas for some function.
3. Investigate the power of quantum formulas that can access an entangled state as an additional input, thus introducing entanglement into the model.
4. Separate quantum and probabilistic one-way communication complexity for some total function or show that both are related.
5. Prove superquadratic lower bounds for formulas over the basis of all fan-in 2 gates.

Acknowledgments. The author thanks Gregor Gramlich for a talk on the Nečiporuk method, which inspired this research, Georg Schnitger for stimulating discussions, and the anonymous referees for their detailed suggestions and pointing out an error in an earlier version.

REFERENCES

[1] D. Aharonov, A. Kitaev, and N. Nisan, *Quantum Circuits with Mixed States*, in Proceedings of the 30th Annual ACM Symposium on Theory of Computing, Dallas, TX, 1998, pp. 20–30.

[2] A. Ambainis, A. Nayak, A. Ta-Shma, and U. Vazirani, *Quantum dense coding and quantum finite automata*, J. ACM, 49 (2002), pp. 496–511. Earlier versions appeared in STOC '99 and FOCS '99 (authored by Nayak alone).

[3] L. Babai, N. Nisan, and M. Szegedy, *Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs*, J. Comput. System Sci., 45 (1992), pp. 204–232.

[4] A. Barenco, C. Bennett, R. Cleve, D. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter, *Elementary gates for quantum computation*, Phys. Rev. A, 52 (1995), pp. 3457–3467.

[5] C. H. Bennett and S. J. Wiesner, *Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states*, Phys. Rev. Lett., 69 (1992), pp. 2881–2884.

[6] R. B. Boppana, *Amplification of probabilistic Boolean formulas*, in Proceedings of the 26th IEEE Symposium on Foundations of Computer Science, Portland, OR, 1985, pp. 20–29.

[7] R. B. Boppana and M. Sipser, *The Complexity of Finite Functions*, in Handbook of Theoretical Computer Science A, Elsevier, New York, 1990.

[8] H. Buhrman and R. de Wolf, *Communication complexity lower bounds by polynomials*, in Proceedings of the 16th Annual IEEE Conference on Computational Complexity, Chicago, IL, 2001, pp. 120–130.

[9] R. Cleve and H. Buhrman, *Substituting quantum entanglement for communication*, Phys. Rev. A, 56 (1997), pp. 1201–1204.

[10] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley Ser. Telecomm., 1991.

[11] D. Deutsch, *Quantum computational networks*, Proc. R. Soc. Lond. A Ser. Math. Eng. Sci., 425 (1989), pp. 73–90.

[12] M. Dubiner and U. Zwick, *Amplification by read-once formulae*, SIAM J. Comput., 26 (1997), pp. 15–38.

[13] A. Einstein, B. Podolsky, and N. Rosen, *Can quantum-mechanical description of physical reality be considered complete?*, Phys. Rev., 47 (1935), pp. 777–780.

[14] J. Goldsmith, M. A. Levy, and M. Mundhenk, *Limited nondeterminism*, SIGACT News, 27 (1996), pp. 20–29.

[15] J. Håstad, *The shrinkage exponent of de Morgan formulas is* 2, SIAM J. Comput., 27 (1998), pp. 48–64.

[16] A.S. Holevo, *Some estimates on the information transmitted by quantum communication channels*, Prob. Inf. Transm., 9 (1973), pp. 177–183.

[17] J. Hromkovič and M. Sauerhoff, *Tradeoffs between nondeterminism and complexity for communication protocols and branching programs*, in Proceedings of the 17th Symposium on Theoretical Aspects of Computer Science, Lille, France, 2000, Lect. Notes Comput. Sci. 1770, Springer, New York, 2000, pp. 145–156.

[18] J. Hromkovič and G. Schnitger, *Nondeterministic communication with a limited number of advice bits*, SIAM J. Comput., 33 (2003), pp. 43–68.

[19] J. Hromkovič and G. Schnitger, *On the power of Las Vegas for one-way communication complexity, OBDDs, and finite automata*, Inform. Comput., 169 (2001), pp. 284–296.

[20] J. Katz and L. Trevisan, *On the efficiency of local decoding procedures for error-correcting codes*, in Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, Portland, OR, 2000, pp. 80–86.

[21] H. Klauck, *On the size of probabilistic formulae*, in Proceedings of the 8th International Symposium on Algorithms and Computation, Singapore, 1997, Lect. Notes Comput. Sci. 1350, Springer, 1997, pp. 243–252.

[22] H. Klauck, *Lower bounds for quantum communication complexity*, SIAM J. Comput., 37 (2007), pp. 20–46.

[23] H. Klauck, *On quantum and probabilistic communication: Las Vegas and one-way protocols*, in Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, Portland, OR, 2000, pp. 644–651.

[24] H. Klauck, *Quantum communication complexity*, in Proceedings of the Workshop on Boolean Functions and Applications at 27th ICALP, Geneva, Switzerland, 2000, pp. 241–252.

[25] H. Klauck, A. Nayak, A. Ta-Shma, and D. Zuckerman, *Interaction in quantum communication and the complexity of set disjointness*, in Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, Hersonissos, Greece, 2001, pp. 124–133.

[26] I. Kremer, N. Nisan, and D. Ron, *On randomized one-round communication complexity*, Comput. Complexity, 8 (1999), pp. 21–49.

[27] E. Kushilevitz and N. Nisan, *Communication Complexity*, Cambridge University Press, London, 1997.

[28] H. Lo and H. Chau, *Why quantum bit commitment and ideal quantum coin tossing are impossible*, Phys. D, 120 (1998), pp. 177–187.

[29] D. Mayers, *Unconditionally secure quantum bit commitment is impossible*, Phys. Rev. Lett., 78 (1997), pp. 3414–3417.

[30] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, London, 1995.

[31] A. Nayak, *Lower Bounds for Quantum Computation and Communication*, Ph.D. thesis, University of California, Berkeley, 1999.

[32] E. I. Nečiporuk, *A Boolean function*, Sov. Math. Dokl., 7 (1966), pp. 999–1000.

[33] I. Newman, *Private vs. common random bits in communication complexity*, Inform. Process. Lett., 39 (1991), pp. 67–71.

[34] M. A. Nielsen and I. Chuang, *Programmable quantum gate arrays*, Phys. Rev. Lett., 79 (1997), pp. 321–324.

[35] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, London, 2000.

[36] C. Papadimitriou and M. Sipser, *Communication complexity*, J. Comput. System Sci., 28, (1984), pp. 260–269.

[37] V. P. Roychowdhury and F. Vatan, *Quantum formulas: A lower bound and simulation*, SIAM J. Comput., 31 (2001), pp. 460–476.

[38] L. G. Valiant, *Short monotone formulae for the majority function*, J. Algorithms, 5 (1984), pp. 363–366.

[39] V. N. Vapnik and A. Y. Chervonenkis, *On the uniform convergence of relative frequencies of events to their probabilities*, Theory Probab. Appl., 16 (1971), pp. 264–280.

[40] A. C. Yao, *Some complexity questions related to distributed computing*, in Proceedings of the 11th Annual ACM Symposium on Theory of Computing, Atlanta, GA, 1979, pp. 209–213.

[41] A. C. Yao, *Quantum circuit complexity*, in Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science, Palo Alto, CA, 1993, pp. 352–361.

[42] U. Zwick, *Boolean Circuit Complexity*, lecture notes, Tel Aviv University, 1995.

SIAM J. Comput.
Vol. 37, No. 2, pp. 584–610

OPTIMAL EXPECTED-CASE PLANAR POINT LOCATION[*]

SUNIL ARYA[†], THEOCHARIS MALAMATOS[‡], DAVID M. MOUNT[§], AND KA CHUN WONG[†]

Abstract. Point location is the problem of preprocessing a planar polygonal subdivision S of size n into a data structure in order to determine efficiently the cell of the subdivision that contains a given query point. We consider this problem from the perspective of expected query time. We are given the probabilities p_z that the query point lies within each cell $z \in S$. The entropy H of the resulting discrete probability distribution is the dominant term in the lower bound on the expected-case query time. We show that it is possible to achieve query time $H + O(\sqrt{H} + 1)$ with space $O(n)$, which is optimal up to lower order terms in the query time. We extend this result to subdivisions with convex cells, assuming a uniform query distribution within each cell. In order to achieve space efficiency, we introduce the concept of entropy-preserving cuttings.

Key words. point location, polygonal subdivision, expected-case complexity, entropy, trapezoidal maps, entropy-preserving cuttings

AMS subject classifications. 68P05, 68P10, 68W40

DOI. 10.1137/S0097539704446724

1. Introduction.

A planar straight-line graph defines a subdivision of the plane into (possibly unbounded) polygonal regions called *cells*. Planar point location is the problem of preprocessing such a polygonal subdivision S so that, given any query point q, the cell containing q can be computed efficiently. Throughout, we let n denote the total *size* of S, defined to be the total number of vertices, edges, and faces of S.

The point-location problem has a considerable history. The first asymptotically worst-case optimal result in the area was Kirkpatrick's elegant method based on hierarchical triangulations [19], which supported query processing in $O(\log n)$ time using $O(n)$ space. This was followed by a number of other optimal methods with better practical performance including the layered directed acyclic graph of Edelsbrunner, Guibas, and Stolfi [13], searching in similar lists by Cole [8], the method based on persistent search trees by Sarnak and Tarjan [25], and the randomized incremental algorithms of Mulmuley [23] and Seidel [26]. The important question of determining the exact constant factor in query time was raised in work by Goodrich, Orletsky, and Ramaiyer [16] and was solved subsequently by Seidel and Adamy [27], who showed

[*] Received by the editors December 16, 2004; accepted for publication (in revised form) July 17, 2006; published electronically June 12, 2007. Preliminary results appeared in the papers *Nearly optimal expected-case planar point location*, in Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science, 2000, pp. 208–218 and *Entropy-preserving cuttings and space-efficient planar point location*, in Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms, 2001, pp. 256–261. The work of the first, second, and fourth authors was supported in part by the Research Grants Council, Hong Kong, China (HKUST6229/00E and HKUST6080/01E).
http://www.siam.org/journals/sicomp/37-2/44672.html

[†] Department of Computer Science, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong (arya@cs.ust.hk, odan@cs.ust.hk).

[‡] Max Planck Institut für Informatik, Saarbrücken, Germany (tmalamat@mpi-sb.mpg.de). Part of this research was conducted while the author was at the Hong Kong University of Science and Technology.

[§] Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742 (mount@cs.umd.edu). This author's work was supported by the National Science Foundation (CCR–0098151 and CCF–0635099).

that point-location queries can be answered in $\log_2 n + 2\sqrt{\log_2 n} + o(\sqrt{\log n})$ time and $O(n)$ space. They also provided a nearly matching lower bound.

Adamy and Seidel's results were based on a model of computation, called the *trapezoidal search graph model*, in which the result of the query is based entirely on binary tests called *primitive comparisons*. There are two types of primitive comparisons. The first determines whether the query point q lies to the left or right of a vertical line passing through a vertex of the subdivision. (See Figure 1(a).) The other determines whether the query point lies above or below an edge of the subdivision. This latter comparison is performed only after we have already determined that the x-coordinates of the point lie between the x-coordinates of the endpoints of the edge. (See Figure 1(b).) Note that both comparisons can be expressed as standard *orientation tests* [11] (in 1 or 2 dimensions). Orientation tests form the basis of many algorithms in discrete computational geometry [14]. In particular, all of the point-location algorithms mentioned above are easily formulated in this model. Our main result (Theorem 1 below) assumes this same model of computation.

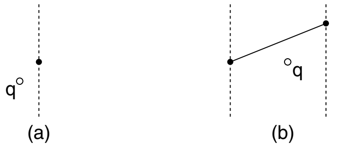


FIG. 1. *Primitive comparisons.*

All the previous results on point location were considered in the context of worst-case query times. In many applications, point-location queries tend to be clustered in regions of greater interest. This raises the question of whether it is possible to use knowledge of the query distribution to achieve better query times in the expected case. We model this by assuming that, for each cell $z \in S$, we are given the probability p_z that a query point lies in z. We call the result a *weighted subdivision.* Unless otherwise stated we make no assumptions about the probability distribution within each cell. To avoid dealing with many special cases, we assume that the probability that the query point lies on an edge or vertex of the subdivision is zero, but this restriction can be overcome, for example, by treating edges and vertices of nonzero probability as cells that have infinitesimal width or extent.

An important concept in characterizing the complexity of the search is the *entropy* of S, denoted throughout as H:

$$entropy(S) \ = \ H \ = \ \sum_{z \in S} p_z \log(1/p_z).$$

(Unless otherwise stated all logarithms are taken in base 2.) It is well known that entropy is maximized when all of the cells have equal probability [10], in which case $H = \log n_2$, where n_2 denotes the number of cells (faces of dimension 2) of S. Conversely, entropy decreases as the disparity among the probabilities increases. Note that entropy may be arbitrarily close to 0. Unlike n, when stating asymptotic bounds, we cannot assume that H is larger than a fixed constant. For this reason, throughout the paper we will follow the convention that the expression "f is $O(g)$" means that there exists a constant c (independent of n and H) such that $f \leq c \cdot g$. Furthermore, we assume throughout that $n \geq n_0$ for some constant n_0.

For the 1-dimensional restriction of this problem, a classical result due to Shannon implies that the expected number of binary comparisons needed to answer such queries is at least as large as the entropy of the probability distribution [20, 28], and clearly this lower bound applies to the 2-dimensional case as well. Mehlhorn [22] showed that it is possible to build a binary search tree whose expected search time is at most $H + 2$. The related problem of computing the binary search tree that minimizes the expected search time is considered in [17, 20].

The idea of using the entropy of the query distribution as the basis for an analysis for geometric data structures is a recent development in computational geometry. Arya and Fu [2] first applied this approach to analyzing the complexity of approximate nearest neighbor queries. Arya et al. [1] then applied this to point location in subdivisions having convex cells. They assumed that the x- and y-coordinates of the query point were chosen independently from some probability distribution. They showed that $O(H + 1)$ expected query time was achievable, where the multiplicative constant factor was a function of the amount of space used. Arya, Malamatos, and Mount [5] presented a simple and practical randomized algorithm that answers queries in $O(H + 1)$ expected time with $O(n)$ space, and Iacono [18] presented a similar deterministic method achieving the same bounds.

In the spirit of prior work on optimal search structures [16, 22, 27], a fundamental question is whether it is possible to achieve the expected query time of H including only additive lower order terms. In our earlier work on this problem [3,4] we presented such data structures, but the space was linear only in special cases, for example, if the cells are axis-parallel rectangles. Otherwise, the space requirements were superlinear, ranging from $O(n \log^* n)$ up to $O(n^{1+\epsilon})$ depending on the nature of the subdivision and assumptions about the probability distribution.

All of the existing solutions fall short of the goal of producing a point-location structure of linear space whose expected query time matches the information-theoretic lower bound of H (up to lower order terms) and which makes no restrictions on the probability distribution within each cell. In this paper we present such a solution. Here is our main result.

Theorem 1. *Consider a polygonal subdivision S of size n consisting of cells of constant combinatorial complexity and a query distribution presented as a weight assignment to the cells of S. In time $O(n \log n)$ it is possible to construct a search structure of space $O(n)$ that answers point-location queries (in the trapezoidal search graph model) in expected time $H + O(\sqrt{H} + 1)$, where $H = entropy(S)$.*

Recall that H may generally be arbitrarily close to 0, which is why the extra "+1" is added to the asymptotic term. Due to our reliance on geometric cuttings of line segments in the plane [7, 16], our construction is randomized, and so the $O(n \log n)$ construction time holds in expectation. Otherwise, our construction runs in $O(n \log n)$ time and is deterministic. Throughout, we make the usual general-position assumption that no two vertices have the same x-coordinate, and thus no edge is vertical. This assumption can be overcome, for example, by standard perturbation methods [14].

The requirement that cells have constant complexity applies only to cells of nonzero probability, since cells of zero probability can be triangulated without affecting the entropy of the subdivision. (This applies to the unbounded external cell of the subdivision as well.) If no assumptions are made about the query distribution, then the assumption that cells have constant cell complexity seems to be critical. In the next section, for example, we show that if the query distribution is arbitrary, then even determining whether a query point lies within a single n-sided convex polygon requires expected time $\Omega(\log n)$, irrespective of entropy. Nonetheless, we show (in

Theorem 2 below) that if we are given a *convex subdivision* (that is, one whose faces are convex) and assume that the query distribution within each cell is *uniform*, then it is possible to answer point-location queries just as efficiently even if the cells have an arbitrary number of sides. In order to handle convex cells it will be necessary to refine the subdivision through the insertion of new edges. For this reason we define the *extended trapezoidal search graph model* to include primitive comparisons involving line segments that are not necessarily part of the original subdivision, but that join two vertices of the original subdivision.

THEOREM 2. *The complexity bounds of Theorem* 1 *apply as well (in the extended trapezoidal search graph model) if S is a weighted convex planar polygonal subdivision such that the query distribution within each cell is uniform.*

In order to provide a formal definition of expected query time in the (standard or extended) trapezoidal search graph model, we begin with a brief discussion of binary space partition trees. Observe that any point-location algorithm that is based on binary comparisons can be modeled abstractly as a decision-tree structure called a *binary space partition* (BSP) [11]. (The search structure that we present is not a proper binary tree, since it allows sharing of substructures, but this affects only the space requirements and not the query time.) For our purposes, a BSP is a rooted binary tree in which each internal node is associated with a line. This line subdivides the plane into two half-planes, one open and one closed, which are then associated with the node's two children. (Since we assume that query points do not fall on edges, the question of which is open and which is closed is not significant.) Each node of a BSP is implicitly associated with a (possibly unbounded) convex polygon, called its *region*, which is the intersection of the half-planes corresponding to the path from the root to this node.

Given a BSP, point-location queries are answered by performing a simple descent in the tree. At each internal node we visit the child corresponding to the half-plane that contains the query point until arriving at a leaf. It is easy to see that the query point lies within the regions associated with each of the nodes along the search path. It follows that the BSP correctly solves the point-location problem if and only if the region associated with every leaf of the tree lies entirely within a single cell of the subdivision. (If the region were to overlap two or more cells, we could not unambiguously determine which of these cells contains the query point.) When the search arrives at a leaf, the associated cell is returned as the answer. Given a query distribution, each leaf of the BSP is associated with the probability that the query point lies within this leaf's region. The *weighted external path length* of a BSP is the weighted average of the depths of all its leaves, where the weight is this probability [20]. We define the *expected query time* of a BSP to be this weighted external path length. An example is shown in Figure 2, where (a) shows the original subdivision, (b) shows the binary space partition induced by three lines L_1, L_2, and L_3 and the associated probabilities with each region, and (c) shows the associated tree. In this case the weighted external path length is $1 \cdot p_1 + 2 \cdot p_2 + 3(p_3 + p_4)$.

The rest of the paper is organized as follows. The next section presents mathematical preliminaries and provides an overview of our approach. In section 3 we present an algorithm for answering point-location queries that is optimal in expected time (up to lower order terms) but suboptimal in space. It answers queries in expected time $H + O(\sqrt{H} + 1)$ and space $O(n^{1+\epsilon})$ for any $\epsilon > 0$. In section 4 we show how to reduce this to $O(n)$ space. We first introduce the notion of entropy-preserving cuttings, and in section 4.1 and section 4.2 we show how to apply this concept to complete the space bound of Theorem 1. Finally, in section 5, we show that these

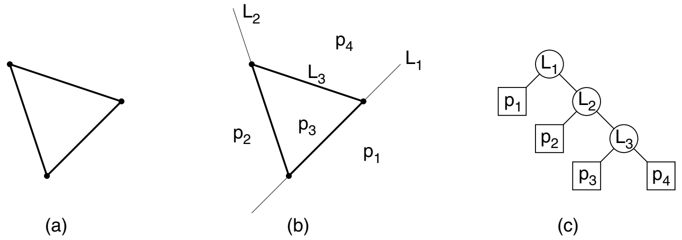


FIG. 2. *BSP and the associated search tree.*

methods can be generalized to convex subdivisions with uniform query distributions within each cell in order to prove Theorem 2.

2. Preliminaries. Consider a weighted subdivision S. Viewing S as a planar graph, let n denote the total numbers of its vertices, edges, and faces (cells), respectively. The (unbounded) external face is also considered a cell. A well-known consequence of Euler's formula is that if this graph is connected, then n is asymptotically bounded by the number of edges of S (see, e.g., [11]). Since the number of edges is clearly a lower bound on the space complexity of any point-location structure, any $O(n)$ space structure is asymptotically optimal with respect to space.

2.1. On subdivisions of unbounded cell complexity. Throughout much of the paper we will concentrate on the case where the cells of S are bounded by a constant number of sides. We will show here why this assumption seems to be critical in the context of achieving query-time bounds based on entropy. Note that this assumption is not required for worst-case optimal planar point location, since it is possible to refine any planar polygonal subdivision into one whose cells have a constant number of sides while increasing the size of the subdivision by just a constant factor. However, we show that if cells have unbounded complexity, even if they are convex, there exist query distributions such that any search structure based on point-line comparisons performs arbitrarily worse than the entropy bound in the expected case.

LEMMA 1. *Given any convex polygon Z with n sides, there exists a discrete query probability distribution such that the probability that a query point lies within Z is $1/2$, and the expected number of point-line comparisons needed to determine whether a point lies within Z is $\Omega(\log n)$.*

Proof. The probability distribution is defined as follows. Let the vertices of Z be $\{v_1, \ldots, v_n\}$. For each vertex v_i we consider two points a_i and b_i placed very close to v_i. The point a_i lies just inside of Z, and b_i lies just outside of Z. The points a_i and b_i all carry a query probability of $1/(2n)$. (See Figure 3.) Observe that the probabilities sum to 1.

Now let Ψ be any BSP that correctly determines membership in Z. Clearly for Ψ to be correct, for each vertex v_i there must be some node of Ψ whose associated line stabs the line segment $\overline{a_i b_i}$, since otherwise both a_i and b_i would lie in the same leaf region, implying that we cannot distinguish between inside and outside in this case. Because Z is convex, we can place the points a_i and b_i sufficiently close to each vertex so that any line can stab at most two such segments. Now consider any node of Ψ. In order to minimize the expected search time, the best that we can hope to accomplish is that the remaining probability in the region is evenly split between the left and right children, implying that, other than the at most two vertices whose

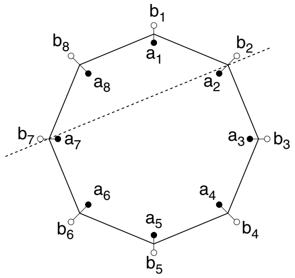


FIG. 3. *The proof of Lemma* 1 *and a stabbing line.*

segments were stabbed, half of the remaining vertices lie on one side of the line and half on the other. It follows easily that $\Omega(\log n)$ such comparisons are needed along any search path of the optimum BSP. □

A convex polygon defines a trivial subdivision consisting of two cells (inside and outside). Irrespective of n, the entropy of the subdivision described in Lemma 1 is easily seen to be 1. Since the query time grows as $\Omega(\log n)$, it is not possible to bound the expected query time purely as a function of entropy. Thus we have the following theorem.

THEOREM 3. *If no restrictions are placed on cell complexity or query distribution, then no search structure based on point-line comparisons can guarantee an expected-case query time that is bounded purely by a function of entropy, even for convex subdivisions.*

2.2. Conditioning the subdivision. If the cells of the subdivision all have constant combinatorial complexity, then we claim that it is possible to condition the problem to bring it into a simpler form without adversely affecting the expected-case query time. Many point-location data structures assume that the subdivision is presented in some canonical subdivision, e.g., a triangulation [19], a monotone subdivision [13], or a trapezoidal map [23,26]. If the cells of S have constant combinatorial complexity, then any of these canonical subdivisions can be realized by refining each cell into at most a constant number of subcells. The most convenient canonical structure for our purposes is a *trapezoidal map.* This is a planar subdivision in which each cell is a trapezoid with vertical parallel sides. These are trapezoids in a general sense and may be unbounded or degenerate to triangles. (To avoid the complexities of unbounded cells, it is common to enclose the entire subdivision in a large bounding rectangle, which contains all the query points.) Any polygonal subdivision can be converted into a trapezoidal map by adding two vertical segments between each vertex and the edges lying immediately above and below it. (See Figure 4.) This can be done in $O(n \log n)$ time either by a straightforward modification of plane sweep [6,11] or through a simple randomized incremental construction [23,26]. (Recall that by our general-position assumption, no segment of the original subdivision is vertical.)

In this section we show that if the initial subdivision has cells of constant complexity, then it is possible to condition the input without significantly affecting the expected-case query time so that it is a trapezoidal map. (The method can be applied to produce any of the other canonical subdivision forms, but this is the one that will be most relevant to our construction.) Before giving a formal statement of the result, we first present a few definitions. Given a planar subdivision S, a *refinement* S^* is a planar subdivision such that each cell of S^* is contained within some cell of

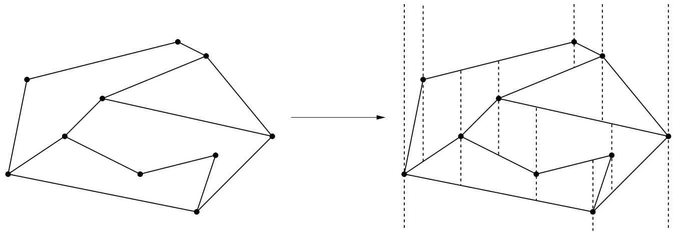


FIG. 4. *A subdivision (left) and its trapezoidal map (right).*

S. Given S and S^*, for each cell $z \in S$, we let $\mathcal{F}_z$ denote the subset of cells of S^* that are contained within z, called its *fragments*. Let $f_z = |\mathcal{F}_z|$ be the number of fragments. We say that S^* has *fragmentation* f if $f_z \leq f$ for all cells z. Clearly answering point-location queries for S can be reduced to answering queries of S^*.

Given a function $g : \mathbb{R}^+ \to \mathbb{R}^+$, we say that a point-location structure Ψ is *g-efficient* for a weighted subdivision S with entropy H if Ψ answers point location queries for S in expected time at most $H + g(H)$. For example, Theorem 1 asserts that it is possible to construct an $O(\sqrt{H} + 1)$-efficient point-location structure. Since the function g is designed to capture the lower order terms of the query-time complexity, it should grow at an asymptotic rate that is less than linear. To make this more formal, we say that a positive function g is *admissible* if for all reals $x, y \geq 0$

- g is subadditive, that is, $g(x + y) \leq g(x) + g(y)$, and
- $g(x)$ is $O(x + 1)$.

It is easy to see that, for any positive constant c, the function $g(H) = c \cdot (\sqrt{H} + 1)$ is admissible.

An important observation about the point-location construction described in Theorem 1 is that it is only given the probability p_z that a query point lies within each cell and knows nothing of the probability distribution within each cell. We say that such a construction is *distribution-oblivious*. It follows that any structure produced by such an algorithm must satisfy its expected-case query-time bound for any choice of the probability distribution within each cell, provided of course that the probability that a point lies within cell z is indeed p_z. This issue arises because in the process of computing the point-location structure we compute a refinement of the original subdivision. We do not know what the query distribution is within the cells of the refined subdivision, and so we cannot compute its expected query time. However, we know that entropy is maximized when all cells have the same probability. Thus, we invent a query distribution by splitting the probabilities evenly among the fragments of each refined cell. We then build an efficient structure for the resulting weighted subdivision using any distribution-oblivious construction. The following result shows formally that this strategy leads to an efficient solution to the original problem, irrespective of the actual query distribution.

LEMMA 2. *Consider a subdivision S and a query distribution on S. Let H denote its entropy. Let S^* be any refinement of S of fragmentation $O(1)$. In $O(n)$ time it is possible to assign nonnegative weights to the cells of S^*, thus producing a weighted subdivision $\widehat{S}$ with the following property. Let Ψ be any g-efficient point-location structure for $\widehat{S}$ produced by a distribution-oblivious construction for an admissible function g. Then Ψ is an $O(g + 1)$-efficient point-location structure for the original subdivision S.*

Before presenting the proof, we show how to apply this result to achieve the desired conditioning. Consider any weighted subdivision S of constant cell complexity and of total size n. In $O(n \log n)$ time we convert this subdivision into a trapezoidal map $\widehat{S}$ through a refinement of fragmentation $O(1)$. Clearly the resulting refinement also has size $O(n)$. Now, we apply the above result to assign weights to this trapezoidal map. Assuming that Theorem 1 holds for trapezoidal maps, it follows that in $O(n \log n)$ time we can construct an $O(\sqrt{H}+1)$-efficient point-location structure Ψ for $\widehat{S}$. By the above result, Ψ can be used as an $O(\sqrt{H}+1)$-efficient point-location structure for the original subdivision S, albeit with a higher constant factor hidden by the "O"-notation.

We devote the remainder of this section to proving Lemma 2. Let S denote the original subdivision of entropy H. Let S^* be a refinement of S of fragmentation f. For each cell z of S, recall that p_z is the probability that the query point lies within z. Also recall that $\mathcal{F}_z$ denotes the set of f_z fragments into which z has been subdivided. For each fragment $y \in \mathcal{F}_z$ there is some probability p_y that the query point lies within y, but the algorithm does not know this probability. Clearly $\sum_{y \in \mathcal{F}_z} p_y = p_z$. Let $H^* = \sum_{z \in S} \sum_{y \in \mathcal{F}_z} p_y \log(1/p_y)$ denote the entropy of this (unknown) distribution.

For the purposes of the proof we need to assign probabilities to the fragments. Since we know that entropy is maximized when probabilities are distributed as evenly as possible, let us split the weight evenly among the fragments by setting $w_y = p_z/f_z$ for each fragment $y \in \mathcal{F}_z$. Since $p_y \le p_z$, and since S^* has fragmentation f, we have $w_y \ge p_y/f$. Clearly $\sum_{y \in \mathcal{F}_z} w_y = p_z$. Let $\widehat{S}$ denote the resulting weighted subdivision, and let

$$\widehat{H} = \sum_{z \in S} \sum_{y \in \mathcal{F}_z} w_y \log \frac{1}{w_y}$$

denote its entropy based on this weight assignment. The assignment can easily be computed in $O(n)$ time. The following lemma asserts that the entropies of all these subdivisions are related to each other, up to an additive term of $\log f$.

Lemma 3. *Given the weighted subdivisions S, S^*, and $\widehat{S}$ defined above, their respective entropies satisfy*

$$H \;\le\; H^* \;\le\; \widehat{H} \;\le\; H + \log f.$$

Proof. To prove the first inequality we observe that if y is a fragment of z, then $p_y \le p_z$ and thus

$$H \;=\; \sum_{z \in S} p_z \log \frac{1}{p_z} \;=\; \sum_{z \in S} \sum_{y \in \mathcal{F}_z} p_y \log \frac{1}{p_z} \;\le\; \sum_{z \in S} \sum_{y \in \mathcal{F}_z} p_y \log \frac{1}{p_y} \;=\; H^*.$$

The second inequality is an immediate consequence of the fact that entropy is maximized when the probabilities (weights) are equal to each other [10], which is clearly the case for the weight assignment defining $\widehat{H}$. Finally, to prove the last inequality we use the facts that $f_z = |\mathcal{F}_z|$, $\sum_z p_z = 1$, and $f_z \le f$ to obtain

$$\begin{aligned}\widehat{H} = \sum_{z \in S} \sum_{y \in \mathcal{F}_z} w_y \log \frac{1}{w_y} \;&=\; \sum_{z \in S} \sum_{y \in \mathcal{F}_z} \frac{p_z}{f_z} \log \frac{f_z}{p_z} \;=\; \sum_{z \in S} p_z \left(\log \frac{1}{p_z} + \log f_z \right) \\ &\le \left(\sum_{z \in S} p_z \log \frac{1}{p_z} \right) + \log f \;=\; H + \log f. \qquad \square\end{aligned}$$

Returning now to the proof of Lemma 2, recall that $\widehat{S}$ is our uniformly weighted refined subdivision. Let Ψ be a g-efficient point-location structure for $\widehat{S}$, which results from a distribution-oblivious construction. Let $\widehat{E}_\Psi$ denote the expected query time for Ψ assuming the weight assignment w_y given above for each cell y of $\widehat{S}$. (Technically, $\widehat{E}_\Psi$ can only be defined relative to a particular probability distribution such that the probability that a query point lies within y is w_y. However, given that the construction is distribution-oblivious, we know that for any choice of such a probability distribution we will have $\widehat{E}_\Psi \leq \widehat{H} + g(\widehat{H})$. Since this is the only assumption we will make about $\widehat{E}_\Psi$ we will tolerate this abuse.)

Because Ψ can be viewed abstractly as a BSP and by the remarks made in section 2.1 on the correctness of BSPs for point location, it follows that the region associated with each leaf cell of Ψ lies within some cell y of $\widehat{S}$. Thus, each fragment y is further decomposed by Ψ into subfragments. Let $\mathcal{F}_y$ denote the subfragments of y, each associated with a leaf of Ψ. (See Figure 5.)

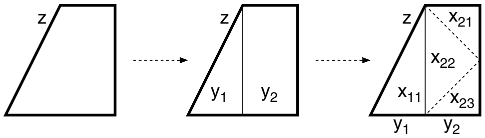


FIG. 5. *Fragments ($\mathcal{F}_z = \{y_1, y_2\}$) and subfragments ($\mathcal{F}_{y_2} = \{x_{21}, x_{22}, x_{23}\}$).*

For each $x \in \mathcal{F}_y$, let p_x denote the (unknown) probability that the query point lies within this subfragment, and let d_x denote its search depth in Ψ, that is, the number of primitive comparisons needed to provide an answer for any query point in x. Since subfragments may have different search depths, we then define the (true) expected search depth for a fragment y of S^* to be

$$D_y = \frac{1}{p_y} \sum_{x \in \mathcal{F}_y} p_x d_x.$$

It follows easily that if we apply Ψ as a point-location structure for the original subdivision S, the expected search time, denoted E_Ψ, is given by summing up the contributions from all the subfragments:

$$E_\Psi \;=\; \sum_{z \in S} \sum_{y \in \mathcal{F}_z} \sum_{x \in \mathcal{F}_y} p_x d_x \;=\; \sum_{z \in S} \sum_{y \in \mathcal{F}_z} p_y D_y.$$

The values of the p_y's are not known to us, and so we cannot compute D_y. But because Ψ is constructed by a distribution-oblivious algorithm, we know that the upper bound on the expected query time for $\widehat{S}$ holds irrespective of the choice of probability distribution within each of the fragments. We define such a probability distribution for $\widehat{S}$ by allocating weights among the subfragments in exactly the same proportions as their true values. Of course, this is done subject to the constraint that they sum to w_y. We also define the expected depth analogously:

$$w_x \;=\; \frac{p_x}{p_y} w_y \qquad \text{and} \qquad \widehat{D}_y \;=\; \frac{1}{w_y} \sum_{x \in \mathcal{F}_y} w_x d_x.$$

We observe that $\widehat{D}_y$ is equal to its counterpart in the true distribution:

$$\widehat{D}_y \;=\; \frac{1}{w_y} \sum_{x \in \mathcal{F}_y} w_x d_x \;=\; \frac{1}{w_y} \sum_{x \in \mathcal{F}_y} \frac{p_x}{p_y} w_y d_x \;=\; \frac{1}{p_y} \sum_{x \in \mathcal{F}_y} p_x d_x \;=\; D_y.$$

The expected search time $\widehat{E}_\Psi$ under our constructed distribution is

$$\widehat{E}_\Psi \;=\; \sum_{z\in S}\sum_{y\in\mathcal{F}_z}\sum_{x\in\mathcal{F}_y} w_x d_x \;=\; \sum_{z\in S}\sum_{y\in\mathcal{F}_z} w_y \widehat{D}_y.$$

By the obliviousness of Ψ's construction and g-efficiency we have

$$\begin{aligned} g(\widehat{H}) \geq \widehat{E}_\Psi - \widehat{H} \;&=\; \sum_{z\in S}\sum_{y\in\mathcal{F}_z} w_y\widehat{D}_y - \sum_{z\in S}\sum_{y\in\mathcal{F}_z} w_y \log\frac{1}{w_y} \\ &= \sum_{z\in S}\sum_{y\in\mathcal{F}_z} w_y\left(\widehat{D}_y - \log\frac{1}{w_y}\right). \end{aligned}$$

From the observation made prior to Lemma 3 that $w_y \geq p_y/f$ and the fact that the probabilities sum to 1, we have

$$\begin{aligned} g(\widehat{H}) &\geq \sum_{z\in S}\sum_{y\in\mathcal{F}_z} \frac{p_y}{f}\left(\widehat{D}_y - \log\frac{f}{p_y}\right) = \frac{1}{f}\sum_{z\in S}\sum_{y\in\mathcal{F}_z} p_y\left(\widehat{D}_y - \log\frac{1}{p_y} - \log f\right) \\ &= \frac{1}{f}\left(\left(\sum_{z\in S}\sum_{y\in\mathcal{F}_z} p_y\left(\widehat{D}_y - \log\frac{1}{p_y}\right)\right) - \log f\right). \end{aligned}$$

Next, we multiply both sides by f and add $\log f$, and then apply the substitution $\widehat{D}_y = D_y$ and the definitions of E_Ψ and H^* to obtain

$$f\cdot g(\widehat{H}) + \log f \;\geq\; \sum_{z\in S}\sum_{y\in\mathcal{F}_z} p_y\left(D_y - \log\frac{1}{p_y}\right) \;=\; E_\Psi - H^*.$$

Now, from Lemma 3 we know that $H^* \leq \widehat{H} \leq H + \log f$. Since g is admissible, it is also subadditive. Combining this with the fact that the fragmentation f is a constant, we obtain

$$\begin{aligned} E_\Psi - H \leq E_\Psi - H^* + \log f \;&\leq\; f\cdot g(H+\log f) + 2\log f \\ &\leq f(g(H) + g(\log f)) + 2\log f \qquad \text{(by subadditivity)} \\ &\leq f(g(H) + O(\log f + 1)) + 2\log f \qquad \text{(by admissibility)} \\ &= O(g(H)+1). \end{aligned}$$

This implies that Ψ is $O(g+1)$-efficient as a point-location structure for S, and so completes the proof of Lemma 2.

In conclusion, we can condition our input subdivision into a trapezoidal map so that the impact of this conditioning on the expected-case query time is to increase the constant factor of the lower order terms. This conditioning has a nice side benefit. Recall from the introduction that the entropy H of the input subdivision may generally be arbitrarily close to zero, but it does not make sense to talk about query times that are smaller than 1. As mentioned after the statement of Theorem 1, we therefore suffer the notational inconvenience of carrying an extra term of "$+1$" in all our complexity bounds. We claim that we can avoid this inconvenience henceforth because the entropy $\widehat{H}$ of the trapezoidal map is at least 1. To see this, observe that every bounded cell of the initial subdivision has at least three sides and thus will be subdivided by a vertical line into at least two trapezoids. Our construction distributes the probability evenly among the fragments of a cell, and so it follows that for all fragments z in the final subdivision we have $w_z \leq 1/2$. Therefore, because the sum of fragment weights is 1, the entropy of the resulting trapezoidal map is

$$\widehat{H} \;=\; \sum_{z \in S} \sum_{y \in \mathcal{F}_z} w_y \log(1/w_y) \;\geq\; \sum_{z \in S} \sum_{y \in \mathcal{F}_z} w_y \;=\; 1,$$

as desired.

2.3. Overview of our methods. Before presenting our algorithms we provide an overview of our methods. Our point-location data structure is based on many of the same methods used in the construction of worst-case efficient point-location structures, particularly the methods given by Preparata [24] and Seidel and Adamy [27]. Establishing efficient expected query time involves considerably different techniques from worst-case query time. As mentioned above, a point-location data structure that is based on linear comparisons can be viewed abstractly as a BSP. What properties must the associated partition tree possess in order to answer point-location queries efficiently on average? Observe that since the expected query time is the tree's weighted external path length, the contribution of each leaf to the expected query time is its tree depth times its probability. The probability that the query point lies in the region associated with a leaf is not known exactly, since we are not given the query distribution within each cell. Our strategy will be to construct a tree in which the depth of any leaf generated from a cell $z \in S$ is close to $\log(1/p_z)$.

As we shall see this will lead to a method, which while optimal with respect to expected-case query time, is not optimal with respect to space. In particular our best upper bound on space is superlinear in n. (See Theorem 4 below.) To reduce the storage further we employ a common strategy in computational geometry, called cuttings. Given a subdivision S with m edges, and a parameter $r \geq 1$, a $(1/r)$-*cutting* [7] is a partition of the plane into $O(r)$ trapezoids such that the interior of each trapezoid is intersected by at most m/r edges of S. If numeric weights are assigned to the edges, then this can be generalized to a *weighted* $(1/r)$-*cutting*, where now the total weight of edges intersecting any trapezoid is at most W/r, where W is the total weight of all the edges. Goodrich, Orletsky, and Ramaiyer [16] and later Seidel and Adamy [27] applied cuttings in a divide-and-conquer manner to produce the most space-efficient data structure.

Cuttings cannot be applied directly in our case, however, since the partitioning process may refine the subdivision in a way that significantly increases its entropy, and this increases the expected query time. An important contribution of this paper is the notion of an *entropy-preserving cutting*, which additionally ensures that the entropy of the subdivision is increased by at most an additive constant. This will be presented in section 4. Our approach will be to apply entropy-preserving cuttings to build a two-level search structure. For the first level we construct an entropy-preserving cutting of an appropriately chosen size and build the initial point-location structure described in section 3 for the cutting. This achieves good expected-case query time but leaves a number of regions to search. We show that the probability that the query point lies within any of these remaining regions is so small that a relatively sloppy worst-case optimal point-location algorithm suffices to achieve our desired results.

Extending these results to convex subdivisions with uniform query distributions involves a simple refinement step. Each convex cell is triangulated by a process that extracts a triangle whose area is a constant fraction of the total area, and then recurses on each of the three resulting fragments. We show that if the query distribution is uniform, then the increase in entropy in the resulting refined subdivision is at most an additive constant.

3. Initial solution.

In this section we present an algorithm for answering point-location queries that is optimal in expected time but suboptimal in space. Recall that from the results of section 2 we may assume that the subdivision S is presented as a weighted trapezoidal map of nonvertical segments and its entropy H is at least 1.

THEOREM 4. *Given a trapezoidal map S of size n, together with probabilities p_z that a query point lies within each cell z, we can build a data structure that answers point-location queries in expected query time*

$$H + 2\sqrt{2H} + \frac{1}{2}\log H + O(1),$$

where H is the entropy of S. The space for the data structure is

$$O\left(n2^{\sqrt{2H}}\frac{1}{\sqrt{H}}\log n\right).$$

Other than the probabilities p_z we make no assumptions about the query probability distribution within each cell. Recall that $1 \leq H \leq \log n$. It is easy to verify that $2^{\sqrt{2H}}/\sqrt{H}$ increases monotonically for $H \geq 1/(\sqrt{2}\ln 2)^2 \approx 1.04$, and so the space is maximized when $H = \log n$. Thus, the space is at most $O(n2^{\sqrt{2\log n}}\sqrt{\log n})$, which is $O(n^{1+\epsilon})$ for any $\epsilon > 0$. The preprocessing time is $O(N + n\log n)$, where N is the total space of the data structure. Thus the asymptotic preprocessing time is dominated by the above space bound.

3.1. Construction of the search tree.

As mentioned earlier, our data structure is based on constructing a BSP Ψ for S. Before building the tree we map the cell probabilities to an assignment of *weights* to the vertices of the subdivision as follows. Recall that n is the combinatorial complexity of S. For each trapezoid $z \in S$, we assign a weight of $w_z/4$ to each of its (at most four) corner vertices, where

$$w_z = \frac{1}{2}\max\left(p_z, \frac{1}{n}\right).$$

(See Figure 6.) If a vertex is a corner of multiple trapezoids, then its weight is the sum of the contributions from all such trapezoids. It is easy to see that the total weight of all the trapezoids is at most 1, and this holds as well for the total weight of all the vertices. The $1/n$ term in the definition of the weight will be important in limiting the fragmentation of cells of very small probability. This in turn will be needed to establish our space bounds.

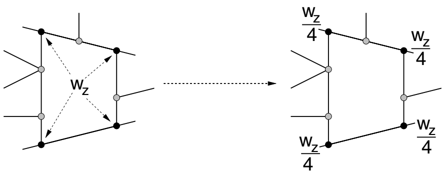


FIG. 6. *Assignment of weights.*

The tree Ψ is built recursively in a top-down fashion. Each node u of Ψ will be associated with a trapezoidal region, denoted Δ_u. We define the *weight* of a region to be the sum of the weights of the subdivision vertices in the interior of the region, and we define the *weight* of a node u to be the weight of Δ_u. The construction is based on a recursive process, which is broken into *stages*. Each stage replaces an existing leaf node whose associated region overlaps two or more cells of S with a new subtree. Let us consider one such stage. Suppose that we are working on the subdivision contained within a trapezoid Δ_u associated with some node u. (Initially u is the root of the tree and Δ_u is the entire space.) Let w_u denote u's weight. (For example, in the upper left of Figure 7(a) w_u is the sum of the weights of the black and gray vertices lying in the interior of the trapezoid Δ_u.) We split Δ_u into two vertical *slabs*, by passing a vertical line through a subdivision segment endpoint, such that the weight of each slab is at most $w_u/2$. We repeat this for t levels, where $t \geq 1$ is a suitable parameter (to be fixed later), each time ensuring that the sum of the weights is halved. This partitioning can be represented in a natural way by a balanced tree having up to 2^t leaves, representing the 2^t vertical slabs. (There may in fact be fewer, since some slabs may contain no interior vertices before the splitting process ends.)

After this, each slab is further partitioned into trapezoids by the segments of the subdivision that completely cross it. (See Figure 7(b).) Following Seidel and Adamy [27], we build a weighted search tree [22] for each slab. There is a technical difficulty, however. A trapezoid that contains no vertices in its interior (called an *empty trapezoid*) has a weight of 0, and so we cannot reasonably bound its depth in the tree. To handle this the weighted search tree is based on the following set of *adjusted weights*. The adjusted weight of a nonempty trapezoid is just its weight, that is, the sum of weights of its interior vertices. The adjusted weight of an empty trapezoid cell is defined to be $w_z/(h2^t)$, where z is the trapezoid of S that contains this cell, and where $h \geq 1$ is a suitable parameter (to be fixed later). (See Figure 7(b).) After building the weighted search tree for the trapezoids of each slab, we recurse on each of the nonempty trapezoids. The process ends when there are no nonempty trapezoids.

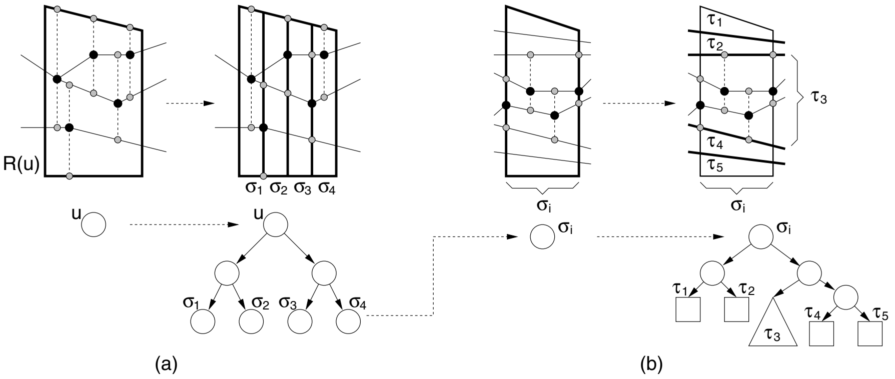


FIG. 7. *Construction of the initial search structure for $t = 2$.*

For the purpose of analysis, it is convenient to view this partitioning scheme as a *multiway tree* as follows. Each stage involves splitting a trapezoid u into 2^t vertical slabs, each of which is then partitioned into smaller trapezoids. In the multiway tree, there is a node representing trapezoid u, which is made the parent of the nodes representing these smaller trapezoids. Let Ψ' denote this multiway tree. Note that each node of Ψ' corresponds to a unique node of Ψ and represents the same region.

This construction can be implemented efficiently as follows. We assume that we maintain the following for each node u:

(i) $S_u = S \cap \Delta_u$ (i.e., the original subdivision S clipped to within Δ_u), and

(ii) a list of the vertices of S that lie within the interior of Δ_u and their associated weights, sorted according to their x-coordinates.

We assume that S_u is represented in a manner that supports efficient processing and traversal, say as a *doubly connected edge list* (DCEL) [11]. We also assume that there are cross links between each vertex of the sorted list and its corresponding vertex of the DCEL. This information can be computed for the root of the tree in $O(n \log n)$ time. Let us also define T_u to be the set of (unclipped) trapezoids of S that intersect the interior of Δ_u. Thus each (clipped) trapezoid $z' \in S_u$ arises by intersecting some (unclipped) trapezoid $z \in T_u$ with Δ_u.

Now consider the single-stage construction for a fixed internal node u of the multiway tree. Let n_u denote the combinatorial complexity of S_u. Let m_u denote the total number of empty and nonempty trapezoids that result from one stage of the construction or, equivalently, the total number of nodes in all the weighted trees for all of u's slabs. In $O(n_u)$ time it is possible to scan the sorted vertex list from left to right, to generate the (up to) 2^t slabs with the desired weights. We then cut S_u into slabs by tracing along each of the $2^t - 1$ vertical lines that define the slab boundaries. Because each cell of S_u is of constant combinatorial complexity, by standard results on DCELs [11] we can do this in $O(1)$ time for each intersection between a segment of S_u and a cutting line. The overall time is $O(m_u)$. As we are cutting out each slab, we note which of the segments of S cut clear through the slab. From this information we can determine which of the resulting trapezoids are empty and which are nonempty. Then, for each nonempty trapezoid, we can construct the DCEL representation of the subdivisions lying within the trapezoid in time proportional to its size. This takes time $O(n_u)$ when summed over all the nonempty trapezoids. After this the subdivisions are ready for the next recursive step.

Next we consider how to update the sorted vertex list. Consider each nonempty trapezoid τ_i. We traverse the associated subdivision and, for each vertex lying in the interior of the trapezoid, we access the cross link to the sorted list and label the associated list entry with the integer i. Through the use of any stable integer sorting algorithm (e.g., counting sort [9]) we sort these labels in $O(n_u)$ time. By collecting adjacent entries with the same label, we can partition the sorted vertex lists among the various nonempty trapezoids, while maintaining the sorted order, all in $O(n_u)$ time.

Thus, the total processing time for node u is $O(n_u + m_u)$. The sum of the m_u terms over all the nodes is essentially equal to the number of nodes of Ψ', which is bounded above by the total number of nodes of Ψ. Because each subdivision vertex appears in the region associated with at most one node at each level of Ψ', the sum of the n_u terms for each level is $O(n)$. In Lemma 4 below we show that the number of levels in Ψ' is at most $(1/t)\log n + 5$. Therefore the overall construction time is $O(N + n \log n)$, where N is the total number of nodes in the tree. We will bound N later in Lemma 5.

3.2. Analysis of the space and query time. We now analyze the space and expected query time as a function of the parameters t and h. For a node u of Ψ, let p_u denote the probability of the query point lying in its associated region Δ_u (or, equivalently, the probability of visiting u during point location). Recall that w_u denotes the weight of all the vertices in the interior of Δ_u. The following lemma bounds the number of levels in multiway tree Ψ'.

LEMMA 4. *The number of levels in* Ψ' *is at most* $(1/t)\log n + 5$.

Proof. Consider any path in the multiway tree Ψ' descending from the root to a leaf. Along any edge that leads from one internal node to another the weight decreases by a factor of at least 2^t. Since the weight of the root is at most 1, the weight of any internal node u at level i of Ψ' is at most $1/2^{t(i-1)}$. Since u is internal, it must contain the corner vertex of some trapezoid z in its interior. Recall that the weight of each of the corner vertices of trapezoid z is at least $w_z/4$ (more if it is a corner of multiple trapezoids). It follows that

$$\frac{1}{2^{t(i-1)}} \geq \frac{w_z}{4}.$$

Simplifying this gives

$$i \;\leq\; \frac{1}{t}\left(\log\frac{1}{w_z} + \log 4\right) + 1 \;\leq\; \frac{1}{t}\log\frac{1}{w_z} + 3.$$

Since $w_z \geq 1/(2n)$ it follows that the level of any internal node is at most $(1/t)\log n + 4$, and the number of levels is higher by 1. ◻

Using this, we can bound the total size of the tree.

LEMMA 5. *The total number of nodes of* Ψ *is at most* $O(n2^t((\log n)/t + 1))$.

Proof. Consider any trapezoid z of S. We first show that the number of leaves of Ψ generated by z is at most

$$4 \cdot 2^t \left(\frac{1}{t}\log n + 4\right).$$

It follows from our construction that any internal node of the multiway tree Ψ' that overlaps the interior of z must contain at least one of the four corner vertices of z. Thus, there are at most four such internal nodes at any level of Ψ'. (In fact, a more careful analysis shows that there are at most two such nodes, one for the left side and one for the right side.) From the proof of Lemma 4 it follows that the total number of internal nodes of Ψ' that overlap the interior of z is at most $4((1/t)\log n + 4)$. Since any node of Ψ' can have at most 2^t children that overlap the interior of z, the above bound follows.

By summing this bound over all $O(n)$ trapezoids, the total number of leaves of Ψ is at most $O(n2^t((\log n)/t + 1))$. Since Ψ is binary, the number of internal nodes cannot be larger. ◻

In Lemma 7, we bound the depth of a leaf generated from a trapezoid $z \in S$. To this end, we need the following technical result.

LEMMA 6. *Let* u *be an internal node in the multiway tree* Ψ'. *Let* σ *be any of the* 2^t *vertical slabs into which* Δ_u *is partitioned. Then the total adjusted weight of the weighted search tree corresponding to* σ *is at most* $(w_u/2^t)(1 + 4/h)$.

Proof. Recall that the segments of S partition σ into empty and nonempty trapezoids. Since the weight associated with σ is at most $w_u/2^t$, the total adjusted weight of all the nonempty trapezoids is at most $w_u/2^t$. We will show that the total adjusted weight of the empty trapezoids is at most $4w_u/(2^t h)$, which will complete the proof.

Recall that T_u denotes the set of (unclipped) trapezoids of S that intersect the interior of Δ_u. The construction implies that the trapezoids of T_u have at least one of their four corner vertices in the interior of Δ_u. Since w_u is the sum of the weights of all the vertices in the interior of Δ_u, it follows that $w_u \geq \sum_{z \in T_u} w_z/4$.

Next observe that a trapezoid of T_u can generate at most one empty trapezoid in slab σ. Recall that the adjusted weight of an empty trapezoid generated by trapezoid $z \in S$ is $w_z/(2^t h)$. Thus the total adjusted weight of the empty trapezoids in σ is at most $\sum_{z \in T_u} w_z/(2^t h)$. By the bound on w_u from the previous paragraph, this is at most $4w_u/(2^t h)$. □

The following lemma establishes the essential property given in section 2.3, by showing that the depth of any leaf associated with a cell is proportional to the logarithm of its reciprocal probability.

LEMMA 7. *Let u be a leaf of Ψ generated by any trapezoid $z \in S$. Then the depth of u in Ψ is at most*

$$\log \frac{1}{w_z} + t + \left(2 + \frac{6}{h}\right) \frac{1}{t} \log \frac{1}{w_z} + \log h + O(1).$$

Proof. Let $P = u_1, u_2, \ldots, u_\ell$ be the path from the root to the leaf $u = u_\ell$ in the multiway tree Ψ'. Recall that each node of Ψ' corresponds to a unique node of Ψ. Consider a fixed i, $1 \le i < \ell$. Let σ denote the vertical slab in Δ_{u_i} that contains the trapezoid associated with u_{i+1}, and let v denote the node of Ψ corresponding to σ. To prove the lemma, we will separately bound the length of the paths in Ψ from u_i to v and from v to u_{i+1}. By construction, the length of the path in Ψ from u_i to v is at most t.

To bound the length of the path in Ψ from v to u_{i+1}, recall that u_{i+1} is a leaf in the weighted search tree for slab σ. By standard results on weighted search trees [22], the length of the path in Ψ from v to u_{i+1} is at most $\log(W/w) + 2$, where W is the total adjusted weight of all the trapezoids in slab σ and w is the adjusted weight of the trapezoid associated with u_{i+1}. By Lemma 6, $W \le (w_{u_i}/2^t)(1 + 4/h)$. We now consider two cases: (i) $1 \le i \le \ell - 2$ and (ii) $i = \ell - 1$. In the first case, u_{i+1} is a nonempty trapezoid, so its adjusted weight w is the same as its weight $w_{u_{i+1}}$. Thus, the length of the path in Ψ from v to u_{i+1} is at most

$$\log\left(\frac{(w_{u_i}/2^t)(1+4/h)}{w_{u_{i+1}}}\right) + 2 \;=\; (\log w_{u_i} - \log w_{u_{i+1}}) - t + \log\left(1 + \frac{4}{h}\right) + 2.$$

In the second case, $u_{i+1} = u_\ell$ is an empty trapezoid, so its adjusted weight w is $w_z/(2^t h)$. Thus, the length of the path in Ψ from v to u_ℓ is at most

$$\log\left(\frac{(w_{u_{\ell-1}}/2^t)(1+4/h)}{w_z/(2^t h)}\right) + 2 \;=\; (\log w_{u_{\ell-1}} - \log w_z) + \log h + \log\left(1 + \frac{4}{h}\right) + 2.$$

By using the above claim to bound the lengths of the paths in Ψ between adjacent pairs of vertices in P, the fact that $w_{u_1} \le 1$, and summing and cancelling the telescoping probability terms, it is easy to see that the depth of u in Ψ is at most

$$\log \frac{1}{w_z} + t + \left(2 + \log\left(1 + \frac{4}{h}\right)\right)(\ell - 1) + \log h. \tag{1}$$

Recall the trapezoid z in the statement of the lemma. Since $u_{\ell-1}$ must contain at least one of the corner vertices of trapezoid z, it follows that $w_{u_{\ell-1}} \ge w_z/4$. Also, since the weight of a node at level i is at most $1/2^{t(i-1)}$, we have $w_{u_{\ell-1}} \le 1/2^{t(\ell-2)}$. Thus,

$$\ell - 2 \;\le\; \frac{1}{t} \log \frac{1}{w_{u_{\ell-1}}} \;\le\; \frac{1}{t}\left(\log \frac{1}{w_z} + \log 4\right) \;\le\; \frac{1}{t}\left(\log \frac{1}{w_z} + 2\right).$$

Substituting this value of ℓ into (1) and using the facts that $\log(1 + 4/h) < 6/h$, $h \geq 1$, and $t \geq 1$, we obtain the bound on the depth of u given in the statement of the lemma. $\square$

We can now bound the expected query time.

LEMMA 8. *The expected query time using the BSP Ψ is at most*

$$H + t + \left(2 + \frac{6}{h}\right)\frac{H}{t} + \log h + O(1).$$

Proof. For any trapezoid $z \in S$, let $\mathcal{L}_z$ denote the set of leaves generated by z. The expected query time is given by

$$\sum_{z\in S}\sum_{u\in\mathcal{L}_z} p_u d_u,$$

where p_u denotes the probability that the query point lies in the region associated with node u, and d_u denotes the depth of u. By applying Lemma 7 it follows that this sum is at most

$$\sum_{z\in S}\sum_{u\in\mathcal{L}_z} p_u \left[\log\frac{1}{w_z} + t + \left(2 + \frac{6}{h}\right)\frac{1}{t}\log\frac{1}{w_z} + \log h + O(1)\right].$$

Using the facts that $\sum_{u\in\mathcal{L}_z} p_u = p_z$ and $\sum_{z\in S} p_z = 1$, this is at most

$$\left(\sum_{z\in S} p_z \log\frac{1}{w_z}\right) + t + \left(2 + \frac{6}{h}\right)\frac{1}{t}\left(\sum_{z\in S} p_z \log\frac{1}{w_z}\right) + \log h + O(1).$$

Noting that $w_z \geq p_z/2$, $h \geq 1$, and $t \geq 1$, we obtain the desired bound. $\square$

In order to obtain the best bound on the expected query time, we choose $t = \lceil\sqrt{2H}\rceil$ and $h = \sqrt{H}$ in Lemma 8. This yields an expected query time of at most

$$H + 2\sqrt{2H} + \frac{1}{2}\log H + O(1).$$

Using Lemma 5 and noting that t is at most $O(\sqrt{\log n})$, we obtain a bound on the space of

$$O\left(n2^{\sqrt{2H}}\frac{\log n}{\sqrt{H}}\right).$$

This completes the proof of Theorem 4.

Remark. By setting $t = \lceil\sqrt{2H}\rceil + c$, where c is a fixed positive integer and $h = \sqrt{H}$, it is easy to see from Lemma 7 that the maximum depth in the search tree, that is, the worst-case query time, is $(1 + O(1/c))\log n$. Simultaneously, the bound on expected performance given by Theorem 4 also holds.

Remark. For the next section on entropy-preserving cuttings, it will be necessary to derive a bound on query times that is sensitive to the cell containing the query point. The following lemma establishes this for us.

LEMMA 9. *We are given a trapezoidal map S of size n, together with a nonnegative weight w_z for each cell $z \in S$, such that $\sum_{z\in S} w_z \leq 1$, and a real parameter* $1 \leq B \leq$

$\sqrt{\log n}$. Then we can build a data structure that answers point-location queries for each cell $z \in S$ in time

$$\left[1 + O\left(\frac{1}{B}\right)\right] \log \frac{1}{w_z} + O(B).$$

The space and preprocessing time for the data structure are $O(n2^{\sqrt{2\log n}}\sqrt{\log n})$.

Proof. Let $h = B$ and $t = \lceil B\sqrt{2}\rceil$. From Lemma 7 it follows that the time to locate a query point in any cell z of S is at most

$$\begin{aligned}\log \frac{1}{w_z} + t + \left(2 + \frac{6}{h}\right)\frac{1}{t}\log\frac{1}{w_z} + \log h + O(1)\\ &= \left[1 + \frac{1}{t}\left(2 + \frac{6}{h}\right)\right]\log\frac{1}{w_z} + (t + \log h + O(1))\\ &= \left[1 + O\left(\frac{1}{B}\right)\right]\log\frac{1}{w_z} + O(B),\end{aligned}$$

as desired.

From Lemma 5, the total number of nodes of Ψ is at most

$$O\left(n2^t\left(\frac{\log n}{t} + 1\right)\right) \;=\; O\left(n2^{B\sqrt{2}}\left(\frac{\log n}{B} + 1\right)\right).$$

For $B \geq 1/(\sqrt{2}\ln 2) \approx 1.02$, it is easy to verify that $2^{B\sqrt{2}}/B$ is an increasing function of B, and since $B \leq \sqrt{\log n}$, it follows that the space is at most

$$O\left(n2^{\sqrt{2\log n}}\left(\frac{\log n}{\sqrt{\log n}} + 1\right)\right) \;=\; O\left(n2^{\sqrt{2\log n}}\sqrt{\log n}\right).$$

Recall that the preprocessing time is $O(N + n\log n)$, where N is the total space of the data structure. Thus the preprocessing time is dominated by the above space bound. This completes the proof. □

4. Entropy-preserving cuttings.

Although the query time given in Theorem 4 is as desired, the space bound is still superlinear in n. In this section we introduce the notion of an entropy-preserving cutting, that is, a cutting that ensures that the entropy of the subdivision is increased by at most an additive constant. Then, in sections 4.1 and 4.2 we will see how to apply this idea to reduce the space to linear.

Recall that we are given a subdivision S, presented to us as a weighted trapezoidal map of a set X of nonvertical segments, whose entropy H is at least 1. The results of the previous section provide optimal expected query time (up to lower order terms), but the space required is superlinear in n. In this section we show how to apply the well-known notion of cuttings in the expected-case setting to produce a structure of linear space. Suppose that each $x \in X$ is associated with a positive weight w_x. Let $W = \sum_{x\in X} w_x$ denote the total weight. Consider a positive parameter r. For our purposes we define a $(1/r)$-*cutting* of X to be a partition of the plane into trapezoids (in general these are canonical shapes of constant combinatorial complexity) such that the total weight of the segments of X that intersect the interior of any trapezoid is at most W/r. It is known that it is possible to compute such a cutting of size $O(r)$ in $O(n\log n)$ time by a randomized algorithm, and it can be computed deterministically in polynomial time [7, 16]. The construction also provides for each trapezoid of the cutting the set of segments that intersect this trapezoid. Furthermore, this construction has the property that each trapezoid in this cutting is bounded from above and below by a subsegment of some segment of X. (See Figure 8.)

The point-location construction of the previous section was based primarily on an assignment of weights to the vertices of S. Our approach here will involve a similar assignment of weights to the segments of X. For each cell $z \in S$, recall that p_z denotes the probability that the query point lies within z, and that we make no other assumptions about the query distribution within a cell.

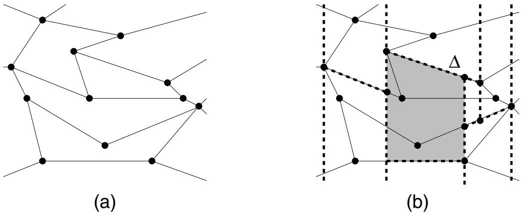


FIG. 8. *A set X of segments* (a) *and a cutting of X (shown with broken lines) and a cell of the cutting (shaded)* (b).

Each trapezoid of S can be associated by a subset of at most four *defining segments* of X, which together define its four sides. These are the segments defining the trapezoid's upper and lower sides, any segment whose endpoint lies on its left vertical side, and any segment whose endpoint lies on its right vertical side. (These segments are indicated with an asterisk in Figure 9.) We begin by assigning weights to the segments of X as follows. For each trapezoid $z \in S$, we assign a weight of $w_z/4$ to each of its defining segments, where

$$w_z = \frac{1}{2}\max\left(p_z, \frac{1}{n}\right).$$

If a segment is a defining segment of multiple trapezoids, then its weight is the sum of the contributions from all such trapezoids. Note that the total weight of all the segments is at most 1.

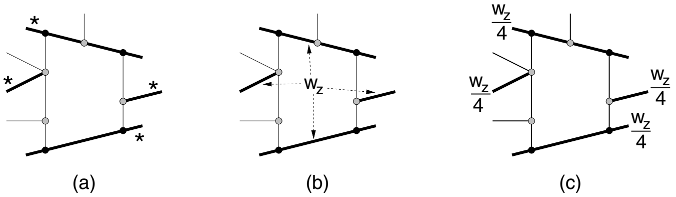


FIG. 9. *Defining segments* (a) *and the weight distribution (*(b) *and* (c)*).*

We next present our construction of entropy-preserving cuttings. Using the weight assignment w_x defined above, we compute a standard weighted $(1/r)$-cutting for X. As mentioned at the start of this section, such a cutting has size $O(r)$ and can be computed in $O(n \log n)$ time. Furthermore, each trapezoid of this cutting is bounded from above and below by a subsegment of some segment of X. Let $\mathcal{C}^*$ denote this cutting.

We modify the cutting $\mathcal{C}^*$ by applying the following procedure on each trapezoid $z \in S$. If a trapezoid of the cutting lies within z, then by the properties of the cutting, this trapezoid is bounded from above and below by the same segments that bound z from above and below. Thus, any adjacent trapezoids of the cutting that are contained within z must be separated from each other by a vertical line segment, and hence a collection of adjacent cutting trapezoids within z occurs as a contiguous sequence. If z contains such a contiguous sequence of trapezoids of $\mathcal{C}^*$, we merge each such maximal subsequence into a single trapezoid. (See Figure 10.) Since no segment of X intersects the interior of the merged trapezoid, it follows that $\mathcal{C}$ also satisfies the properties of a weighted $(1/r)$-cutting. Let $\mathcal{C}$ denote this new cutting, and let S^* denote the subdivision formed by superimposing $\mathcal{C}$ on S. In Lemma 11, we will establish the key properties enjoyed by $\mathcal{C}$. In particular, we will show that $\mathcal{C}$ is entropy-preserving; that is, the entropy of S^* exceeds the entropy of S by at most a constant.

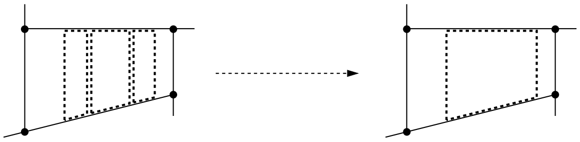


FIG. 10. *Merging contiguous trapezoids of the cutting.*

Clearly, the cells of S^* are trapezoids; we use the term *fragments* to refer to these cells. We distinguish between two types of fragments. Consider a fragment u that arises from the intersection of a cutting cell Δ and a subdivision cell z. If $\Delta \subseteq z$, we call it a *type*-1 *fragment*; otherwise it is a *type*-2 *fragment*. Let $\mathcal{F}_z, \mathcal{F}'_z$, and $\mathcal{F}''_z$ denote the set of all fragments, type-1 fragments, and type-2 fragments, respectively, that are contained within z. Let $\mathcal{F}' = \cup_{z \in S} \mathcal{F}'_z$ denote the set of all type-1 fragments and $\mathcal{F}'' = \cup_{z \in S} \mathcal{F}''_z$ denote the set of all type-2 fragments. Finally, let $\mathcal{C}'$ denote the set of trapezoids of $\mathcal{C}$ that are contained within a single cell of S, and let $\mathcal{C}''$ denote the rest of the trapezoids of $\mathcal{C}$. Note that type-1 fragments are contained inside the trapezoids of $\mathcal{C}'$ (in fact, $\mathcal{F}' = \mathcal{C}'$), while type-2 fragments are contained inside the trapezoids of $\mathcal{C}''$.

For any trapezoid Δ (not necessarily in $\mathcal{C}$), we define the following items. We let $S_\Delta = S \cap \Delta$ (i.e., the original subdivision S clipped to within Δ). We let p_Δ be the probability that the query point lies within Δ. We will use X_Δ to denote the set of segments in X that intersect the interior of Δ, and T_Δ to denote the set of trapezoids of S that intersect the interior of Δ. Observe that S_Δ and T_Δ are exactly analogous to S_u and T_u introduced in section 3.1. In particular, each (clipped) trapezoid $z' \in S_\Delta$ arises by intersecting some (unclipped) trapezoid $z \in T_\Delta$ with Δ.

The following technical lemma will be useful in proving Lemma 11.

LEMMA 10. *Let Δ be any trapezoid of $\mathcal{C}''$. Then*

(i) $\sum_{z \in T_\Delta} p_z = O(1/r)$,

(ii) $p_\Delta = O(1/r)$,

(iii) $|X_\Delta| = O(n/r)$.

Proof. Consider a trapezoid Δ in $\mathcal{C}''$ and any trapezoid z in T_Δ. We claim that at least one of the defining segments of z belongs to X_Δ. To prove this claim, recall that Δ is bounded from above and below by some segment of X. Since S is a trapezoidal map of X, it follows that z cannot cross the segments of X that bound Δ from above and below. We consider two cases. First, suppose either that the segment bounding z from above differs from the segment bounding Δ from above or that the segment bounding z from below differs from the segment bounding Δ from below. (See Figure

11(a).) Then clearly the differing segment must intersect the interior of Δ and thus belongs to X_Δ. Since the segments bounding z from above and below are defining segments of z, the claim holds. Otherwise, it follows that the segments of X bounding Δ from above and below also bound z from above and below. (See Figure 11(b).) In this case, one of the two vertical sides of z must intersect the interior of Δ (because otherwise Δ would be a subset of z, and then Δ would belong to $\mathcal{C}'$). It follows that the segment of X that defines this vertical side of z must belong to X_Δ. This proves the claim.

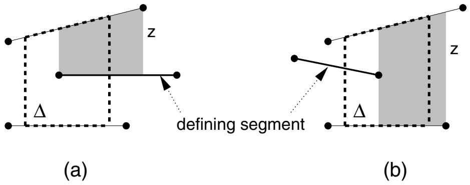


FIG. 11. *Proof of Lemma* 10.

Recall that trapezoid z assigns a weight of at least $p_z/8$ to each of its defining segments. It follows from the above claim that z assigns a weight of at least $p_z/8$ to some segment in X_Δ. Therefore,

$$\sum_{z \in T_\Delta} \frac{p_z}{8} \;\leq\; \sum_{x \in X_\Delta} w_x.$$

Since $\mathcal{C}$ is a weighted $(1/r)$-cutting for X, we have $\sum_{x \in X_\Delta} w_x = O(1/r)$. Thus, $\sum_{z \in T_\Delta} p_z = O(1/r)$, which implies (i). Recalling that each (clipped) trapezoid $z' \in S_\Delta$ is a subset of some (unclipped) trapezoid of $z \in T_\Delta$, we have $p_{z'} \leq p_z$ and so

$$p_\Delta \;=\; \sum_{z' \in S_\Delta} p_{z'} \leq \sum_{z \in T_\Delta} p_z \;=\; O(1/r),$$

which implies (ii). Finally, since the weight of each segment of X is at least $1/(8n)$ and $\sum_{x \in X_\Delta} w_x = O(1/r)$, it follows that there can be at most $O(n/r)$ segments in X_Δ. ▯

LEMMA 11 (entropy-preserving cuttings). *For any $r \geq 1$, in time $O(n \log n)$ we can partition the plane into $O(r)$ trapezoids satisfying the following properties. Let $\mathcal{C}$ denote the cutting formed by these $O(r)$ trapezoids, S^* denote the subdivision formed by superimposing $\mathcal{C}$ on S, and Δ denote any trapezoid of $\mathcal{C}$.*

(i) *If $\Delta \in \mathcal{C}''$, then the probability p_Δ that the query point lies within Δ is $O(1/r)$.*

(ii) *The number of segments, $|X_\Delta|$, intersecting the interior of Δ is $O(n/r)$.*

(iii) $\sum_{\Delta \in \mathcal{C}} \sum_{z \in T_\Delta} p_z = O(1)$.

(iv) *Let $H_S = \sum_{z \in S} p_z \log \frac{1}{p_z}$ and $H_{S^*} = \sum_{u \in S^*} p_u \log \frac{1}{p_u}$ be the respective entropies of these subdivisions. Then the increase in entropy, $H_{S^*} - H_S$, is at most a constant.*

The construction also provides for each trapezoid of the cutting the set of segments that intersect this trapezoid.

Proof. We claim that the cutting $\mathcal{C}$ described above satisfies the four properties given in the statement of the lemma. Let Δ denote any trapezoid of $\mathcal{C}$. If $\Delta \in \mathcal{C}'$, then, by definition, Δ is contained within a trapezoid of S, and so (ii) obviously holds.

Otherwise $\Delta \in \mathcal{C}''$, and, by applying Lemma 10(ii) and (iii), we have $p_\Delta = O(1/r)$ and $|X_\Delta| = O(n/r)$. This proves (i) and (ii).

We next prove (iii). For any trapezoid $z \in S$, let f_z, f'_z, and f''_z denote the number of all fragments, type-1 fragments, and type-2 fragments, respectively, that are contained within z. Observe that $\sum_{\Delta \in \mathcal{C}} \sum_{z \in T_\Delta} p_z$ is the same as $\sum_{z \in S} p_z f_z$. We will show that the latter quantity is $O(1)$, which will prove (iii).

In view of the merging process (recall Figure 10), it is clear that there must be a fragment of type 2 between any two fragments of type 1, and so $f'_z \leq f''_z + 1$. Thus

$$\sum_{z \in S} p_z f_z \;=\; \sum_{z \in S} p_z (f'_z + f''_z) \;\leq\; \sum_{z \in S} p_z (2f''_z + 1) \;=\; 1 + 2 \sum_{z \in S} p_z f''_z. \tag{2}$$

Note that $\sum_{z \in S} p_z f''_z$ is the same as $\sum_{\Delta \in \mathcal{C}''} \sum_{z \in T_\Delta} p_z$. By Lemma 10(i), for each $\Delta \in \mathcal{C}''$, $\sum_{z \in T_\Delta} p_z = O(1/r)$. Since $|\mathcal{C}''| \leq |\mathcal{C}| = O(r)$, it follows that $\sum_{z \in S} p_z f''_z = O(1)$. Substituting into (2), we have $\sum_{z \in S} p_z f_z = O(1)$, which completes the proof of (iii).

For convenience we express part (iv) of the lemma as

$$\sum_{u \in S^*} p_u \log \frac{1}{p_u} - \sum_{z \in S} p_z \log \frac{1}{p_z} \;=\; O(1). \tag{3}$$

The left-hand side of (3) can be written as

$$\sum_{z \in S} \left[\left(\sum_{u \in \mathcal{F}_z} p_u \log \frac{1}{p_u} \right) - p_z \log \frac{1}{p_z} \right].$$

Since $\sum_{u \in \mathcal{F}_z} p_u = p_z$, it follows from basic properties of entropy (see, e.g., [10]) that $\sum_{u \in \mathcal{F}_z} p_u \log(1/p_u)$ is maximized when the probability of all the fragments $u \in \mathcal{F}_z$ is equal, i.e., p_z/f_z. Thus

$$\begin{aligned}\sum_{u \in S^*} p_u \log \frac{1}{p_u} - \sum_{z \in S} p_z \log \frac{1}{p_z} &\leq \sum_{z \in S} \left[f_z \cdot \frac{p_z}{f_z} \log \frac{f_z}{p_z} - p_z \log \frac{1}{p_z} \right] \\ &= \sum_{z \in S} p_z \log f_z \;\leq\; \sum_{z \in S} p_z f_z.\end{aligned}$$

In proving (iii) above, we showed that $\sum_{z \in S} p_z f_z = O(1)$, which establishes (3).

Finally, standard cutting construction already provides the set of segments that intersect each trapezoid, and so it is trivial to adapt our construction to do so as well. □

4.1. Space reduction through entropy-preserving cuttings. We are now ready to describe our space-efficient data structure. We set the parameter r to the value $n/(2^{\sqrt{2 \log n}}\sqrt{\log n})$ and construct the entropy-preserving cutting $\mathcal{C}$ described in Lemma 11. In $O(n \log n)$ time, we obtain $\mathcal{C}$ along with X_Δ for each cutting trapezoid Δ. Let $c = \sum_{\Delta \in \mathcal{C}} \sum_{z \in T_\Delta} p_z$, which by Lemma 11(iii) is $O(1)$. For each trapezoid $\Delta \in \mathcal{C}$, we assign it a weight w_Δ proportional to the sum of probabilities of the overlapping trapezoids, that is, $w_\Delta = \sum_{z \in T_\Delta} p_z / c$. By our choice of c we have $\sum_{\Delta \in \mathcal{C}} w_\Delta = 1$. From our initial conditioning, we know that $H_S \geq 1$. So using these weights, and setting $B = \sqrt{H_S}$, we can now apply Lemma 9 to build a point-location data structure for $\mathcal{C}$. Since $|\mathcal{C}| = O(n/(2^{\sqrt{2 \log n}}\sqrt{\log n}))$, it follows that the space and preprocessing time for this data structure are $O(n)$.

Additionally, for each trapezoid $\Delta \in \mathcal{C}''$, we build any standard worst-case point-location data structure for the subdivision S_Δ. This data structure uses $O(|X_\Delta|)$ space and answers queries in time $O(\log |X_\Delta|)$. Each such structure can be computed in $O(|X_\Delta| \log |X_\Delta|)$ time by standard algorithms [11]. By Lemma 11(ii) we know that $|X_\Delta|$ is $O(2^{\sqrt{2\log n}}\sqrt{\log n})$. Since the number of trapezoids in $\mathcal{C}''$ is $O(n/(2^{\sqrt{2\log n}}\sqrt{\log n}))$, the total space and preprocessing time for all the point location structures corresponding to the trapezoids $\Delta \in \mathcal{C}''$ are $O(n)$ and $O(n\sqrt{\log n})$, respectively.

Together with the space used by the point-location structure for $\mathcal{C}$, it follows that the total space used is $O(n)$. The preprocessing time is $O(n \log n)$ and is dominated by the time to construct the cutting and determine the segments of X intersecting each cutting trapezoid. In the next section we discuss how the data structure is used to answer queries and analyze the expected query time, which is the last step needed to establish Theorem 1.

4.2. Query processing. In order to locate the cell containing a query point q, we use the point-location structure for $\mathcal{C}$ to identify the trapezoid $\Delta \in \mathcal{C}$ that contains q. If $\Delta \in \mathcal{C}'$, then we can directly output the cell of S that contains q. Otherwise $\Delta \in \mathcal{C}''$, and we use the point location structure for S_Δ to locate the cell of S_Δ (and hence of S) that contains q.

To analyze the query time, suppose that q lies in a fragment u that arises from the intersection of a trapezoid $\Delta \in \mathcal{C}$ with a cell $z \in S$. Let t_1 denote the time it takes to determine the trapezoid $\Delta \in \mathcal{C}$ that contains q. By Lemma 9 and using the facts that $B = \sqrt{H_S}$ and $w_\Delta \geq p_z/c$ (where c is the constant defined in the first paragraph of section 4.1), we have

$$\begin{aligned} t_1 &\leq \left[1 + O\left(\frac{1}{\sqrt{H_S}}\right)\right] \log \frac{c}{p_z} + O\left(\sqrt{H_S}\right) \\ &\leq \left[1 + O\left(\frac{1}{\sqrt{H_S}}\right)\right] \log \frac{1}{p_z} + O\left(\sqrt{H_S}\right). \end{aligned}$$

If $\Delta \in \mathcal{C}'$ (that is, $u \in \mathcal{F}'_z$), then we are done after finding Δ, and so the query time is t_1. Otherwise $\Delta \in \mathcal{C}''$ (that is, $u \in \mathcal{F}''_z$), and we need an additional time $t_2 = O(\log |X_\Delta|)$ to search S_Δ and determine u. By Lemma 11(ii), $|X_\Delta| = O(2^{2\sqrt{\log n}}\sqrt{\log n})$, so $t_2 = O(\sqrt{\log n})$. Putting it all together, we have shown that the expected query time is

$$\sum_{z\in S}\left[\sum_{u\in\mathcal{F}_z} p_u \left(\left[1 + O\left(\frac{1}{\sqrt{H_S}}\right)\right] \log\frac{1}{p_z} + O\left(\sqrt{H_S}\right)\right) + \sum_{u\in\mathcal{F}''_z} p_u O\left(\sqrt{\log n}\right)\right].$$

Using the facts that $\sum_{u\in\mathcal{F}_z} p_u = p_z$ and $\sum_{z\in S} p_z = 1$ and substituting H_S for $\sum_{z\in S} p_z \log(1/p_z)$, this simplifies to

$$H_S + O\left(\sqrt{H_S}\right) + O\left(p''\sqrt{\log n}\right), \tag{4}$$

where $p'' = \sum_{z\in S}\sum_{u\in\mathcal{F}''_z} p_u = \sum_{u\in\mathcal{F}''} p_u$.

To complete the analysis we need to establish a relationship between p'' and the entropy H_S. Intuitively, the following lemma shows that p'' is small when the entropy H_S is small.

LEMMA 12. *Let p'' be the probability that the query point falls in a fragment $u \in \mathcal{F}''$. Then $p'' = O(H_S/\log n)$.*

Proof. We first compute a lower bound on H_{S^*}. Clearly

$$H_{S^*} \; = \; \sum_{u \in S^*} p_u \log(1/p_u) \; \geq \; \sum_{u \in \mathcal{F}''} p_u \log(1/p_u).$$

By Lemma 11(i), for each $\Delta \in \mathcal{C}''$, $p_\Delta = O(2^{\sqrt{2\log n}}\sqrt{\log n}/n)$. Let $p_m = \max_{u \in \mathcal{F}''} p_u$. Since any fragment in $\mathcal{F}''$ is a subset of some $\Delta \in \mathcal{C}''$, it follows directly that $p_m = O(2^{\sqrt{2\log n}}\sqrt{\log n}/n)$. Recall that $p'' = \sum_{u \in \mathcal{F}''} p_u$. By basic properties of entropy [10], the entropy is minimized by maximizing the disparity among the probabilities. This is done by setting as many of the p_u's to p_m as possible, subject to the condition that they sum to p''. Thus, we obtain a lower bound on $\sum_{u \in \mathcal{F}''} p_u \log(1/p_u)$ by assuming that $\lfloor p''/p_m \rfloor$ of the fragments $u \in \mathcal{F}''$ have $p_u = p_m$, one fragment $u \in \mathcal{F}''$ has the leftover probability $p_u = p'' - \lfloor p''/p_m \rfloor p_m$, and all remaining fragments $u \in \mathcal{F}''$ have $p_u = 0$. Thus

$$H_{S^*} \; \geq \; \left(\frac{p''}{p_m} - 1\right) p_m \log \frac{1}{p_m} \; \geq \; p'' \log \frac{1}{p_m} - O(1) \; \geq \; \Omega(p'' \log n - 1).$$

Also, by Lemma 11(iv), $H_{S^*} \leq H_S + O(1)$. Combining the lower and upper bound on H_{S^*}, we obtain $H_S + O(1) = \Omega(p'' \log n - 1)$. Recalling that $H_S \geq 1$, the lemma follows. ☐

Applying Lemma 12 to (4), we see that the expected query time is

$$H_S + O\left(\sqrt{H_S}\right) + O\left(H_S/\sqrt{\log n}\right) \; \leq \; H_S + O\left(\sqrt{H_S}\right),$$

where we have used the fact that $H_S = O(\log n)$.

This completes the proof of our main result, Theorem 1.

5. Convex polygons with uniform distribution. In this section we establish Theorem 2. We are given a planar convex subdivision and assume that the query distribution within each cell is uniform. We prove the following lemma, which states that it is possible to triangulate this subdivision so that the entropy increases by only a additive constant. Theorem 2 follows by applying Theorem 1 to the resulting triangulation.

LEMMA 13. *Let S be a planar subdivision of size n whose cells are convex polygons, and assume that the query distribution within each polygon is uniform. We can triangulate each polygon such that the entropy of the resulting subdivision exceeds the entropy of S by at most an additive constant. The new subdivision can be constructed in $O(n \log n)$ time.*

The proof of this lemma relies on the straightforward observation that, given a convex polygon P, in linear time it is possible to compute a triangle whose area is at least $1/4$ the area of P. This is easy to prove by considering the triangle formed by the endpoints of the line segment defining P's diameter and the vertex of P that is farthest from this segment. (Although we do not need it, a better bound can be obtained by combining Tóth's bound of $3\sqrt{3}/4\pi \approx 0.41$ on the fraction of area of the largest triangle in a convex polygon [15] with Dobkin and Snyder's linear-time algorithm for computing the largest triangle contained in a convex polygon [12].)

Proof of Lemma 13. We triangulate each convex cell of S as follows. Let z denote any convex polygon. By the above observation we can find a triangle contained within z whose area is at least $1/4$ the area of z. We insert this triangle into the triangulation. This partitions the remainder of z into at most three convex polygons, which we triangulate recursively.

We bound the entropy of this triangulation. Let $\mathcal{F}_z$ denote the set of triangles in the triangulation of z, constructed by the above procedure. Define Φ_z to be $\sum_{x \in \mathcal{F}_z} p_x \log(1/p_x)$. We claim that

$$\Phi_z \leq p_z \log \frac{1}{p_z} + 8p_z. \tag{5}$$

The proof of this claim is by induction on the number of sides of z. For the basis case, z has three sides, and the claim is trivially true. Suppose that the claim holds for any convex polygon with at most i sides for some $i \geq 3$. We will establish the claim for any convex polygon z with $i+1$ sides.

Let y denote the first triangle added to the triangulation of z. Since the area of y is at least $1/4$ the area of z and the query distribution within z is uniform, $p_y \geq p_z/4$. Note that the remainder $z \setminus y$ consists of (at most) three convex polygons, denoted z_1, z_2, and z_3. By the induction hypothesis, $\Phi_{z_i} \leq p_{z_i} \log(1/p_{z_i}) + 8p_{z_i}$ for $1 \leq i \leq 3$. Thus Φ_z can be written as

$$\begin{aligned} p_y \log \frac{1}{p_y} + \sum_{i=1}^{3} \Phi_{z_i} &\leq p_y \log \frac{1}{p_y} + \sum_{i=1}^{3} \left(p_{z_i} \log \frac{1}{p_{z_i}} + 8p_{z_i} \right) \\ &= \left(p_y \log \frac{1}{p_y} + \sum_{i=1}^{3} p_{z_i} \log \frac{1}{p_{z_i}} \right) + 8 \sum_{i=1}^{3} p_{z_i}. \end{aligned} \tag{6}$$

Obviously $p_y + \sum_{i=1}^{3} p_{z_i} = p_z$. Since $p_y \geq p_z/4$, it follows that $\sum_{i=1}^{3} p_{z_i} \leq 3p_z/4$. Also, by basic properties of entropy [10], the maximum value of

$$p_y \log \frac{1}{p_y} + \sum_{i=1}^{3} p_{z_i} \log \frac{1}{p_{z_i}}$$

subject to the constraint that $p_y + \sum_{i=1}^{3} p_{z_i} = p_z$ occurs when $p_y = p_{z_1} = p_{z_2} = p_{z_3} = p_z/4$, and is given by $p_z \log(4/p_z)$. Using these bounds in (6), we obtain

$$\Phi_z \leq p_z \log \frac{4}{p_z} + 8 \left(\frac{3}{4} p_z \right) = p_z \log \frac{1}{p_z} + 8p_z,$$

which completes the proof by induction.

Summing both sides of (5) over all the polygons of S, it follows that the entropy of the triangulation exceeds the entropy of S by at most 8.

Finally, we discuss the time it takes to construct the triangulation. Let m denote the size of a convex polygon $z \in S$. By the observation made just prior to the proof of Lemma 13, it takes $O(m)$ time to find the first triangle y in z and decompose the remainder $z \setminus y$ into (at most) three convex polygons z_1, z_2, and z_3. If $z \setminus y$ consists of just one polygon with $m-1$ vertices, then continuing in this way, it may take $O(m^2)$ time to complete the triangulation of z. To reduce this to $O(m \log m)$, we make a small change to the construction. At each step in the recursion, we partition the current polygon into two polygons with roughly equal numbers of vertices, and then find a triangle with large area in each of these two polygons. This reduces the depth of the recursion to $O(\log m)$, and since the time taken for each level of the recursion is $O(m)$, the total time becomes $O(m \log m)$. It is now a simple exercise to extend the above proof to show that this modified triangulation algorithm also preserves the entropy (only the constant 8 increases). ◻

6. Concluding remarks. We have shown that, given a polygonal subdivision S of size n consisting of cells of constant combinatorial complexity and a query distribution presented as a weight assignment to the cells of S, it is possible to answer point location queries $H + O(\sqrt{H} + 1)$, where H is the entropy of the subdivision. Our data structure requires $O(n)$ space, and it can be constructed in $O(n \log n)$ time by a randomized algorithm. We make no assumptions about the distribution of query points within each cell of the subdivision. We have also shown that this result can be extended to convex subdivisions of arbitrary combinatorial complexity, assuming that the query distribution is uniform within each cell.

There are a number of open problems suggested here. If point location is based on the results of primitive comparisons, it is known that the entropy is a lower bound on the expected running time, and so our results are optimal up to lower order terms. Assuming that the query distribution within the cells is unknown, a stronger lower bound of $H + \sqrt{H} - O(1)$ is known in the trapezoidal search graph model [21]. Can the lower-bound analysis be refined to justify the presence of the $O(\sqrt{H})$ term, when information on the query distribution within the cells is available? Taking this in a different direction, suppose that the query distribution is not known at all. That is, the probabilities that the query point lies within the various cells of the subdivision are unknown. In the 1-dimensional case it is known that there exist self-adjusting data structures, such as splay trees [29], that achieve good expected query time in the limit. Do such self-adjusting structures exist for planar point location?

A related observation is that, in the case of worst-case query time, Seidel and Adamy [27] showed an analogous lower order term of $2\sqrt{\log n}$. An interesting question along these lines is whether it is possible to reduce the lower order term in Theorem 4, say to $2\sqrt{H}$. Another interesting question is the computational complexity of computing the BSP that minimizes the expected search time, assuming, say, the extended trapezoidal search graph model and an oracle that can answer questions about the query distribution.

As mentioned earlier, our results are based on the trapezoidal search graph model used by Seidel and Adamy [27]. The two comparison primitives used in the trapezoidal search graph model have significantly different computation times. (One is a 1-dimensional orientation test, and the other is a 2-dimensional orientation test.) This raises the question of the computational complexity of point location in even more primitive models of computation, for example, the number of arithmetic operations on coordinates.

Acknowledgments. We would like to thank Siu-Wing Cheng, Ramesh Hariharan, and Raimund Seidel for helpful discussions. We would also like to thank the anonymous referees for their careful reading and many helpful suggestions.

REFERENCES

[1] S. Arya, S.-W. Cheng, D. M. Mount, and H. Ramesh, *Efficient expected-case algorithms for planar point location*, in Proceedings of the 7th Annual Scandinavian Workshop on Algorithm Theory (Bergen, Norway, 2000), Lecture Notes in Comput. Sci. 1851, Springer-Verlag, Berlin, 2000, pp. 353–366.

[2] S. Arya and H.-Y. A. Fu, *Expected-case complexity of approximate nearest neighbor searching*, SIAM J. Comput., 32 (2003), pp. 793–815.

[3] S. Arya, T. Malamatos, and D. M. Mount, *Nearly optimal expected-case planar point location*, in Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science, 2000, pp. 208–218.

[4] S. Arya, T. Malamatos, and D. M. Mount, *Entropy-preserving cuttings and space-efficient planar point location*, in Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms, Washington, DC, 2001, SIAM, Philadelphia, 2001, pp. 256–261.

[5] S. Arya, T. Malamatos, and D. M. Mount, *A simple entropy-based algorithm for planar point location*, ACM Trans. Algorithms, 3 (2007), article 17.

[6] J. L. Bentley and T. A. Ottmann, *Algorithms for reporting and counting geometric intersections*, IEEE Trans. Comput., C-28 (1979), pp. 643–647.

[7] B. Chazelle and J. Friedman, *A deterministic view of random sampling and its use in geometry*, Combinatorica, 10 (1990), pp. 229–249.

[8] R. Cole, *Searching and storing similar lists*, J. Algorithms, 7 (1986), pp. 202–220.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed., MIT Press, Cambridge, MA, 2001.

[10] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley-Interscience, New York, 1991.

[11] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, 2nd ed., Springer-Verlag, Berlin, 2000.

[12] D. P. Dobkin and L. Snyder, *On a general method for maximizing and minimizing among certain geometric problems*, in Proceedings of the 20th Annual IEEE Symposium on Foundations of Computer Science, 1979, pp. 9–17.

[13] H. Edelsbrunner, L. J. Guibas, and J. Stolfi, *Optimal point location in a monotone subdivision*, SIAM J. Comput., 15 (1986), pp. 317–340.

[14] H. Edelsbrunner and E. P. Mücke, *Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms*, ACM Trans. Graph., 9 (1990), pp. 66–104.

[15] L. Fejes Tóth, *Lagerungen in der Ebene, auf der Kugel und im Raum*, 1st ed., Springer-Verlag, Berlin, 1953.

[16] M. T. Goodrich, M. Orletsky, and K. Ramaiyer, *Methods for achieving fast query times in point location data structures*, in Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, 1997, SIAM, Philadelphia, 1997, pp. 757–766.

[17] T. C. Hu and A. C. Tucker, *Optimal computer search trees and variable length alphabetical codes*, SIAM J. Appl. Math., 21 (1971), pp. 514–532.

[18] J. Iacono, *Optimal planar point location*, in Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms, Washington, DC, 2001, SIAM, Philadelphia, 2001, pp. 340–341.

[19] D. Kirkpatrick, *Optimal search in planar subdivisions*, SIAM J. Comput., 12 (1983), pp. 28–35.

[20] D. E. Knuth, *Sorting and Searching*, The Art of Computer Programming, Vol. 3, 2nd ed., Addison-Wesley, Reading, MA, 1998.

[21] T. Malamatos, *Expected-case Planar Point Location*, Ph.D. thesis, Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong, 2002.

[22] K. Mehlhorn, *A best possible bound for the weighted path length of binary search trees*, SIAM J. Comput., 6 (1977), pp. 235–239.

[23] K. Mulmuley, *A fast planar partition algorithm.* I, J. Symbolic Comput., 10 (1990), pp. 253–280.

[24] F. P. Preparata, *A new approach to planar point location*, SIAM J. Comput., 10 (1981), pp. 473–482.

[25] N. Sarnak and R. E. Tarjan, *Planar point location using persistent search trees*, Comm. ACM, 29 (1986), pp. 669–679.

[26] R. Seidel, *A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons*, Comput. Geom., 1 (1991), pp. 51–64.

[27] R. Seidel and U. Adamy, *On the exact worst case complexity of planar point location*, J. Algorithms, 37 (2000), pp. 189–217.

[28] C. E. Shannon, *A mathematical theory of communication*, Bell System Tech. J., 27 (1948), pp. 379–423, 623–656.

[29] D. D. Sleator and R. E. Tarjan, *Self-adjusting binary search trees*, J. ACM, 32 (1985), pp. 652–686.

SIAM J. COMPUT.
Vol. 37, No. 2, pp. 611–629

SELF-TESTING OF UNIVERSAL AND FAULT-TOLERANT SETS OF QUANTUM GATES*

WIM VAN DAM[†], FRÉDÉRIC MAGNIEZ[‡], MICHELE MOSCA[§], AND MIKLOS SANTHA[‡]

Abstract. We consider the design of self-testers for quantum gates. A self-tester for the gates $\boldsymbol{F}_1, \dots, \boldsymbol{F}_m$ is a procedure that, given any gates $\boldsymbol{G}_1, \dots, \boldsymbol{G}_m$, decides with high probability if each $\boldsymbol{G}_i$ is close to $\boldsymbol{F}_i$. This decision has to rely only on measuring in the computational basis the effect of iterating the gates on the classical states. It turns out that, instead of individual gates, we can design only procedures for families of gates. To achieve our goal we borrow some elegant ideas of the theory of program testing: We characterize the gate families by specific properties, develop a theory of robustness for them, and show that they lead to self-testers. In particular we prove that the universal and fault-tolerant set of gates consisting of a Hadamard gate, a c-NOT gate, and a phase rotation gate of angle $\pi/4$ is self-testable.

Key words. self-testing, quantum circuit, fault-tolerance, robustness

AMS subject classifications. 68M15, 94C12, 68Q60, 14P10

DOI. 10.1137/S0097539702404377

1. Introduction.

As experimentalists attempt to realize quantum computers, we need some way to test whether the desired quantum operations are actually being implemented. Our motivation is to derive sufficient and self-contained tests for verifying the action of specific finite sets of quantum gates. One of the most important features of our work is that our tests do not rely on the use of some other trusted quantum operations that have somehow already been characterized and tested.

Inspired by classical work on self-testing programs [8, 26, 21, 15] (see section 1.1), our approach is to characterize quantum gates by testable properties. For example, one testable property of the Hadamard gate $\boldsymbol{H}$ is that if one starts with input $|0\rangle$, applies $\boldsymbol{H}$, and then measures, one should measure $|0\rangle$ with probability $\frac{1}{2}$. This of course does not uniquely characterize the Hadamard gate; for instance, there are many nonunitary quantum gates with the same property. If a gate is known to be unitary, then it is quite easy to find a set of testable properties that characterize it. So one of our key techniques for characterizing gates is a test for unitarity. Since any reasonable test could verify only that the probability of outputting $|0\rangle$ is likely very close to $\frac{1}{2}$, we need *robust* properties. Informally, a property is robust if, whenever a function satisfies the property approximately, it is close to a function that satisfies it exactly.

*Received by the editors March 20, 2002; accepted for publication (in revised form) February 11, 2007; published electronically June 12, 2007. A preliminary version of this paper appeared in *Proceedings of the* 32*nd ACM Symposium on Theory of Computing*, 2000, pp. 688–696. This work has been supported by C.E.S.G., Wolfson College Oxford, Hewlett-Packard, European TMR Research Network ERP-4061PL95-1412, the Institute for Logic, Language and Computation in Amsterdam, the EC programs RESQ IST-2001-37559 and IST Integrated Project Qubit Applications (QAP) 015848, the ANR Blanc AlgoQP grant of the French Research Ministry, NSERC, MITACS, ORDCF, British-French Bilateral Project ALLIANCE no. 98101.

http://www.siam.org/journals/sicomp/37-2/40437.html

[†] Department of Computer Science, University of California, Santa Barbara, Santa Barbara, CA 93106 (vandam@cs.ucsb.edu).

[‡] Centre National de la Recherche Scientifique, Laboratoire de Recherche en Informatique, Université Paris-Sud, F-91405 Orsay Cedex, France (magniez@lri.fr, santha@lri.fr).

[§] Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON, N2L 3G1, Canada (mmosca@iqc.uwaterloo.ca).

Our tests are in the quantum circuit model of computation, which corresponds most naturally to what experimentalists are implementing. The quantum circuit model and the quantum Turing machine are the first formal models of quantum computing that were defined by Deutsch [11, 12]. Yao has shown [31] that these two models have polynomially equivalent computational power when the circuits are uniform.

A quantum circuit operates on n quantum bits (qubits), where n is some integer. The actual computation takes place in the Hilbert space $\mathbb{C}^{\{0,1\}^n}$, whose computational basis consists of the 2^n orthonormal vectors $|i\rangle$ for $i \in \{0,1\}^n$. According to the standard model, during the computation the state of the system is a unit length linear combination, or a superposition, of the basis states. The computational steps of the system are done by quantum gates, which perform unitary operations and are local in the sense that they involve only a constant number of qubits. At the end of the computation a measurement takes place on one of the qubits. This is a probabilistic experiment whose outcome can be 0 or 1, and the probability of measuring the bit b is the squared length of the projection of the superposition to the subspace spanned by the basis states that are compatible with the outcome. As a result of a measurement, the state of the system becomes this projected state.

The most convenient way to describe all possible operations on a quantum register is in the formalism of "density matrices." In this approach, which differs from the Dirac notation, the quantum operations are described by completely positive superoperators (CPSOs) that act on matrices. These density matrices describe mixed states (that is, classical probability distributions over pure quantum states), and the CPSOs correspond exactly to all of the physically allowed transformations on them. Such a model of quantum circuits with mixed states was described by Aharonov, Kitaev, and Nisan [3], and we will adopt it here. The unitary quantum gates of the standard model and measurements are special CPSOs. CPSOs can be simulated by unitary quantum gates on a larger number of qubits, and in [3] it was shown that the computational powers of the two models are polynomially equivalent.

Unitary quantum gates for a small number of qubits have been extensively studied. One reason is that, although quantum gates for up to three qubits have already been realized (e.g., in [20]), constructing gates for large numbers seems to be elusive. Another reason is that universal sets of gates can be built from them, which means that they can simulate (approximately) any unitary transformation on an arbitrary number of qubits. The first universal quantum gate which operates on three qubits was identified by Deutsch [12]. After a long sequence of work on universal quantum gates [14, 4, 13, 22, 6, 30, 18, 17], Boykin et al. [7] have recently shown that the set consisting of a Hadamard gate, a c-NOT gate, and a phase rotation gate of angle $\pi/4$ is universal. In order to form a practical basis for quantum computation, a universal set must also be able to operate in a noisy environment, and therefore there has to be an implementation of fault-tolerant quantum computation using this set of gates [30, 2, 17, 19]. The above set of three gates has the additional advantage of also being fault-tolerant in this sense.

In this paper we develop the theory of self-testing of quantum gates by classical procedures. Given a CPSO $\boldsymbol{G}$ for n qubits, and a family $\mathcal{F}$ of unitary CPSOs, we would like to decide if $\boldsymbol{G}$ belongs to $\mathcal{F}$. Intuitively, a self-tester is a procedure that answers the question "$\boldsymbol{G} \in \mathcal{F}$?" by interacting with the CPSO $\boldsymbol{G}$ in a purely classical way. More precisely, it will be a probabilistic algorithm that is able to access $\boldsymbol{G}$ as a black box in the following sense: It can prepare the classical states $w \in \{0,1\}^n$, iterate $\boldsymbol{G}$ on these states, and, afterwards, measure in the computational basis. The access must be seen as a whole, performed by a specific, experimental oracle for $\boldsymbol{G}$: Once the basis state w and the number of iterations k have been specified, the program in one step gets back one of the possible probabilistic outcomes of measuring the state of the system after $\boldsymbol{G}$ is iterated k times on w. The intermediate quantum states of this process cannot be used by the program, which cannot perform any other quantum operations either. For $0 \le \delta_1 \le \delta_2$, such an algorithm will be a (δ_1, δ_2)-tester for $\mathcal{F}$ if for every CPSO $\boldsymbol{G}$, whenever the distance of $\boldsymbol{G}$ and $\mathcal{F}$ is at most δ_1 (in some norm), it accepts with high probability, and whenever the same distance is greater than δ_2, it rejects with high probability, where the probability is taken over the measurements performed by the oracle and by the coin tosses of the algorithm. Finally we will say that $\mathcal{F}$ is *testable* if for every $\delta_2 > 0$, there exists $0 < \delta_1 \le \delta_2$ such that there exists a (δ_1, δ_2)-tester for $\mathcal{F}$. These definitions can be extended to several classes of CPSOs.

We note in section 2 that for any real φ the states $|1\rangle$ and $e^{i\varphi}|1\rangle$ are experimentally indistinguishable. This implies that if we start by distinguishing only the classical states 0 and 1, then there are families of CPSOs which are indistinguishable as well. For example, let $\boldsymbol{H}$ be the well-known Hadamard gate, and let $\boldsymbol{H}_\varphi$ be the same gate expressed in the basis $(|0\rangle, e^{i\varphi}|1\rangle)$ for $\varphi \in [0, 2\pi)$. Any experiment that starts in state 0 or 1 and uses only $\boldsymbol{H}$ will produce outcomes 0 and 1 with the same probabilities as the same experiment with $\boldsymbol{H}_{\boldsymbol{\varphi}}$. Thus no experiment that uses this quantum gate alone can distinguish it from all of the other Hadamard gates. Indeed, a family $\mathcal{F}$ containing $\boldsymbol{H}$ can be tested only if the entire Hadamard family $\mathcal{H} = \{\boldsymbol{H}_\varphi : \varphi \in [0, 2\pi)\}$ is included in $\mathcal{F}$. This degree of freedom is formalized generally for any gate in Fact 4.1.

It might seem at first sight that not being able to get rid of this degree of freedom is a serious handicap. Nonetheless, it remains coherent when we test several gates simultaneously. Thus, for example, if we define $\mathbf{NOT}_\varphi$ similarly to $\boldsymbol{H}_\varphi$, we are able to test the family of couples $\{(\mathbf{NOT}_\varphi, \boldsymbol{H}_\varphi) : \varphi \in [0, 2\pi)\}$.

The main result of this paper is Theorem 6.5, which states that for several sets of unitary CPSOs, in particular, the Hadamard gates family, Hadamard gates together with c-NOT gates, and Hadamard gates with c-NOT and phase rotation gates of angle $\pm\pi/4$, are testable. This last family is of particular importance since every triplet in the family forms a universal and fault-tolerant set of gates for quantum computation [7].

For the proof we will define the notion of experimental equations which are functional equations for CPSOs corresponding to the properties of the quantum gate that a self-tester can approximately test. These tests are done via the interaction with the experimental oracle. The proof itself contains three parts. In Theorems 4.2, 4.4, and 4.5 we will exhibit experimental equations for the families of unitary CPSOs that we want to characterize. In Theorem 5.2 we will show that actually all experimental equations are robust; in fact, the distance of a CPSO from the target family is polynomially related to the error tolerated in the experimental equations. Finally Theorem 6.3 gives self-testers for CPSO families which are characterized by a finite set of robust experimental equations.

In some cases, we are able to calculate explicitly the polynomial bound in the robustness of experimental equations. Such a result will be illustrated in Theorem 5.4 for the equations characterizing the Hadamard family $\mathcal{H}$.

Technically, these results will be based on the representation of one-qubit states and CPSOs in $\mathbb{R}^3$, where they are, respectively, vectors in the unit ball of $\mathbb{R}^3$ and particular affine transformations. This correspondence is known as the Bloch ball representation.

1.1. Related prior work. Experimental procedures for determining the properties of quantum "black boxes" were given by Chuang and Nielsen [10] and Poyatos, Cirac, and Zoller [24]; however, these procedures implicitly require an apparatus that has already been tested and characterized.

The idea of self-testing in quantum devices is implicit in the work of Adleman, Demarrais, and Huang [1]. They have developed a procedure by which a quantum Turing machine is able to estimate its internal angle by its own means under the hypothesis that the machine is unitary. In the context of quantum cryptography Mayers and Yao [23] have designed tests for deciding if a photon source is perfect. These tests guarantee that if a source passes them, then it is adequate for the security of the Bennett-Brassard [5] quantum key distribution protocol.

The study of self-testing programs is a well-established research area which was initiated by the work of Blum, Luby, and Rubinfeld [8], Rubinfeld [26], Lipton [21], and Gemmel et al. [15]. The purpose of a self-tester for a function family is to detect by simple means if a program which is accessible as an oracle computes a function from the given family. This clearly inspired the definition of our self-testers, which have the particular feature that they should test quantum objects that they can access only in some particular way. The analogy with self-testing does not stop with the definition. One of the main tools in self-testing of function families is the characterization of these families by robust properties. The concept of robustness was introduced and its implication for self-testing was first studied by Rubinfeld and Sudan [27] and by Rubinfeld [28]. It will play a crucial role in our case.

2. Preliminaries.

2.1. The quantum state. A *pure state* in a quantum physical system is described by a unit vector in a Hilbert space. In the *Dirac* notation it is denoted by $|\psi\rangle$. In particular a *qubit* (a quantum two-state system) is an element of the Hilbert space $\mathbb{C}^{\{0,1\}}$. The orthonormal basis containing $|0\rangle$ and $|1\rangle$ is called the *computational basis* of $\mathbb{C}^{\{0,1\}}$. Therefore a pure state $|\psi\rangle \in \mathbb{C}^{\{0,1\}}$ is a linear combination, or a *superposition*, of the computational basis states, that is, $|\psi\rangle = c_0|0\rangle + c_1|1\rangle$, with $|c_0|^2 + |c_1|^2 = 1$. A physical system which deals with n qubits is described mathematically by the 2^n-dimensional Hilbert space which is by definition $\mathbb{C}^{\{0,1\}} \otimes \cdots \otimes \mathbb{C}^{\{0,1\}}$, that is, the n^{th} tensor power of $\mathbb{C}^{\{0,1\}}$. Let $N = 2^n$. The computational basis of this space consists of the N orthonormal states $|i\rangle$ for $0 \leq i < N$. If i is in binary notation $i_1 i_2 \ldots i_n$, then $|i_1 \ldots i_n\rangle = |i_1\rangle \ldots |i_n\rangle$, where this is a short notation for $|i_1\rangle \otimes \cdots \otimes |i_n\rangle$. All vectors and matrices will be expressed in the computational basis. The transposed complex conjugate $|\psi\rangle^\dagger$ of $|\psi\rangle$ is denoted by $\langle\psi|$. The inner product between $|\psi\rangle$ and $|\psi'\rangle$ is denoted by $\langle\psi|\psi'\rangle$ and their outer product by $|\psi\rangle\langle\psi'|$.

Quantum systems can also be in more general states than what can be described by pure states. The most general states are *mixed states*, described by a probability distribution over pure states. Such a mixture can be denoted by $\{(p_k, |\psi_k\rangle) : k \in \mathbb{N}\}$, where the system is in the pure state $|\psi_k\rangle$ with probability p_k.

Different mixtures (even different pure states $|\psi\rangle$) can represent the same physical system. This notational redundancy can be avoided if we use the formalism of the density matrices. A *density matrix* that represents an n-qubit state is an $N \times N$ Hermitian semipositive matrix with trace 1. The pure state $|\psi\rangle$ in this representation is described by the density matrix $\psi = |\psi\rangle\langle\psi|$ and a mixture $\{(p_k, |\psi_k\rangle) : \ k \in \mathbb{N}\}$ by the density matrix $\psi = \sum_{k\in\mathbb{N}} p_k |\psi_k\rangle\langle\psi_k|$. For example, the pure states $e^{i\gamma}|\psi\rangle$, for $\gamma \in [0, 2\pi)$, or the mixtures $\{(\frac{1}{2}, |0\rangle), (\frac{1}{2}, |1\rangle)\}$ and $\{(\frac{1}{2}, \frac{|0\rangle+|1\rangle}{\sqrt{2}}), (\frac{1}{2}, \frac{|0\rangle-|1\rangle}{\sqrt{2}})\}$ have, respectively, the same density matrix.

Since a density matrix is Hermitian semipositive, its eigenvectors are orthogonal and its eigenvalues are nonnegative. Because the density matrix has trace 1, its eigenvalues sum to 1. Therefore a density matrix represents the mixture of its orthonormal eigenvectors, where the probabilities are the respective eigenvalues. Note that diagonal density matrices correspond to a mixture over pure states $|i\rangle$ for $0 \leq i < N$. Density matrices that represent pure states have a simple algebraic characterization: ρ is a pure state if and only if it has two eigenvalues: 0 with multiplicity $N-1$ and 1 with multiplicity 1; equivalently, ρ is a pure state exactly when $\rho^2 = \rho$.

A 2×2 Hermitian matrix of unit trace is semipositive if and only if its determinant is between 0 and 1/4. Therefore in the case of one qubit, any density matrix ρ can be written as $\rho = p|0\rangle\langle 0| + (1-p)|1\rangle\langle 1| + \alpha|1\rangle\langle 0| + \alpha^*|0\rangle\langle 1|$, where $p \in [0,1]$, and α is a complex number such that $|\alpha|^2 \leq p(1-p)$. This density matrix will be denoted by $\rho(p,\alpha)$. Remark that $\rho(p,\alpha)$ is a pure state exactly when $|\alpha|^2 = p(1-p)$; that is, its determinant is 0.

2.2. Superoperators. The evolution of physical systems is described by specific transformations on density matrices, that is, on operators. A *superoperator* for n qubits is a linear transformation on $\mathbb{C}^{N\times N}$. A *positive* superoperator (PSO) is a superoperator that maps density matrices to density matrices. A *completely positive* superoperator (CPSO) $\boldsymbol{G}$ is a PSO such that, for all positive integers M, $\boldsymbol{G} \otimes \boldsymbol{I}_M$ is also a PSO, where $\boldsymbol{I}_M$ is the identity on $\mathbb{C}^{M\times M}$. CPSOs are exactly the physically allowed transformations on density matrices. An example of a PSO for one qubit that is not a CPSO is the transpose superoperator $\boldsymbol{T}$ defined by $\boldsymbol{T}(|i\rangle\langle j|) = |j\rangle\langle i|$ for $0 \leq i,j \leq 1$.

Quantum computation is traditionally based on the possibility of constructing some particular CPSOs, *unitary* superoperators, which preserve the set of pure states. These operators are characterized by transformations from $\mathrm{U}(N)$, the set of $N \times N$ unitary matrices. For any $A \in \mathrm{U}(N)$, we define a CPSO which maps a density matrix ρ into $A\rho A^\dagger$. When the underlying unitary transformation A is clear from the context, by a slight abuse of notation we will denote this CPSO simply by $\boldsymbol{A}$. If $|\psi'\rangle$ denotes $A|\psi\rangle$, then the unitary superoperator $\boldsymbol{A}$ maps the pure state ψ to the pure state ψ'. As was the case in the Dirac representation of states, there is the same phase redundancy in the set of unitary transformations $\mathrm{U}(N)$. If $A \in \mathrm{U}(N)$, then for all $\gamma \in [0, 2\pi)$ the transformations $e^{i\gamma}A$ are different; however, the corresponding superoperators are identical. We will therefore focus on $\mathrm{U}(N)/\mathrm{U}(1)$.

Conversely, CPSOs can be defined using unitary transformations. For every CPSO $\boldsymbol{G}$ for n qubits, there exists a unitary transformation $A \in \mathrm{U}(2^{3n})$ for $3n$ qubits such that $\boldsymbol{G}$ corresponds to the application of A after tracing out the additional n qubits [3]: G maps a density matrix ρ into $\boldsymbol{G}(\rho) = \mathrm{Tr}_2(A(\rho \otimes I_{2^{2n}})A^\dagger)$, where Tr_2 denotes the trace out over the last $2n$ qubits.

2.3. Measurements. Measurements form another important class of (nonunitary) CPSOs. They describe physical transformations corresponding to the observation of the system. We will now formally define one of the simplest classes of measurements which correspond to the projections to elements of the computational basis.

A *von Neumann measurement in the computational basis* of n qubits is the n-qubit CPSO $\boldsymbol{M}$ that, for every density matrix ρ, satisfies $\boldsymbol{M}(\rho)_{i,i} = \rho_{i,i}$ and $\boldsymbol{M}(\rho)_{i,j} = 0$ for $i \neq j$.

In the case of one qubit, the von Neumann measurement in the computational basis maps the density matrix $\rho(p,\alpha)$ into $\rho(p,0)$. We will say that $p = \langle 0|\rho|0\rangle$ is the *probability of measuring* $|0\rangle\langle 0|$, and we will denote it by $\Pr^0[\rho]$.

In general, a *von Neumann measurement* of n qubits in any basis can be viewed as the von Neumann measurement in the computational basis preceded by some unitary superoperator.

2.4. The Bloch ball representation. Specific for the one-qubit case, there is an isomorphism between the group $\mathrm{U}(2)/\mathrm{U}(1)$ and the special rotation group $\mathrm{SO}(3)$, the set of 3×3 orthogonal matrices with determinant 1. This allows us to represent one-qubit states as vectors in the unit ball of $\mathbb{R}^3$ and unitary superoperators as rotations on $\mathbb{R}^3$. We will now describe exactly this correspondence.

The *Bloch ball* $\mathcal{B}$ (respectively, *Bloch sphere* $\mathcal{S}$) is the unit ball (respectively, unit sphere) of the Euclidean affine space $\mathbb{R}^3$. Any point $\overline{u} \in \mathbb{R}^3$ determines a vector with the same coordinates which we will also denote by $\overline{u}$. The inner product of $\overline{u}$ and $\overline{v}$ will be denoted by $(\overline{u}, \overline{v})$ and their Euclidean norm by $\|\overline{u}\|$.

Each point $\overline{u} \in \mathbb{R}^3$ can be also characterized by its norm $r \geq 0$, its latitude $\theta \in [0, \pi]$, and its longitude $\varphi \in [0, 2\pi)$. The *latitude* is the angle between the z-axis and the vector $\overline{u}$, and the *longitude* is the angle between the x-axis and the orthogonal projection of $\overline{u}$ in the plane defined by $z = 0$. If $\overline{u} = (x, y, z)$, then these parameters satisfy $x = r \sin\theta \cos\varphi$, $y = r \sin\theta \sin\varphi$, and $z = r\cos\theta$.

For every density matrix ρ for one qubit there exists a unique point $\overline{\rho} = (x, y, z) \in \mathcal{B}$ such that

$$\rho = \frac{1}{2}\begin{pmatrix} 1+z & x-iy \\ x+iy & 1-z \end{pmatrix}.$$

This mapping is a bijection that also obeys

$$\overline{\rho(p,\alpha)} = (2\mathrm{Re}(\alpha), 2\mathrm{Im}(\alpha), 2p-1).$$

In this formalism, the pure states are nicely characterized in $\mathcal{B}$ by their norm.

FACT 2.1. *A density matrix ρ represents a pure state if and only if $\overline{\rho} \in \mathcal{S}$; that is, $\|\overline{\rho}\| = 1$.*

Also, if $\theta \in [0, \pi]$ and $\varphi \in [0, 2\pi)$ are, respectively, the latitude and the longitude of $\overline{\psi} \in \mathcal{S}$, then the corresponding density matrix represents a pure state and satisfies $|\psi\rangle = \cos(\theta/2)|0\rangle + \sin(\theta/2)e^{i\varphi}|1\rangle$ (see Figure 2.1). Observe that the pure states $|\psi\rangle$ and $|\psi^\perp\rangle$ are orthogonal if and only if $\overline{\psi} = -\overline{\psi^\perp}$. We will use the following notation for the six pure states along the x, y, and z axes: $|\zeta_x^\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$, $|\zeta_y^\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm i|1\rangle)$, $|\zeta_z^+\rangle = |0\rangle$, and $|\zeta_z^-\rangle = |1\rangle$, with the respective coordinates $(\pm 1, 0, 0)$, $(0, \pm 1, 0)$, and $(0, 0, \pm 1)$ in $\mathbb{R}^3$. Recall that for every density matrix ρ for one qubit there exists two orthogonal pure states $|\psi\rangle$ and $|\psi^\perp\rangle$ such that $\rho = p|\psi\rangle\langle\psi| + (1{-}p)|\psi^\perp\rangle\langle\psi^\perp|$, where $0 \leq p \leq 1$. Thus $\overline{\rho}$ is just the barycenter of $\overline{\psi}$ and $\overline{\psi^\perp}$ with respective weights p and $(1{-}p)$ (see Figure 2.2).

For each CPSO $\boldsymbol{G}$, there exists a unique affine transformation $\overline{\boldsymbol{G}}$ over $\mathbb{R}^3$, which maps the ball $\mathcal{B}$ into $\mathcal{B}$ and is such that, for all density matrices ρ, $\overline{\boldsymbol{G}}(\overline{\rho}) = \overline{\boldsymbol{G}(\rho)}$. Unitary superoperators have a nice characterization in $\mathcal{B}$.

FACT 2.2. *The map between* $\mathrm{U}(2)/\mathrm{U}(1)$ *and* $\mathrm{SO}(3)$*, which sends A to $\overline{\boldsymbol{A}}$, is an isomorphism.*

For $\alpha \in (-\pi, \pi]$, $\theta \in [0, \frac{\pi}{2}]$, and $\varphi \in [0, 2\pi)$, we will define the unitary transformation $R_{\alpha,\theta,\varphi}$ over $\mathbb{C}^2$. If $|\psi\rangle = \cos(\theta/2)|0\rangle + e^{i\varphi}\sin(\theta/2)|1\rangle$ and $|\psi^\perp\rangle = \sin(\theta/2)|0\rangle - e^{i\varphi}\cos(\theta/2)|1\rangle$, then by definition $R_{\alpha,\theta,\varphi}|\psi\rangle = |\psi\rangle$ and $R_{\alpha,\theta,\varphi}|\psi^\perp\rangle = e^{i\alpha}|\psi^\perp\rangle$. If $\boldsymbol{A}$ is a unitary superoperator, then we have $\boldsymbol{A} = \boldsymbol{R}_{\alpha,\theta,\varphi}$ for some α, θ, and φ. In $\mathbb{R}^3$ the transformation $\overline{\boldsymbol{R}}_{\alpha,\theta,\varphi}$ is the rotation of angle α whose axis cuts the sphere $\mathcal{S}$ in the

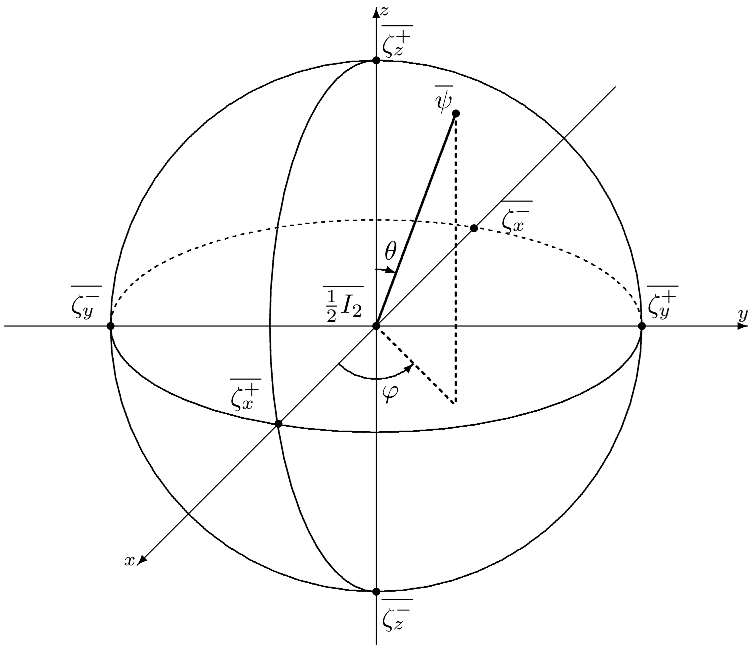


FIG. 2.1. *Bloch ball representation of a pure state.*

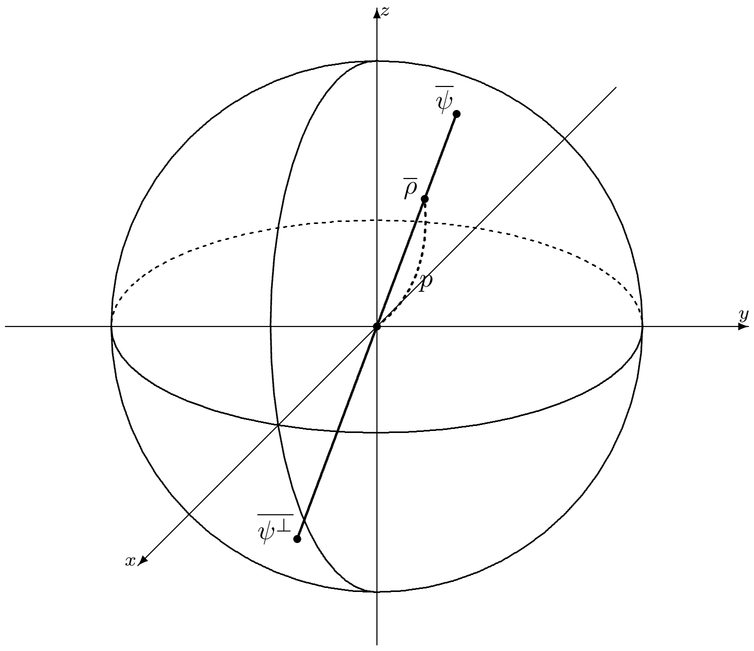


FIG. 2.2. *Bloch ball representation of a density matrix.*

points $\overline{\psi}$ and $\overline{\psi^\perp}$. Note that for $\theta = 0$ the CPSO $\boldsymbol{R}_{\alpha,0,\varphi}$ does not depend on φ. We will denote this phase rotation by $\boldsymbol{R}_\alpha$.

The affine transformation in $\mathcal{B}$ which corresponds to the von Neumann measurement in the computational basis is the orthogonal projection to the z-axis. Therefore it maps $\overline{\rho} = (x, y, z)$ into $(0, 0, z)$, the point which corresponds to the density matrix $\frac{1+z}{2}|0\rangle\langle 0| + \frac{1-z}{2}|1\rangle\langle 1|$. Thus $\Pr^0[\rho] = \frac{1+z}{2}$.

2.5. Norm and distance. Let $N = 2^n$. We will consider the *trace norm* on $\mathbb{C}^{N\times N}$ which is defined as follows: For all $V \in \mathbb{C}^{N\times N}$, $\|V\|_1 = \mathrm{Tr}\sqrt{V^\dagger V}$. This norm has several advantages when we consider the difference of density matrices. Given a von Neumann measurement, a density matrix induces a probability distribution over the basis of the measurement. The trace norm of the difference of two density matrices is the maximal variation distance between the two induced probability distributions, over all von Neumann measurements. It also satisfies the following properties.

FACT 2.3. *For all density matrices $\rho(p, \alpha)$ and $\rho(q, \beta)$ for one qubit, we have*

$$\begin{aligned}\|\rho(p,\alpha) - \rho(q,\beta)\|_1 &= \|\overline{\rho(p,\alpha)} - \overline{\rho(q,\beta)}\| \\ &= 2\sqrt{(p-q)^2 + |\alpha - \beta|^2}.\end{aligned}$$

FACT 2.4. *For all $V \in \mathbb{C}^{N\times N}$ and $W \in \mathbb{C}^{M\times M}$, we have $\|V \otimes W\|_1 = \|V\|_1\|W\|_1$ and $\sqrt{\mathrm{Tr}(V^\dagger V)} \leq \|V\|_1$. For density matrices ρ it holds that $\|\rho\|_1 = 1$.*

For n-qubit superoperators, the superoperator norm associated to the trace norm is defined as

$$\|\boldsymbol{G}\|_\infty = \sup\{\|\boldsymbol{G}(V)\|_1 : \|V\|_1 = 1\}.$$

This norm is always 1 when $\boldsymbol{G}$ is a CPSO (see, e.g., [3, Lem. 12]). The norm $\| \|_\infty$ can be easily generalized for k-tuples of superoperators by $\|(\boldsymbol{G}_1, \ldots, \boldsymbol{G}_k)\|_\infty = \max(\|\boldsymbol{G}_1\|_\infty, \ldots, \|\boldsymbol{G}_k\|_\infty)$. We will denote by dist_∞ the natural distance induced by the norm $\| \|_\infty$.

For our purposes we could have considered any other norm on superoperators since our results are motivated by the testability of universal sets of gates which act on a constant number of qubits. Indeed, it is a well-known fact that in fixed dimension all of the norms are equivalent. As stated in Fact 6.2, the testability remains invariant under changing norms.

3. Properties of CPSOs.

Here we will establish the properties of CPSOs that we will need for the characterization of our CPSO families. The first lemma does not use the complete positivity; thus, it is stated in general for PSOs for one qubit. Note that in the Bloch ball formalism PSOs for one qubit are exactly affine maps that preserve $\mathcal{B}$.

LEMMA 3.1. *Let $\boldsymbol{G}$ be a PSO for one qubit, and let ρ and τ be density matrices for one qubit.*

(a) $\|\boldsymbol{G}(\rho) - \boldsymbol{G}(\tau)\|_1 \leq \|\rho - \tau\|_1$.

(b) *If $\boldsymbol{G}$ is not constant and $\boldsymbol{G}(\rho)$ is a pure state, then ρ is a pure state.*

The first property is clear when $\boldsymbol{G}$ is a CPSO since $\|\boldsymbol{G}\|_\infty = 1$ and $\boldsymbol{G}$ is linear. Moreover, the second property does not hold for PSOs (and even for CPSOs) that act on more than one qubit. For example, the CPSO on two qubits that is the identity on the first qubit and constant to some pure state on the second qubit is a counterexample (take, for instance, $\rho = \psi \otimes (\frac{1}{2}I_2)$, where ψ is any pure state).

Proof. We prove the lemma using the Bloch ball formalism.

(a) Let $\overline{\rho}, \overline{\tau} \in \mathcal{B}$ be two distinct elements. Let $\overline{\boldsymbol{L}}$ and $\overline{u}$ be, respectively, the linear part and the constant part of the affine map $\overline{\boldsymbol{G}}$; that is, $\overline{\boldsymbol{G}} = \overline{\boldsymbol{L}} + \overline{u}$. Then we have

$$\overline{\boldsymbol{G}}(\overline{\rho}) - \overline{\boldsymbol{G}}(\overline{\tau}) = \|\overline{\rho} - \overline{\tau}\|\overline{\boldsymbol{L}}\left(\frac{\overline{\rho} - \overline{\tau}}{\|\overline{\rho} - \overline{\tau}\|}\right).$$

Note that $\overline{v} = \frac{\overline{\rho}-\overline{\tau}}{\|\overline{\rho}-\overline{\tau}\|}$ has norm 1. To conclude the proof, we now show that $\|\overline{\boldsymbol{L}}(\overline{v})\| \leq 1$. Observe that

$$\begin{aligned}\|\overline{\boldsymbol{G}}(\overline{v})\|^2 + \|\overline{\boldsymbol{G}}(-\overline{v})\|^2 &= \|\overline{\boldsymbol{L}}(\overline{v}) + \overline{u}\|^2 + \|-\overline{\boldsymbol{L}}(\overline{v}) + \overline{u}\|^2\\ &= 2(\|\overline{\boldsymbol{L}}(\overline{v})\|^2 + \|\overline{u}\|^2).\end{aligned}$$

Since $\overline{\boldsymbol{G}}$ preserves $\mathcal{B}$ and $\pm\overline{v} \in \mathcal{B}$, the images $\overline{\boldsymbol{G}}(\pm\overline{v})$ are also in $\mathcal{B}$. Therefore $\|\overline{\boldsymbol{G}}(\pm\overline{v})\| \leq 1$, and then $\|\overline{\boldsymbol{L}}(\overline{v})\| \leq 1$.

(b) We prove the second property by contradiction. Let us recall that $\mathcal{S}$ denotes the Bloch sphere. Suppose that there exists $\overline{\rho} \in \mathcal{B} - \mathcal{S}$ such that $\overline{\boldsymbol{G}}(\overline{\rho}) \in \mathcal{S}$. Since $\overline{\boldsymbol{G}}$ is not constant, there exists an element $\overline{\tau} \in \mathcal{B}$ such that $\overline{\boldsymbol{G}}(\overline{\tau}) \neq \overline{\boldsymbol{G}}(\overline{\rho})$. For every real $\varepsilon > 0$, let $\overline{w}_\varepsilon = \overline{\rho} + \varepsilon(\overline{\rho} - \overline{\tau})$. Fix some $\varepsilon > 0$ such that $\overline{w}_\varepsilon \in \mathcal{B}$. Such an ε exists since, by hypothesis, $\overline{\rho} \in \mathcal{B} - \mathcal{S}$. Moreover $\overline{\boldsymbol{G}}$ is affine; thus

$$\overline{\boldsymbol{G}}(\overline{w}_\varepsilon) = \overline{\boldsymbol{G}}(\overline{\rho}) + \varepsilon(\overline{\boldsymbol{G}}(\overline{\rho}) - \overline{\boldsymbol{G}}(\overline{\tau})).$$

Therefore, using $\|\overline{\boldsymbol{G}}(\rho)\| = 1$, the norm of $\overline{\boldsymbol{G}}(\overline{w}_\varepsilon)$ satisfies

$$\begin{aligned}\|\overline{\boldsymbol{G}}(\overline{w}_\varepsilon)\|^2 &= 1 + 2\varepsilon\big((\overline{\boldsymbol{G}}(\overline{\rho}) - \overline{\boldsymbol{G}}(\overline{\tau})), \overline{\boldsymbol{G}}(\overline{\rho})\big) + \varepsilon^2\|\overline{\boldsymbol{G}}(\overline{\rho}) - \overline{\boldsymbol{G}}(\overline{\tau})\|^2\\ &= 1 + 2\varepsilon\Big(1 - \big(\overline{\boldsymbol{G}}(\overline{\tau}), \overline{\boldsymbol{G}}(\overline{\rho})\big)\Big) + \varepsilon^2\|\overline{\boldsymbol{G}}(\overline{\rho}) - \overline{\boldsymbol{G}}(\overline{\tau})\|^2\\ &\geq 1 + \varepsilon^2\|\overline{\boldsymbol{G}}(\overline{\rho}) - \overline{\boldsymbol{G}}(\overline{\tau})\|^2\\ &> 1.\end{aligned}$$

Therefore there exists some element $\overline{w}_\varepsilon \in \mathcal{B}$ such that $\overline{\boldsymbol{G}}(\overline{w}_\varepsilon) \notin \mathcal{B}$, which contradicts $\overline{\boldsymbol{G}}(\mathcal{B}) \subseteq \mathcal{B}$. □

An affine transformation of $\mathbb{R}^3$ is uniquely defined by the images of four noncoplanar points. Surprisingly, if the transformation is a CPSO for one qubit, the images of three points are sometimes sufficient. The following will make this precise more generally for n qubits.

LEMMA 3.2. *Let $n \geq 1$ be an integer, and let ρ_1, ρ_2, and ρ_3 be three distinct one-qubit density matrices representing pure states, such that the plane in $\mathbb{R}^3$ containing the points $\overline{\rho_1}, \overline{\rho_2}, \overline{\rho_3}$ goes through the center of $\mathcal{B}$. If $\boldsymbol{G}$ is a CPSO for n qubits that acts as the identity on the set $\{\rho_1, \rho_2, \rho_3\}^{\otimes n}$, then $\boldsymbol{G}$ is the identity mapping.*

Proof. Let P be the plane defined in $\mathbb{R}^3$ by $\overline{\rho_1}$, $\overline{\rho_2}$, and $\overline{\rho_3}$. To simplify the discussion, we suppose without loss of generality (w.l.o.g.) that $\overline{\zeta_z^\pm}$ and $\overline{\zeta_x^\pm}$ are in P. Every one-qubit ρ satisfying $\overline{\rho} \in P$ is a linear combination of ρ_1, ρ_2 and ρ_3. Therefore by linearity of $\boldsymbol{G}$ we get that it acts as the identity on $\{\rho : \overline{\rho} \in P\}^{\otimes n}$. Moreover it is sufficient to show that $\boldsymbol{G}$ is the identity on density matrices representing nonentangled pure states, since they form a basis for all density matrices. To see this fact, note that any 2×2 complex matrix can be expressed as a linear combination of pure state density matrices. For example, the elementary matrix $|0\rangle\langle 1|$ can be written as

$$2|0\rangle\langle 1| = (|0\rangle + |1\rangle)(\langle 0| + \langle 1|) + i(|0\rangle + i|1\rangle)(\langle 0| - i\langle 1|) - (1+i)|0\rangle\langle 0| - (1+i)|1\rangle\langle 1|.$$

Thus any tensor product of 2×2 matrices can be expanded as a linear combination of the tensor product of single qubit pure state density matrices. Since a $2^n \times 2^n$ density matrix can be written as a linear combination of tensor products of 2×2 matrices (see, e.g., section 3.1 of [29]), it follows that any such density matrix can be expressed as a linear combination of the density matrices representing nonentangled pure states.

Using the fact that $\boldsymbol{G}$ is the identity on both the computational basis and the diagonal basis, that is, on $\{\zeta_x^\pm,\zeta_z^\pm\}^{\otimes n}$, we would like to derive that $\boldsymbol{G}$ acts as the identity everywhere. One way of proving this is to use the correspondence between unitary transformations and CPSOs. Let A be a unitary matrix such that $\boldsymbol{G}(\rho)=\mathrm{Tr}_2(A(\rho\otimes I_{2^n})A^\dagger)$ for every n-qubit state ρ (recall that Tr_2 denotes the trace out over half of the last qubits). By assumption, for every n-qubit pure state $|\psi\rangle\in\{\zeta_x^\pm,\zeta_z^\pm\}^{\otimes n}$, there exists a n-qubit pure state $|\varphi_\psi\rangle$ such that $A|\psi\rangle|0^n\rangle=|\psi\rangle|\varphi_\psi\rangle$. Therefore, by the linearity of A, we get that $|\varphi_\psi\rangle$ does not depend on $|\psi\rangle$, which implies by again the linearity of A that for every n-qubit pure state $|\psi\rangle$, $A|\psi\rangle|0^n\rangle=|\psi\rangle|\varphi\rangle$ for some n-qubit pure state $|\varphi\rangle$. Then we directly conclude that $\boldsymbol{G}$ is the identity.

For the sake of completeness, we now prove the result in more detail by induction using our first definition of CPSOs. For this, for every k, let E_k be the set of density matrices representing k-qubit nonentangled pure states, and let $F_k=\{\zeta_x^\pm,\zeta_z^\pm\}^{\otimes k}$. We will show by induction on k that, for every $0\le k\le n$, the CPSO $\boldsymbol{G}$ acts as the identity on $E_k\otimes F_{n-k}$. The case $k=0$ follows by the hypothesis of the lemma.

Suppose the statement is true for some k. Fix $\sigma\in E_k$ and $\tau\in F_{n-k-1}$. For every one-qubit density matrix ρ, let $\tilde\rho$ denote the n-qubit density matrix $\sigma\otimes\rho\otimes\tau$.

We now prove that $G(\tilde\rho)=\tilde\rho$ for every $\rho\in E_1$. For this, we use the fact that the density matrix Ψ^+ representing the entangled EPR state $(|00\rangle+|11\rangle)/\sqrt{2}$ can be written in terms of tensor products of the ζ states:

$$\Psi^+=\tfrac12(\zeta_x^+\otimes\zeta_x^++\zeta_x^-\otimes\zeta_x^-+\zeta_z^+\otimes\zeta_z^++\zeta_z^-\otimes\zeta_z^-)-\tfrac12(\zeta_y^+\otimes\zeta_y^++\zeta_y^-\otimes\zeta_y^-).$$

This can be generalized for the pure state $|\mu\rangle=(|\tilde0\rangle|\tilde0\rangle+|\tilde1\rangle|\tilde1\rangle)/\sqrt{2}$:

$$\mu=\tfrac12(\tilde\zeta_x^+\otimes\tilde\zeta_x^++\tilde\zeta_x^-\otimes\tilde\zeta_x^-+\tilde\zeta_z^+\otimes\tilde\zeta_z^++\tilde\zeta_z^-\otimes\tilde\zeta_z^-)-\tfrac12(\tilde\zeta_y^+\otimes\tilde\zeta_y^++\tilde\zeta_y^-\otimes\tilde\zeta_y^-).$$

If we apply the CPSO $\boldsymbol{I}_{2^n}\otimes\boldsymbol{G}$ to the state μ, we get

$$\begin{aligned}&(\boldsymbol{I}_{2^n}\otimes\boldsymbol{G})(\mu)\\&=\tfrac12[\tilde\zeta_x^+\otimes\tilde\zeta_x^++\tilde\zeta_x^-\otimes\tilde\zeta_x^-+\tilde\zeta_z^+\otimes\tilde\zeta_z^++\tilde\zeta_z^-\otimes\tilde\zeta_z^--\tilde\zeta_y^+\otimes\boldsymbol{G}(\tilde\zeta_y^+)-\tilde\zeta_y^-\otimes\boldsymbol{G}(\tilde\zeta_y^-)].\end{aligned}$$

If $|\varphi\rangle$ and $|\varphi'\rangle$ are orthogonal n-qubit pure states, then let $\Phi^-_{\varphi\varphi'}=(|\varphi\rangle|\varphi'\rangle-|\varphi'\rangle|\varphi\rangle)/\sqrt{2}$. Since $\Phi^-_{\varphi\varphi'}$ is orthogonal to all symmetric $2n$-qubit pure states of the form $\psi\otimes\psi$, by projecting $(\boldsymbol{I}_{2^n}\otimes\boldsymbol{G})(\mu)$ to $\Phi^-_{\varphi\varphi'}$ we obtain

$$\langle\Phi^-_{\varphi\varphi'}|(\boldsymbol{I}_{2^n}\otimes\boldsymbol{G})(\mu)|\Phi^-_{\varphi\varphi'}\rangle=-\tfrac12\langle\Phi^-_{\varphi\varphi'}|\tilde\zeta_y^+\otimes\boldsymbol{G}(\tilde\zeta_y^+)|\Phi^-_{\varphi\varphi'}\rangle-\tfrac12\langle\Phi^-_{\varphi\varphi'}|\tilde\zeta_y^-\otimes\boldsymbol{G}(\tilde\zeta_y^-)|\Phi^-_{\varphi\varphi'}\rangle.$$

Since $\boldsymbol{G}$ is a CPSO, the left-hand side of this equality is nonnegative, and in the right-hand side both terms are nonpositive. Therefore for every orthogonal n-qubit pure states $|\varphi\rangle$ and $|\varphi'\rangle$, we get

$$\langle\Phi^-_{\varphi\varphi'}|\tilde\zeta_y^+\otimes\boldsymbol{G}(\tilde{\zeta_y}^+)|\Phi^-_{\varphi\varphi'}\rangle=\langle\Phi^-_{\varphi\varphi'}|\tilde\zeta_y^-\otimes\boldsymbol{G}(\tilde{\zeta_y}^-)|\Phi^-_{\varphi\varphi'}\rangle=0.$$

A straightforward calculation then shows that $\boldsymbol{G}(\tilde{\zeta_y}^\pm)=\tilde\zeta_y^\pm$. Therefore $\boldsymbol{G}$ acts as the identity on density matrices $\tilde\zeta_z^\pm$, $\tilde\zeta_x^+$, and $\tilde\zeta_y^+$, which generate all density matrices, and thus $\boldsymbol{G}(\tilde\rho)=\tilde\rho$. □

We also use the property that for CPSOs unitarity and invertibility are equivalent (see, e.g., [25, Chap. 3, sect. 8]).

LEMMA 3.3. *Let $\boldsymbol{G}$ be a CPSO for n qubits. If there exists a CPSO $\boldsymbol{H}$ for n qubits such that $\boldsymbol{H} \circ \boldsymbol{G}$ is the identity mapping, then $\boldsymbol{G}$ is a unitary superoperator.*

Using the following lemma, we can give a version of Lemma 3.2 in the approximate context.

LEMMA 3.4. *Let $\boldsymbol{G}$ be a superoperator for one qubit. Let $0 \leq \varepsilon \leq 1$ be such that $\|\boldsymbol{G}(\zeta_x^\pm) - \zeta_x^\pm\|_1$, $\|\boldsymbol{G}(\zeta_y^\pm) - \zeta_y^\pm\|_1$, $\|\boldsymbol{G}(\zeta_z^\pm) - \zeta_z^\pm\|_1 \leq \varepsilon$. Then $\|\boldsymbol{G} - \boldsymbol{I}_2\|_\infty \leq \sqrt{10}\varepsilon$.*

Proof. Every 2×2 complex matrix V can be decomposed as

$$V = \begin{pmatrix} a & b \\ c & d \end{pmatrix} = a\zeta_z^+ + d\zeta_z^- + \frac{b+c}{2}\left(\zeta_x^+ - \frac{1}{2}(\zeta_z^+ + \zeta_z^-)\right) + i\frac{b-c}{2}\left(\zeta_y^+ - \frac{1}{2}(\zeta_z^+ + \zeta_z^-)\right).$$

All norms $\|\zeta_\cdot^\pm\|_1$ are 1; therefore, the hypotheses on G imply

$$\|\boldsymbol{G}(V) - V\|_1 \leq \varepsilon(|a| + 2|b| + 2|c| + |d|).$$

From Fact 2.4 we also have that $\sqrt{\mathrm{Tr}(V^\dagger V)} \leq \|V\|_1$. Moreover $\mathrm{Tr}(V^\dagger V) = |a|^2 + |b|^2 + |c|^2 + |d|^2$. Then we conclude the proof by the Cauchy–Schwarz inequality $|a| + 2|b| + 2|c| + |d| \leq \sqrt{10}\sqrt{|a|^2 + |b|^2 + |c|^2 + |d|^2}$. □

LEMMA 3.5. *Let $\overline{u}$ and $\overline{v}$ be two orthonormal vectors in $\mathbb{R}^3$, and let $0 \leq \varepsilon \leq 1$ be a constant. If $\boldsymbol{G}$ is a CPSO for one qubit such that $\|\overline{\boldsymbol{G}}(\pm\overline{u}) - \pm\overline{u}\| \leq \varepsilon$ and $\|\overline{\boldsymbol{G}}(\pm\overline{v}) - \pm\overline{v}\| \leq \varepsilon$, then $\|\boldsymbol{G} - \boldsymbol{I}_2\|_\infty \leq 96\varepsilon$.*

Proof. We can suppose w.l.o.g. that $u = \zeta_x^+$ and $v = \zeta_z^+$. Let $\rho = \boldsymbol{G}(\zeta_y^+)$, where $\overline{\rho} = (x, y, z)$. From Lemma 3.1 it follows that $\|\boldsymbol{G}(\zeta_z^+) - \rho\|_1 \leq \|\zeta_z^+ - \zeta_y^+\|_1 = \sqrt{2}$. By the assumption of this lemma we have that $\|\boldsymbol{G}(\zeta_z^+) - \zeta_z^+\|_1 \leq \varepsilon$, and hence $\|\zeta_z^+ - \rho\|_1 \leq \sqrt{2} + \varepsilon$. The same relation holds also for the other three fixed points ζ_z^-, ζ_x^+, and ζ_x^-. As a result, the three coordinates of $\overline{\rho}$ have to obey the four inequalities

$$\begin{aligned} & x^2 + y^2 + (z \pm 1)^2 \leq (\sqrt{2} + \varepsilon)^2 \leq 2 + 4\varepsilon \\ \text{and} \quad & (x \pm 1)^2 + y^2 + z^2 \leq (\sqrt{2} + \varepsilon)^2 \leq 2 + 4\varepsilon. \end{aligned} \tag{3.1}$$

A second set of restrictions on (x, y, z) comes from the complete positivity of $\boldsymbol{G}$. Again we use the decomposition of the EPR state Ψ^+ to analyze the two-qubit state:

$$\begin{aligned} (\boldsymbol{I}_2 \otimes \boldsymbol{G})(\Psi^+) = {} & \tfrac{1}{2}(\zeta_x^+ \otimes \boldsymbol{G}(\zeta_x^+) + \zeta_x^- \otimes \boldsymbol{G}(\zeta_x^-)) \\ & + \tfrac{1}{2}(\zeta_z^+ \otimes \boldsymbol{G}(\zeta_z^+) + \zeta_z^- \otimes \boldsymbol{G}(\zeta_z^-)) \\ & - \tfrac{1}{2}(\zeta_y^+ \otimes \boldsymbol{G}(\zeta_y^+) + \zeta_y^- \otimes \boldsymbol{G}(\zeta_y^-)). \end{aligned}$$

Using the hypothesis, the projection of this state onto the antisymmetrical entangled qubit pair $|\Phi^-\rangle = (|01\rangle - |10\rangle)/\sqrt{2}$ yields

$$\langle\Phi^-|(\boldsymbol{I}_2 \otimes \boldsymbol{G})(\Psi^+)|\Phi^-\rangle \leq 2\varepsilon - \tfrac{1}{2}\langle\Phi^-|\zeta_y^+ \otimes \boldsymbol{G}(\zeta_y^+)|\Phi^-\rangle - \tfrac{1}{2}\langle\Phi^-|\zeta_y^- \otimes \boldsymbol{G}(\zeta_y^-)|\Phi^-\rangle.$$

Since $\boldsymbol{G}$ is a CPSO, as in Lemma 3.2 we get $\langle\Phi^-|\zeta_y^+ \otimes \rho|\Phi^-\rangle \leq 4\varepsilon$. A straightforward calculation shows that this last relation is equivalent with a restriction on the y coordinate: $y \geq 1 - 16\varepsilon$.

This last inequality implies $y^2 \geq 1 - 32\varepsilon$, which combined with the restrictions of (3.1) leads to the conclusion that $(x \pm 1)^2 \leq 2 + 4\varepsilon - y^2 - z^2 \leq 1 + 36\varepsilon$, and similarly $(z \pm 1)^2 \leq 1 + 36\varepsilon$. The x and z coordinates of $\overline{\rho}$ satisfy $|x|, |z| \leq 18\varepsilon$.

These bounds imply

$$\|\boldsymbol{G}(\zeta_y^+) - \zeta_y^+\|_1 = \sqrt{x^2 + (y-1)^2 + z^2} \leq \sqrt{904}\varepsilon.$$

The same result can be proved for ζ_y^-. Therefore by Lemma 3.4 we can conclude the proof. □

4. Characterization.

4.1. One-qubit CPSO families.

In this section, every CPSO will be for one qubit. First we define the notion of experimental equations, and then we show that several important CPSO families are characterizable by them.

An *experimental equation* in one variable is a CPSO equation of the form

$$\mathrm{Pr}^0[\boldsymbol{G}^k(|b\rangle\langle b|)] = r, \tag{4.1}$$

where k is a nonnegative integer, $b \in \{0,1\}$, and $0 \le r \le 1$. We will call the left-hand side of the equation the *probability term* and the right-hand side the *constant term.* The *size* of this equation is k. A CPSO $\boldsymbol{G}$ will "almost" satisfy the equations if, for example, it is the result of adding small systematic and random errors (independent of time) to a CPSO that does satisfy them. For $\varepsilon \ge 0$, the CPSO $\boldsymbol{G}$ *ε-satisfies* (4.1) if $|\mathrm{Pr}^0[\boldsymbol{G}^k(|b\rangle\langle b|)] - r| \le \varepsilon$, and when $\varepsilon = 0$ we will just say that $\boldsymbol{G}$ *satisfies* (4.1). Let (E) be a finite set of experimental equations. If $\boldsymbol{G}$ ε-satisfies all equations in (E), we say that $\boldsymbol{G}$ ε-satisfies (E). If some $\boldsymbol{G}$ satisfies (E), then (E) is *satisfiable.* The set $\{\boldsymbol{G} : \boldsymbol{G} \text{ satisfies } (E)\}$ will be denoted by $\mathcal{F}_{(E)}$. A family $\mathcal{F}$ of CPSOs is *characterizable* if it is $\mathcal{F}_{(E)}$ for some finite set (E) of experimental equations. In this case we say that (E) *characterizes* $\mathcal{F}$.

All of these definitions generalize naturally for m-tuples of CPSOs for $m \ge 2$. In what follows we will need only the case $m = 2$. An *experimental equation* in two CPSO variables is an equation of the form

$$\mathrm{Pr}^0[\boldsymbol{F}^{k_1} \circ \boldsymbol{G}^{l_1} \circ \cdots \circ \boldsymbol{F}^{k_t} \circ \boldsymbol{G}^{l_t}(|b\rangle\langle b|)] = r, \tag{4.2}$$

where $k_1, \ldots, k_t, l_1, \ldots, l_t$ are nonnegative integers, $b \in \{0,1\}$, and $0 \le r \le 1$.

We discuss now the existence of finite sets of experimental equations in one variable that characterize unitary superoperators, that is, the operators $\boldsymbol{R}_{\alpha,\theta,\varphi}$, for $\alpha \in (-\pi,\pi]$, $\theta \in [0,\pi/2]$, and $\varphi \in [0,2\pi)$. First observe that, due to the restrictions of experimental equations, there are unitary superoperators that they cannot distinguish.

FACT 4.1. *Let $\alpha \in [0,\pi]$, $\theta \in [0,\pi/2]$, and $\varphi_1, \varphi_2 \in [0,2\pi)$ such that $\varphi_1 \ne \varphi_2$. Let (E) be a finite set of experimental equations in m variables. If $(\boldsymbol{R}_{\alpha,\theta,\varphi_1}, \boldsymbol{G}_2, \ldots, \boldsymbol{G}_m)$ satisfies (E), then there exist $\boldsymbol{G}'_2, \ldots, \boldsymbol{G}'_m$ and $\boldsymbol{G}''_2, \ldots, \boldsymbol{G}''_m$ such that $(\boldsymbol{R}_{-\alpha,\theta,\varphi_1}, \boldsymbol{G}'_2, \ldots, \boldsymbol{G}'_m)$ and $(\boldsymbol{R}_{\alpha,\theta,\varphi_2}, \boldsymbol{G}''_2, \ldots, \boldsymbol{G}''_m)$ both satisfy (E).*

In the Bloch ball formalism this corresponds to the following degrees of freedom in the choice of the orthonormal basis of $\mathbb{R}^3$. Since experimental equations contain exactly the states $|0\rangle\langle 0|$ and $|1\rangle\langle 1|$, there is no freedom in the choice of the z-axis, but there is complete freedom in the choice of the x and y axes. The indistinguishability of the latitude φ corresponds to the freedom of choosing the oriented x-axis, and the indistinguishability of the sign of α corresponds to the freedom of choosing the orientation of the y-axis.

We introduce the following notations. Let $\mathcal{R}_{\alpha,\theta}$ denote the superoperator family $\{\boldsymbol{R}_{\pm\alpha,\theta,\varphi} : \varphi \in [0,2\pi)\}$. For $\varphi \in [0,2\pi)$, let the NOT_φ transformation be defined by $\mathrm{NOT}_\varphi|0\rangle = e^{i\varphi}|1\rangle$ and $\mathrm{NOT}_\varphi(e^{i\varphi}|1\rangle) = |0\rangle$, and recall that the Hadamard transformation H_φ obeys $H_\varphi|0\rangle = (|0\rangle + e^{i\varphi}|1\rangle)/\sqrt{2}$ and $H_\varphi(e^{i\varphi}|1\rangle) = (|0\rangle - e^{i\varphi}|1\rangle)/\sqrt{2}$. Observe that $\boldsymbol{H}_\varphi = \boldsymbol{R}_{\pi,\pi/4,\varphi}$ and $\mathbf{NOT}_\varphi = \boldsymbol{R}_{\pi,\pi/2,\varphi}$ for $\varphi \in [0,2\pi)$. Finally let $\mathcal{H} = \{\boldsymbol{H}_\varphi : \varphi \in [0,2\pi)\}$, and $\mathcal{N} = \{\mathbf{NOT}_\varphi : \varphi \in [0,2\pi)\}$.

Since the sign of α cannot be determined, we will assume that α is in the interval $[0,\pi]$. We will also consider only unitary superoperators such that α/π is rational. This is a reasonable choice, since these superoperators form a dense subset of all unitary superoperators. For such a unitary superoperator, let n_α be the smallest positive integer n for which $n\alpha = 0 \mod 2\pi$. Then either $n_\alpha = 1$ or $n_\alpha \geq 2$, and there exists $t \geq 1$ which is coprime with n_α such that $\alpha = (t/n_\alpha)2\pi$. Observe that the case $n_\alpha = 1$ corresponds to the identity superoperator.

Our first theorem shows that almost all families $\mathcal{R}_{\alpha,\theta}$ are characterizable by some finite set of experimental equations.

Theorem 4.2. *Let $(\alpha,\theta) \in (0,\pi] \times (0,\pi/2]\backslash\{(\pi,\pi/2)\}$ be such that α/π is rational. Let $z_k(\alpha,\theta) = \cos^2\theta + \sin^2\theta\cos(k\alpha)$. Then the following experimental equations characterize $\mathcal{R}_{\alpha,\theta}$:*

$$\Pr{}^0[\boldsymbol{G}^{n_\alpha}(|1\rangle\langle 1|)] = 0, \tag{4.3}$$

$$\Pr{}^0[\boldsymbol{G}^{k}(|0\rangle\langle 0|)] = (1+z_k(\alpha,\theta))/2,\ k \in \{1,2,\ldots,n_\alpha\}. \tag{4.4}$$

In particular, since $\mathcal{H} = \mathcal{R}_{\pi,\pi/4}$, the family $\mathcal{H}$ is characterized by

$$\begin{aligned}&\Pr{}^0[\boldsymbol{G}^2(|1\rangle\langle 1|)] = 0, \quad \Pr{}^0[\boldsymbol{G}^2(|0\rangle\langle 0|)] = 1,\\ &\Pr{}^0[\boldsymbol{G}(|0\rangle\langle 0|)] = 1/2.\end{aligned}$$

Proof. First observe that every CPSO in $\mathcal{R}_{\alpha,\theta}$ satisfies the experimental equations of the theorem since the z-coordinate of $\overline{\boldsymbol{R}^k_{\alpha,\theta,\varphi}(|0\rangle\langle 0|)}$ is $z_k(\alpha,\theta)$ for every $\varphi \in [0,2\pi)$. Let $\boldsymbol{G}$ be a CPSO that satisfies these equations. We will prove that $\boldsymbol{G}$ is a unitary superoperator. Then, Fact 4.3 implies that $\boldsymbol{G} \in \mathcal{R}_{\alpha,\theta}$.

Since $z_1(\alpha,\theta) \neq \pm 1$, $\boldsymbol{G}(|0\rangle\langle 0|) \notin \{|0\rangle\langle 0|, |1\rangle\langle 1|\}$. Observing that $\boldsymbol{G}^{n_\alpha}(|0\rangle\langle 0|) = |0\rangle\langle 0|$, Lemma 3.1(b) implies that $\boldsymbol{G}(|0\rangle\langle 0|)$ is a pure state. Thus $|0\rangle\langle 0|$, $|1\rangle\langle 1|$, and $\boldsymbol{G}(|0\rangle\langle 0|)$ are distinct pure states, and since $\boldsymbol{G}^{n_\alpha}$ acts as the identity on them, by Lemma 3.2 it is the identity mapping. Hence, by Lemma 3.3, $\boldsymbol{G}$ is a unitary superoperator. □

Fact 4.3. *Let $\alpha \in (0,\pi]$, $\theta \in (0,\pi/2]$, $\alpha' \in (-\pi,\pi]$, $\theta' \in (0,\pi/2]$ be such that α/π is rational. If $z_k(\alpha,\theta) = z_k(\alpha',\theta')$ for $k \in \{1,2,\ldots,n_\alpha\}$, then $|\alpha'| = \alpha$ and $\theta' = \theta$.*

The remaining families $\mathcal{R}_{\alpha,\theta}$ for which α/π is rational are $\{\boldsymbol{R}_{-\alpha},\boldsymbol{R}_\alpha\}$, for $\alpha \in [0,\pi]$, and $\mathcal{N}$. Let us recall that $\boldsymbol{M}$ is the CPSO that represents the von Neumann measurement in the computational basis. Since $\boldsymbol{M}$ satisfies exactly the same equations as $\boldsymbol{R}_{\pm\alpha}$, and $\mathbf{NOT}_0 \circ \boldsymbol{M}$ satisfies exactly the same equations as $\mathbf{NOT}_\varphi$, for any $\varphi \in [0,2\pi)$, these families are not characterizable by experimental equations in one variable. Nevertheless it turns out that together with the family $\mathcal{H}$ they become characterizable. This is stated in the following theorem.

Theorem 4.4. *The family $\{(\boldsymbol{H}_\varphi,\mathbf{NOT}_\varphi) : \varphi \in [0,2\pi)\} \subset \mathcal{H}\times\mathcal{N}$ is characterized by the experimental equations in two variables $(\boldsymbol{F},\boldsymbol{G})$:*

$$\begin{aligned}&\Pr{}^0[\boldsymbol{F}(|0\rangle\langle 0|)] = 1/2, \quad \Pr{}^0[\boldsymbol{F}^2(|0\rangle\langle 0|)] = 1, \quad \Pr{}^0[\boldsymbol{F}^2(|1\rangle\langle 1|)] = 0,\\ &\Pr{}^0[\boldsymbol{G}(|0\rangle\langle 0|)] = 0, \qquad \Pr{}^0[\boldsymbol{G}(|1\rangle\langle 1|)] = 1,\\ &\Pr{}^0[\boldsymbol{F}\circ\boldsymbol{G}^2\circ\boldsymbol{F}(|0\rangle\langle 0|)] = 1, \quad \Pr{}^0[\boldsymbol{F}\circ\boldsymbol{G}\circ\boldsymbol{F}(|0\rangle\langle 0|)] = 1.\end{aligned}$$

If α/π is rational, then the family $\mathcal{H}\times\{\boldsymbol{R}_{\pm\alpha}\}$ is characterized by the experimental *equations in two variables* $(\boldsymbol{F}, \boldsymbol{G})$*:*

$$\begin{array}{lll} \Pr^0[\boldsymbol{F}(|0\rangle\langle 0|)] = 1/2, & \Pr^0[\boldsymbol{F}^2(|0\rangle\langle 0|)] = 1, & \Pr^0[\boldsymbol{F}^2(|1\rangle\langle 1|)] = 0, \\ \Pr^0[\boldsymbol{G}(|0\rangle\langle 0|)] = 1, & \Pr^0[\boldsymbol{G}(|1\rangle\langle 1|)] = 0, & \\ \Pr^0[\boldsymbol{F} \circ \boldsymbol{G}^{n_\alpha} \circ \boldsymbol{F}(|0\rangle\langle 0|)] = 1, & \Pr^0[\boldsymbol{F} \circ \boldsymbol{G} \circ \boldsymbol{F}(|0\rangle\langle 0|)] = (1 + \cos\alpha)/2. & \end{array}$$

In particular, since $\boldsymbol{I}_2 = \boldsymbol{R}_0$, the identity transformation on 1 qubit is characterizable; namely, the family $\mathcal{H} \times \{\boldsymbol{I}_2\}$ is characterized by

$$\begin{array}{lll} \Pr^0[\boldsymbol{F}(|0\rangle\langle 0|)] = 1/2, & \Pr^0[\boldsymbol{F}^2(|0\rangle\langle 0|)] = 1, & \Pr^0[\boldsymbol{F}^2(|1\rangle\langle 1|)] = 0, \\ \Pr^0[\boldsymbol{G}(|0\rangle\langle 0|)] = 1, & \Pr^0[\boldsymbol{G}(|1\rangle\langle 1|)] = 0, & \\ \Pr^0[\boldsymbol{F} \circ \boldsymbol{G} \circ \boldsymbol{F}(|0\rangle\langle 0|)] = 1. & & \end{array}$$

Proof. Let us consider the first characterization. Observe that every couple $(\boldsymbol{F}, \boldsymbol{G})$ of $\{(\boldsymbol{H}_\varphi, \mathbf{NOT}_\varphi) : \varphi \in [0, 2\pi)\}$ satisfies the system of experimental equations.

Let now $\boldsymbol{F}$ and $\boldsymbol{G}$ be two CPSOs that satisfy the system. The CPSO $\boldsymbol{F}$ satisfies also the system in (4.3) for $\alpha = \pi$ and $\theta = \pi/4$; thus, from Theorem 4.2 there exists $0 \leq \varphi < 2\pi$ such that $\boldsymbol{F} = \boldsymbol{H}_\varphi$. By hypothesis, $\boldsymbol{G}^2$ acts as the identity on $|0\rangle\langle 0|$ and $|1\rangle\langle 1|$. Moreover $\boldsymbol{H}_\varphi \circ \boldsymbol{G}^2 \circ \boldsymbol{H}_\varphi(|0\rangle\langle 0|) = |0\rangle\langle 0|$. Let us apply $\boldsymbol{H}_\varphi$ on both sides of the previous equality. Since $\boldsymbol{H}_\varphi^2 = \boldsymbol{I}_2$, the CPSO $\boldsymbol{G}^2$ acts also as the identity on $\boldsymbol{H}_\varphi(|0\rangle\langle 0|)$. Therefore using Lemma 3.2 we get that $\boldsymbol{G}^2$ is the identity; then, by Lemma 3.3, $\boldsymbol{G}$ is a unitary CPSO. Since $\overline{|0\rangle\langle 0|}$ and $\overline{|1\rangle\langle 1|}$ are exchanged together under the action of $\overline{\boldsymbol{G}}$, the rotation axis of $\overline{\boldsymbol{G}}$ is necessarily in the plane with equation $z = 0$. Moreover this axis goes through $\overline{\boldsymbol{H}_\varphi(|0\rangle\langle 0|)}$ because from the last experimental equation $\boldsymbol{G}$ acts as the identity on $\boldsymbol{H}_\varphi(|0\rangle\langle 0|)$. We conclude that the CPSO $\boldsymbol{G}$ is $\mathbf{NOT}_\varphi$.

We now consider the second characterization. The system is clearly satisfied by every pair $(\boldsymbol{F}, \boldsymbol{G})$ in $\mathcal{H} \times \{\boldsymbol{R}_{\pm\alpha}\}$.

Let now $\boldsymbol{F}$ and $\boldsymbol{G}$ be two CPSOs that satisfy the system of experimental equations. Like in the previous characterization, there exists a real $0 \leq \varphi < 2\pi$ such that $\boldsymbol{F} = \boldsymbol{H}_\varphi$, and $\boldsymbol{G}^{n_\alpha}$ is the identity. Therefore $\boldsymbol{G}$ is a unitary CPSO. Since $\boldsymbol{G}$ acts as the identity on $|0\rangle\langle 0|$ and $|1\rangle\langle 1|$, the rotation axis of $\overline{\boldsymbol{G}}$ is the z-axis. The last experimental equation implies that the angle $\alpha' \in (-\pi, \pi]$ of the rotation $\overline{\boldsymbol{G}}$ satisfies $\cos\alpha' = \cos\alpha$; that is, $\alpha' = \pm\alpha$. □

4.2. Characterization of c-NOT gates. In this section we will extend our theory of characterization of CPSO families for several qubits. In particular, we will show that the family of **c-NOT** gates together with the family $\mathcal{H}$ is characterizable. First we need some definitions.

For every $\varphi \in [0, 2\pi)$, we define c-NOT$_\varphi$ as the only unitary transformation over $\mathbb{C}^4$ satisfying c-NOT$_\varphi(|0\rangle|\psi\rangle) = |0\rangle|\psi\rangle$ and c-NOT$_\varphi|1\rangle|\psi\rangle = |1\rangle$NOT$_\varphi|\psi\rangle$ for all $|\psi\rangle \in \mathbb{C}^2$.

We extend the definition of the experimental equation for CPSOs given in (4.2) for n qubits. When variables denote CPSOs for n qubits, it is an equation of the form

$$\Pr^v[\boldsymbol{F}^{k_1} \circ \boldsymbol{G}^{l_1} \circ \cdots \circ \boldsymbol{F}^{k_t} \circ \boldsymbol{G}^{l_t}(|w\rangle\langle w|)] = r, \tag{4.5}$$

where, in addition to the notation of (4.2), $v, w \in \{0,1\}^n$, and $\Pr^v$ is the probability of measuring $|v\rangle\langle v|$. When variables denote CPSOs for less than n qubits, we also allow both the tensor product of two CPSO variables and the tensor product of a CPSO variable with the identity. We now state the characterization.

THEOREM 4.5. *The family* $\{(\boldsymbol{H}_\varphi, \textbf{c-NOT}_\varphi) : \varphi \in [0, 2\pi)\}$ *is characterized by the experimental equations in two variables* $(\boldsymbol{F}, \boldsymbol{G})$*:*

$$\begin{aligned}
&\Pr{}^0[\boldsymbol{F}(|0\rangle\langle 0|)] = 1/2, \quad \Pr{}^0[\boldsymbol{F}^2(|0\rangle\langle 0|)] = 1, \quad \Pr{}^0[\boldsymbol{F}^2(|1\rangle\langle 1|)] = 0,\\
&\Pr{}^{00}[\boldsymbol{G}(|00\rangle\langle 00|)] = 1, \quad \Pr{}^{01}[\boldsymbol{G}(|01\rangle\langle 01|)] = 1,\\
&\Pr{}^{11}[\boldsymbol{G}(|10\rangle\langle 10|)] = 1, \quad \Pr{}^{10}[\boldsymbol{G}(|11\rangle\langle 11|)] = 1,\\
&\Pr{}^{00}[(\boldsymbol{I}_2 \otimes \boldsymbol{F}) \circ \boldsymbol{G} \circ (\boldsymbol{I}_2 \otimes \boldsymbol{F})(|00\rangle\langle 00|)] = 1,\\
&\Pr{}^{10}[(\boldsymbol{I}_2 \otimes \boldsymbol{F}) \circ \boldsymbol{G} \circ (\boldsymbol{I}_2 \otimes \boldsymbol{F})(|10\rangle\langle 10|)] = 1,\\
&\Pr{}^{00}[(\boldsymbol{F} \otimes \boldsymbol{I}_2) \circ \boldsymbol{G}^2 \circ (\boldsymbol{F} \otimes \boldsymbol{I}_2)(|00\rangle\langle 00|)] = 1,\\
&\Pr{}^{01}[(\boldsymbol{F} \otimes \boldsymbol{I}_2) \circ \boldsymbol{G}^2 \circ (\boldsymbol{F} \otimes \boldsymbol{I}_2)(|01\rangle\langle 01|)] = 1,\\
&\Pr{}^{00}[(\boldsymbol{F} \otimes \boldsymbol{F}) \circ \boldsymbol{G} \circ (\boldsymbol{F} \otimes \boldsymbol{F})(|00\rangle\langle 00|)] = 1.
\end{aligned}$$

Proof. First observe that every pair $(\boldsymbol{F}, \boldsymbol{G})$ in $\{(\boldsymbol{H}_\varphi, \textbf{c-NOT}_\varphi) : \varphi \in [0, 2\pi)\}$ satisfies the experimental equations of the theorem.

Let $\boldsymbol{F}$ and $\boldsymbol{G}$ satisfy these equations. By Theorem 4.2, with $\alpha = \pi$ and $\theta = \pi/4$, the first three equations imply that $\boldsymbol{F} = \boldsymbol{H}_\varphi$ for some $\varphi \in [0, 2\pi)$. Let $\rho = \boldsymbol{H}_\varphi(|0\rangle\langle 0|)$. The remaining equations imply that $\boldsymbol{G}^2$ acts as the identity on $\{|0\rangle\langle 0|, |1\rangle\langle 1|, \rho\}^{\otimes^2}$. Then Lemma 3.2 implies that $\boldsymbol{G}^2 = \boldsymbol{I}_4$, and it follows from Lemma 3.3 that $\boldsymbol{G}$ is a unitary CPSO.

We now show that indeed $\boldsymbol{G} = \textbf{c-NOT}_\varphi$. To simplify we will suppose that $\varphi = 0$, since one can replace $|1\rangle$ by $|1'\rangle = e^{i\varphi}|1\rangle$. Let $G \in \mathrm{U}(4)$ be a unitary transformation such that $\boldsymbol{G}$ is the corresponding CPSO. Then, since $\boldsymbol{G}$ acts as the identity on $|00\rangle\langle 00|$, there exists a real $0 \le \gamma < 2\pi$ such that $G|00\rangle = e^{i\gamma}|00\rangle$. Since $\boldsymbol{G}$ is also the corresponding CPSO of the unitary transformation $e^{-i\gamma}G$, we can suppose that $\gamma = 0$ w.l.o.g.. By hypothesis $\boldsymbol{G}$ acts as the identity on the density matrices $|01\rangle\langle 01|$ and $|0\rangle\langle 0| \otimes \rho$. Therefore the linearity of G necessarily implies that $G|01\rangle = |01\rangle$.

Using a similar argument, since $\boldsymbol{G}$ acts as $\textbf{c-NOT}_0$ on the density matrices $|10\rangle\langle 10|$, $|11\rangle\langle 11|$, and $|1\rangle\langle 1| \otimes \rho$, there exists a real $0 \le \gamma' < 2\pi$ such that $G|10\rangle = e^{i\gamma'}|11\rangle$ and $G|11\rangle = e^{i\gamma'}|10\rangle$.

Then the last experimental equation, which states that $\boldsymbol{G}$ acts as the identity on $\rho \otimes \rho$, implies

$$G(|00\rangle + |01\rangle + |10\rangle + |11\rangle) = e^{i\gamma''}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

for some $0 \le \gamma'' < 2\pi$. We now conclude the proof by observing that the linearity of G implies $\gamma' = 0$, $\gamma'' = 0$, and therefore $G = \text{c-NOT}_0$. □

5. Robustness. In this section we introduce the notion of robustness for experimental equations which will be the crucial ingredient for proving self-testability. For simplicity we will deal only with the case of experimental equations for one qubit and in one variable. From now on (E) will always denote a set of such equations. Similar results can be obtained for several qubits and several variables.

DEFINITION 5.1. *Let* $\varepsilon, \delta \ge 0$*, and let* (E) *be a finite satisfiable set of experimental equations. We say that* (E) *is* (ε, δ)-robust *if whenever a CPSO* $\boldsymbol{G}$ ε*-satisfies* (E)*, we have* $\mathrm{dist}_\infty(\boldsymbol{G}, \mathcal{F}_{(E)}) \le \delta$*.*

When a CPSO family is characterized by a finite set of experimental equations (E), one would like to prove that (E) is robust. The next theorem shows that this is always the case.

Theorem 5.2. *Let (E) be a finite satisfiable set of experimental equations. Then there exists an integer $k \geq 1$ and a real $C > 0$ such that, for all $\varepsilon \geq 0$, (E) is $(\varepsilon, C\varepsilon^{1/k})$-robust.*

The proof uses the structure of semialgebraic sets. Therefore we introduce a few notions of algebraic geometry over reals for which the reader can refer, for example, to [9]. A (real) *semialgebraic* set is a subset of $\mathbb{R}^m$ such that $X = \{x \in \mathbb{R}^m : Q(x)\}$, where Q a finite Boolean combination of expressions of type $P(x) > 0$, $P(x) < 0$, or $P(x) = 0$ for any real polynomial P. Finite unions, finite intersections, and complements of such sets remain semialgebraic sets. One of the main results on these sets is that their projections also remain semialgebraic sets. This is Tarski-Seidenberg's theorem (see, e.g., [9, Thm. 2.3.4]). A consequence of that theorem is that we can also use quantifiers $\exists y \in Y$ and $\forall y \in Y$, where Y is a semialgebraic set, for defining semialgebraic sets.

Let $X \subseteq \mathbb{R}^m$. A function $f : X \to \mathbb{R}^{m'}$ is *semialgebraic* if its graph representation is a semialgebraic set. The composition of two semialgebraic functions is also semialgebraic. Tarski-Seidenberg's theorem implies that every real function defined over $X \subseteq \mathbb{R}^m$ by $x \mapsto \inf\{f(x, y) : (x, y) \in X'\}$ (respectively, $x \mapsto \sup\{f(x, y) : (x, y) \in X'\}$), where $X' \subseteq \mathbb{R}^{m'}$ and $f : X' \to \mathbb{R}$ are semialgebraic, is also semialgebraic (see, e.g., [16, Cor. A.2.4]). In particular, the function that maps an element toward its distance to a compact semialgebraic set is a continuous semialgebraic function. Another fundamental consequence of Tarski-Seidenberg's theorem for continuous semialgebraic functions is Lojasiewicz's inequality. For a proof of the following fact, see, for example, [9, Prop. 2.3.11].

Fact 5.3 (Lojasiewicz's inequality). *Let $X \subseteq \mathbb{R}^m$ be a compact semialgebraic set. Let $f, g : X \to \mathbb{R}$ be continuous semialgebraic functions. Assume that for all $x \in X$ if $f(x) = 0$, then $g(x) = 0$. Then there exists an integer $k \geq 1$ and a real $C > 0$ such that, for all $x \in X$, $|g(x)|^k \leq C|f(x)|$.*

We can now prove Theorem 5.2, that is, the generic robustness for experimental equations.

Proof. In the proof, $\mathbb{C}$ is identified with $\mathbb{R}^2$. Then the set K of CPSOs for one qubit is a real compact semialgebraic set. Indeed we prove that, for every n, the set K_n of all CPSOs for n qubits is a real compact semialgebraic set using one of the Kraus representations for CPSOs. For that, let Tr_2 be the *partial trace* operator. Namely, Tr_2 is the unique linear map $\mathbb{C}^{N^2} \otimes \mathbb{C}^{N^2}$, where $N = 2^n$, such that, for every $i, j = 1, \ldots, N^2$ and every $V \in \mathbb{C}^{N^2}$,

$$\mathrm{Tr}_2(V \otimes |i\rangle\langle j|) = \begin{cases} V & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

Then the set K_n satisfies the following (see, e.g., [25, Chap. 3, sect. 3]):

$$K_n = \{\boldsymbol{G} : \exists A \in \mathrm{U}(N^2), \quad \forall V \in \mathbb{C}^{N \times N},\ \boldsymbol{G}(V) = \mathrm{Tr}_2(A(V \otimes I_N)A^\dagger)\}.$$

Since $\mathrm{U}(N^2)$ is a compact semialgebraic set, K_n is also a compact semialgebraic set.

Suppose now that in (E) there are d equations. Let $f : K \to \mathbb{R}$ be the function that maps the CPSO $\boldsymbol{G}$ to the maximum of the magnitudes of the difference between the probability term and the constant term of the ith equation in (E) for $i = 1, \ldots, d$. By definition of f, we get $f^{-1}(0) = \mathcal{F}_{(E)}$. Moreover, f is a continuous semialgebraic function, since it is the maximum of the magnitudes of polynomial functions in the (real) coefficients of $\boldsymbol{G}$.

Let $g : K \to \mathbb{R}$ be defined in $\boldsymbol{G}$ by $g(\boldsymbol{G}) = \mathrm{dist}_\infty(\boldsymbol{G}, \mathcal{F}_{(E)})$. Since K is a compact semialgebraic set, g is a continuous semialgebraic function. Moreover, for all $\boldsymbol{G} \in K$, we have $f(\boldsymbol{G}) = 0$ if and only if $g(\boldsymbol{G}) = 0$. Then Fact 5.3 concludes the proof. □

In some cases we can explicitly compute the constants C and k of Theorem 5.2. We will illustrate these techniques with the equations in Theorem 4.2 for the case $\alpha = \pi$ and $\theta = \pi/4$. Let us recall that these equations characterize the set $\mathcal{H}$.

THEOREM 5.4. *For every $0 \le \varepsilon \le 1$, the following equations are $(\varepsilon, 1824\sqrt{\varepsilon})$-robust:*

$$\begin{aligned}&\Pr{}^0[\boldsymbol{G}(|0\rangle\langle 0|)] = 1/2, \quad \Pr{}^0[\boldsymbol{G}^2(|0\rangle\langle 0|)] = 1,\\ &\Pr{}^0[\boldsymbol{G}^2(|1\rangle\langle 1|)] = 0.\end{aligned}$$

Proof. Let $\boldsymbol{G}$ be a CPSO which ε-satisfies the equations. First we will show that there is a point $\overline{\rho} \in \mathcal{S}$ with z-coordinate 0 whose distance from $\overline{\boldsymbol{G}(|0\rangle\langle 0|)}$ is at most $10\sqrt{\varepsilon}$. The last two equations imply that $\|\boldsymbol{G}^2(|b\rangle\langle b|) - |b\rangle\langle b|\|_1 \le 3\sqrt{\varepsilon}$ for $b = 0, 1$. Therefore $\|\boldsymbol{G}^2(|0\rangle\langle 0|) - \boldsymbol{G}^2(|1\rangle\langle 1|)\|_1 \ge 2 - 6\sqrt{\varepsilon}$, and by Lemma 3.1(a) we have $\|\boldsymbol{G}(|0\rangle\langle 0|) - \boldsymbol{G}(|1\rangle\langle 1|)\|_1 \ge 2 - 6\sqrt{\varepsilon}$. Thus $\|\overline{\boldsymbol{G}(|b\rangle\langle b|)}\| \ge 1 - 6\sqrt{\varepsilon}$ for $b = 0, 1$. Let $\tau = \rho(1/2, \alpha)$, where $\boldsymbol{G}(|0\rangle\langle 0|) = \rho(p, \alpha)$. The first equation implies that $\|\overline{\tau} - \overline{\boldsymbol{G}(|0\rangle\langle 0|)}\| \le 2\varepsilon$. Therefore for $\overline{\rho} = \overline{\tau}/\|\overline{\tau}\|$ we get $\|\boldsymbol{G}(|0\rangle\langle 0|) - \rho\|_1 \le 10\sqrt{\varepsilon}$.

The point $\overline{\rho}$ on $\mathcal{S}$ uniquely defines $\varphi \in [0, 2\pi)$ such that $\overline{\boldsymbol{H}_\varphi(|0\rangle\langle 0|)} = \overline{\rho}$. One can verify that $\boldsymbol{H}_\varphi^{-1} \circ \boldsymbol{G}$ acts as the identity with error at most $19\sqrt{\varepsilon}$ on the four density matrices $|0\rangle\langle 0|$, $|1\rangle\langle 1|$, $\boldsymbol{H}_\varphi(|0\rangle\langle 0|)$, and $\boldsymbol{H}_\varphi(|1\rangle\langle 1|)$. From Lemma 3.5 we conclude that $\|\boldsymbol{G} - \boldsymbol{H}_\varphi\|_\infty \le 1824\sqrt{\varepsilon}$. □

6. Quantum self-testers. In this final section we formally define our testers and establish the relationship between robust equations and testability. Again, we will do it here only for the case of one qubit and one variable. Let $\boldsymbol{G}$ be a CPSO. The *experimental oracle* $\mathcal{O}[\boldsymbol{G}]$ for $\boldsymbol{G}$ is a probabilistic procedure. It takes inputs from $\{0, 1\} \times \mathbb{N}$ and generates outcomes from the set $\{0, 1\}$ such that for every $k \in \mathbb{N}$

$$\Pr[\mathcal{O}[\boldsymbol{G}](b, k) = 0] = \Pr{}^0[\boldsymbol{G}^k(|b\rangle\langle b|)].$$

An oracle program T with an experimental oracle $\mathcal{O}[\boldsymbol{G}]$ is a program denoted by $T^{\mathcal{O}[\boldsymbol{G}]}$ which can ask queries from the experimental oracle in the following sense: When it presents a query (b, k) to the oracle, in one computational step it receives the probabilistic outcome of $\mathcal{O}[\boldsymbol{G}]$ on it.

DEFINITION 6.1. *Let $\mathcal{F}$ be a family of CPSOs, and let $0 \le \delta_1 \le \delta_2 < 1$. A (δ_1, δ_2)-*tester *for $\mathcal{F}$ is a probabilistic oracle program T such that for every CPSO $\boldsymbol{G}$*

- *if $\mathrm{dist}_\infty(\boldsymbol{G}, \mathcal{F}) \le \delta_1$, then $\Pr[T^{\mathcal{O}[\boldsymbol{G}]}$ says $\mathtt{PASS}] \ge 2/3$,*
- *if $\mathrm{dist}_\infty(\boldsymbol{G}, \mathcal{F}) > \delta_2$, then $\Pr[T^{\mathcal{O}[\boldsymbol{G}]}$ says $\mathtt{FAIL}] \ge 2/3$,*

where the probability is taken over the probability distribution of the outcomes of the experimental oracle and the coin tosses of the program.

Since norms are equivalent in fixed dimension, the testability of families of CPSOs acting on a constant number of qubits does not change for any norm. This is stated in the following fact.

FACT 6.2. *Assume that T is a (δ_1, δ_2)-tester for a family $\mathcal{F}$ of CPSOs for k-qubits. Then T is a $(\delta_1/\alpha, \delta_2/\beta)$-tester for $\mathcal{F}$ when dist_∞ is replaced by any distance d such that $\beta\mathrm{d}(\boldsymbol{G}, \boldsymbol{G'}) \le \mathrm{dist}_\infty(\boldsymbol{G}, \boldsymbol{G'}) \le \alpha\mathrm{d}(\boldsymbol{G}, \boldsymbol{G'})$, for all CPSOs $\boldsymbol{G}, \boldsymbol{G'}$ for k-qubits and for $0 < \beta \le \alpha$.*

THEOREM 6.3. *Let $\varepsilon, \delta > 0$, and let (E) be a satisfiable set of d experimental equations such that the size of every equation is at most k. If (E) is (ε, δ)-robust, then there exists an $(\varepsilon/(3k), \delta)$-tester for $\mathcal{F}_{(E)}$ which makes $O(d\ln(d)/\varepsilon^2)$ queries.*

Proof. We will describe a probabilistic oracle program T. Let $\boldsymbol{G}$ be a CPSO. We can suppose that, for every equation in (E), T has a rational number $\tilde{r}$ such that $|\tilde{r} - r| \leq \varepsilon/6$, where r is the constant term of the equation. By sampling the oracle $\mathcal{O}[\boldsymbol{G}]$, for every equation in (E), T obtains a value $\tilde{p}$ such that $|\tilde{p} - p| \leq \varepsilon/6$ with probability at least $1 - 1/(3d)$, where p is the probability term of the equation. A standard Chernoff bound argument shows that this is feasible with $O(\ln(d)/\varepsilon^2)$ queries for each equation. If for every equation $|\tilde{p} - \tilde{r}| \leq 2\varepsilon/3$, then T says `PASS`; otherwise, T says `FAIL`. Using the robustness of (E) and Fact 6.4, one can verify that T is a $(\varepsilon/(3k), \delta)$-tester for $\mathcal{F}_{(E)}$. □

FACT 6.4. *Let (E) be a finite satisfiable set of experimental equations such that the size of every equation is at most k, and let $\boldsymbol{G}$ be a CPSO. For every $\varepsilon \geq 0$, if $\mathrm{dist}_\infty(\boldsymbol{G}, \mathcal{F}_{(E)}) \leq \varepsilon$, then $\boldsymbol{G}$ $(k\varepsilon)$-satisfies (E).*

Our main result is the consequence of Theorems 4.2, 4.4, 4.5, 5.2, 5.4, and 6.3 and the many-qubit generalizations of them.

THEOREM 6.5. *Let $\mathcal{F}$ be one of the following families:*

- $\mathcal{R}_{\alpha,\theta}$ *for* $(\alpha, \theta) \in (0, \pi] \times (0, \pi/2] \backslash \{(\pi, \pi/2)\}$, *where* α/π *is rational,*
- $\{(\boldsymbol{H}_\varphi, \mathbf{NOT}_\varphi) : \varphi \in [0, 2\pi)\}$,
- $\mathcal{H} \times \{\boldsymbol{R}_{\pm\alpha}\}$ *for* α/π *rational,*
- $\{(\boldsymbol{H}_\varphi, \mathbf{c\text{-}NOT}_\varphi) : \varphi \in [0, 2\pi)\}$,
- $\{(\boldsymbol{H}_\varphi, \boldsymbol{R}_{s\pi/4}, \mathbf{c\text{-}NOT}_\varphi) : \varphi \in [0, 2\pi), s = \pm 1\}$.

Then there exists an integer $k \geq 1$ and a real $C > 0$ such that, for all $\varepsilon > 0$, $\mathcal{F}$ has an $(\varepsilon, C\varepsilon^{1/k})$-tester which makes $O(1/\varepsilon^2)$ queries. Moreover, for every $0 < \varepsilon \leq 1$, $\mathcal{H}$ has an $(\varepsilon/6, 4579\sqrt{\varepsilon})$-tester which makes $O(1/\varepsilon^2)$ queries.

Note that each triplet of the last family forms a universal and fault-tolerant set of quantum gates [7].

Acknowledgments. We thank Jean-Benoit Bost, Stéphane Boucheron, Charles Delorme, Stéphane Gonnord, Lucien Hardy, Richard Jozsa, and Vlatko Vedral for several useful discussions and advice. We are very grateful to the anonymous referee for the helpful comments and suggestions.

REFERENCES

[1] L. ADLEMAN, J. DEMARRAIS, AND M. HUANG, *Quantum computability*, SIAM J. Comput., 26 (1997), pp. 1524–1540.

[2] D. AHARONOV AND M. BEN-OR, *Fault-tolerant quantum computation with constant error*, in Proceedings of the 29th STOC, ACM, New York, 1997, pp. 46–55.

[3] D. AHARONOV, A. KITAEV, AND N. NISAN, *Quantum circuits with mixed states*, in Proceedings of the 30th STOC, ACM, New York, 1998, pp. 20–30.

[4] A. BARENCO, *A universal two-bit gate for quantum computation*, Proc. R. Soc. Lond. Ser. A, 449 (1995), pp. 679–683.

[5] C.H. BENNETT AND G. BRASSARD, *Quantum cryptography: Public key distribution and coin tossing*, in Proceedings of the IEEE International Conference on Computers, Systems, and Signal Processing, 1984, pp. 175–179.

[6] A. BARENCO, C.H. BENNETT, R. CLEVE, D. DIVINCENZO, N. MARGOLUS, P. SHOR, T. SLEATOR, J. SMOLIN, AND H. WEINFURTER, *Elementary gates for quantum computation*, Phys. Rev. A, 52 (1995), pp. 3457–3467.

[7] P. BOYKIN, T. MOR, M. PULVER, V. ROYCHOWDHURY, AND F. VATAN, *On universal and fault-tolerant quantum computing: A novel basis and a new constructive proof of universality for Shor's basis*, in Proceedings of the 40th FOCS, ACM, New York, 1999, pp. 486–494.

[8] M. BLUM, M. LUBY, AND R. RUBINFELD, *Self-testing/correcting with applications to numerical problems*, J. Comput. System Sci., 47 (1993), pp. 549–595.

[9] R. BENEDETTI AND J.-J. RISLER, *Real Algebraic and Semi-algebraic Sets*, Hermann, Paris, 1990.

[10] I.L. Chuang and M.A. Nielsen, *Prescription for experimental determination of the dynamics of a quantum black box*, J. Modern Opt., 44 (1997), pp. 732–744.

[11] D. Deutsch, *Quantum theory, the Church-Turing principle and the universal quantum computer*, Proc. R. Soc. Lond. Ser. A, 400 (1985), pp. 97–117.

[12] D. Deutsch, *Quantum computational networks*, Proc. R. Soc. Lond. Ser. A, 425 (1989), pp. 73–90.

[13] D. Deutsch, A. Barenco, and A. Ekert, *Universality in quantum computation*, Proc. R. Soc. Lond. Ser. A, 449 (1995), pp. 669–677.

[14] D. DiVincenzo, *Two-bit gates are universal for quantum computation*, Phys. Rev. A, 51 (1995), pp. 1015–1022.

[15] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan, and A. Wigderson, *Self-testing/correcting for polynomials and for approximate functions*, in Proceedings of the 23rd STOC, ACM, New York, pp. 32–42, 1991.

[16] L. Hörmander, *The Analysis of Linear Partial Differential Operators* II, Springer-Verlag, Berlin, 1983.

[17] A. Kitaev, *Quantum computations: Algorithms and error correction*, Russian Math. Surveys, 52 (1997), pp. 1191–1249.

[18] E. Knill, R. Laflamme, and W.H. Zurek, *Threshold Accuracy for Quantum Computation*, http://xxx. lanl.gov/abs/quant-ph/9610011.

[19] E. Knill, R. Laflamme, and W.H. Zurek, *Resilient quantum computation: Error models and thresholds*, Proc. R. Soc. Lond. Ser. A, 454 (1998), pp. 365–384.

[20] N. Linden, H. Barjat, and R. Freeman, *An implementation of the Deutsch-Jozsa algorithm on a three-qubit NMR quantum computer*, Chem. Phys. Lett., 296 (1998), pp. 61–67.

[21] R. Lipton, *New Directions in Testing*, Discrete Math. Theor. Comput. Sci. 2, ACM/AMS, New York, 1991, pp. 191–202.

[22] S. Lloyd, *Almost any quantum logic gate is universal*, Phys. Rev. Lett., 75 (1995), pp. 346–349.

[23] D. Mayers and A. Yao, *Quantum cryptography with imperfect apparatus*, in Proceedings of the 39th FOCS, ACM, New York, 1998, pp. 503–509.

[24] J.F. Poyatos, J.I. Cirac, and P. Zoller, *Complete characterization of a quantum process: The two-bit quantum gate*, Phys. Rev. Lett., 78 (1997), pp. 390-393.

[25] J. Preskill, *Quantum Computing*, lecture notes, http://www.theory.caltech.edu/people/preskill/ph229/.

[26] R. Rubinfeld, *A Mathematical Theory of Self-Checking, Self-Testing and Self-Correcting Programs*, Ph.D. thesis, University of California, Berkeley, CA, 1990.

[27] R. Rubinfeld and M. Sudan, *Robust characterizations of polynomials with applications to program testing*, SIAM J. Comput., 25 (1996), pp. 252–271.

[28] R. Rubinfeld, *On the robustness of functional equations*, SIAM J. Comput., 28 (1999), pp. 1972–1997.

[29] S.G. Schirmer, T. Zhang, and J.V. Leahy, *Orbits of quantum states and geometry of Bloch vectors for N-level systems*, J. Phys. A, 37 (2004), pp. 1389–1402.

[30] P. Shor, *Fault-tolerant quantum computation*, in Proceedings of the 37th FOCS, ACM, New York, 1996, pp. 56–65.

[31] A. Yao, *Quantum circuit complexity*, in Proceedings of the 34th FOCS, ACM, New York, 1993, pp. 352–361.

SIAM J. Comput.
Vol. 37, No. 2, pp. 630–652

FASTER AND SIMPLER ALGORITHMS FOR MULTICOMMODITY FLOW AND OTHER FRACTIONAL PACKING PROBLEMS*

NAVEEN GARG† AND JOCHEN KÖNEMANN‡

Abstract. This paper considers the problem of designing fast, approximate, combinatorial algorithms for multicommodity flows and other fractional packing problems. We present new, faster, and much simpler algorithms for these problems.

Key words. multicommodity flow, fractional packing, concurrent flow, approximation algorithms

AMS subject classifications. 68Q25, 90C08, 90C27, 90C35, 90C59

DOI. 10.1137/S0097539704446232

1. Introduction.

Consider the problem of computing a maximum s-t flow in a graph with unit edge capacities. While there are many different algorithms known for this problem we discuss one which views the problem purely as one of packing s-t paths so that constraints imposed by edge capacities are not violated. The algorithm associates a length with each edge and at any step it routes a unit flow along the shortest s-t path. It then multiplies the length of every edge on this path by $1+\epsilon$ for a fixed ϵ. Thus the longer an edge is the greater the flow through it is. Since we always choose the shortest s-t path to route flow along, we essentially try to balance the flow on all edges in the graph. One can argue that, if, after sufficiently many steps, M is the maximum flow through an edge, then the flow computed is almost M times the maximum s-t flow. Therefore scaling the flow by M gives a feasible flow which is almost maximum.

Note that the length of an edge at any step is exponential in the total flow going through the edge. Such a length function was first proposed by Shahrokhi and Matula [22], who used it to compute the throughput of a given multicommodity flow instance. While this problem (and all other problems considered in this paper) can be formulated as a linear program and solved to optimality using fast matrix multiplication [24], the authors of [22] were mainly interested in providing fast, possibly approximate, combinatorial algorithms. Their procedure, which applied only to the case of uniform edge capacities, computed a $(1+\omega)$-approximation to the maximum throughput in time polynomial in ω^{-1}. The key idea of their procedure, which was adopted in numerous subsequent papers, was to compute an initial flow by disregarding edge capacities and then to reroute this, iteratively, along short paths so as to reduce the maximum congestion on any edge.

The running time of [22] was improved significantly by Klein et al. [17]. It was then extended and refined to the case of arbitrary edge capacities by Leighton et al.

*Received by the editors September 30, 2004; accepted for publication (in revised form) October 30, 2006; published electronically June 15, 2007.
http://www.siam.org/journals/sicomp/37-2/44623.html

†Computer Science and Engineering, Indian Institute of Technology, New Delhi, India (naveen@cse.iitd.ac.in). This work of this author was supported by the EU ESPRIT LTR Project N. 20244 (ALCOM-IT). This work was done while the author was at the Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany.

‡Department of Combinatorics and Optimization, University of Waterloo, ON N2L 3G1, Canada (jochen@math.uwaterloo.ca). This work was done while the author was at the Universität des Saarlandes, Im Stadtwald, 66123 Saarbrücken, Germany.

[18], Goldberg [11], and Radzik [21] to obtain better running times; see Table 1 for the current best bound.

Plotkin, Shmoys, and Tardos [20] observed that a similar technique could be applied to solve any fractional packing problem. Their approach to packing problems starts with an infeasible solution. The amount by which a packing constraint is violated is captured by a variable which is exponential in the extent of this violation. At any step the packing is modified by a *fixed amount* in a direction determined by these variables. Hence, the running time of the procedure depends upon the maximum extent to which any constraint could be violated; this is referred to as the *width* of the problem [20]. In [20], the authors propose several width-reduction techniques in order to decrease the (pseudopolynomial) running time of the algorithm.

Grigoriadis and Khachiyan [13] consider block angular packing problems which are problems of the form

$$\min \left\{ \lambda | \sum_{i=1}^{k} f^i(x^i) \leq \lambda e, x^i \in B^i, 1 \leq i \leq k \right\},$$

where B^i is a convex set, $f^i : B^i \to \mathbb{R}^m$ is a nonnegative convex function, and e is the vector of all 1's. They assume the existence of an oracle which, given $i, 1 \leq i \leq k$, nonnegative vector y, and scalar μ, computes $\min \left\{ y^T f^i(x) | f^i(x) \leq \mu e, x \in B^i \right\}$, and they show how to find a $(1+\epsilon)$-approximation to the block angular packing problem with only $k^2 \ln m(\epsilon^{-2} + \ln k)$ calls to this oracle. In [14], Grigoriadis and Khachiyan show that this problem can also be solved in $km(\epsilon^{-2} \ln \epsilon^{-1} + \ln m)$ calls to an oracle which computes $\min \left\{ y^T f^i(x) | x \in B^i \right\}$. Note that both these running times are independent of the width of the problem.

All the problems that we consider in this paper can be formulated as block angular packing problems. This is immediate for the maximum concurrent flow and the min-cost multicommodity flow problems. For these problems, the blocks are single commodity flows. In [13], the oracle corresponds to finding a min-cost single-commodity flow, while in [14], the oracle is a shortest path computation.

The maximum multicommodity flow problem can also be formulated as a block angular packing problem with one block, B, which is the set of all multicommodity flows of total value 1. For a flow $x \in B$, $f(x)$ is a vector denoting the fraction of the capacity utilized by x on the edges. If for a flow x, $f(x) \leq \lambda e$, then the flow x/λ satisfies all capacities and routes $1/\lambda$ units. Thus computing maximum flow is the same as minimizing λ. A similar idea can also be used to formulate fractional packing as a block angular convex program.

In a significant departure from this line of research and motivated by ideas from randomized rounding, Young [25] proposed an *oblivious rounding* approach to packing problems. Young's approach has the essential ingredient of previous approaches—a length function which measures, and is exponential in, the extent to which each constraint is violated by a given solution. However, [25] builds the solution from scratch and at each step adds to the packing a variable which violates only such packing constraints that are not already too violated. In particular, for multicommodity flow, it implies a procedure which does not involve rerouting flow (the flow is scaled only at the end) and which for the case of maximum s-t flow reduces to the algorithm discussed at the beginning of this section.

Our contributions. In this paper we provide a unified framework for multicommodity flow and packing problems which yields significantly simpler and faster algorithms than previously known. Our approach is similar to Young's approach to packing problems. However, we develop a new and simple combinatorial analysis which has the added flexibility that it allows us to make the greatest possible advance at each step. Thus for the setting of maximum s-t flows with integral edge capacities, Young's procedure routes a unit flow at each step, while our procedure would route enough flow so as to saturate the minimum capacity edge on the shortest s-t path. This simple modification is quite powerful and delivers a slightly better running time and a much simpler proof.

Our approach yields a new, very natural, algorithm for maximum concurrent flow (section 5) which extends in a straightforward manner to min-cost multicommodity flows (section 6). These algorithms use a min-cost flow computation as a subroutine and have running times that match the best known. We also provide algorithms for these two problems which use shortest path computations as a subroutine and are faster than previous algorithms. One idea in these algorithms which is key to the faster running times is to organize all computation sequentially and to use the length updates done at one step in the computations done at all subsequent steps. This is, in some ways, similar to the round-robin idea employed by Radzik [21].

This paper first appeared as a technical report in [9] and then as an extended abstract in [10]. Subsequently the approach presented here has been extended and improved results have been obtained for almost all the problems considered here. For the maximum multicommodity flow problem, Fleischer [7] obtained a running time that is independent of the number of commodities. Karakostas [15] obtained a corresponding result for the maximum multicommodity flow and min-cost multicommodity flow problems. We discuss the ideas behind these improvements in the appropriate sections.

Bienstock and Iyengar [4] recently adapted a method by Nesterov [19] in order to obtain a $(1+\omega)$-approximation for generalized packing problems. The ω-dependence of the running time of their algorithm is $O((1/\omega)\cdot\log 1/\omega)$, as opposed to a dependence of $O(1/\omega^2)$ of our algorithms. However, their algorithm needs to solve a convex quadratic program in each iteration, and this is computationally substantially more expensive than the oracle calls necessary in our algorithms. As a result, our algorithms remain faster than the algorithm by Bienstock and Iyengar for a fixed or moderately small ω.

Table 1 summarizes our results. All our algorithms are deterministic and compute a $(1+\omega)$-approximation to the optimum solution. In giving the running times we ignore polylog factors; the $\tilde{O}$ denotes this fact.

TABLE 1

A summary of our results. Here, D denotes the number of nonzero entries in the given constraint matrix and L is the maximum number of nonzero entries in a column.

Problem	Previous best	Our running time	Subseq. improvement
Max. mult. flow	$\tilde{O}(\omega^{-2}km^2)$ [14]	$\tilde{O}(\omega^{-2}km^2)$	$\tilde{O}(\omega^{-2}m^2)$ [7]
Frac. packing	$\tilde{O}(\omega^{-2}mD)$ [14]	$\tilde{O}(\omega^{-2}mD)$	$\tilde{O}(\omega^{-2}(mL+D))$ [26]
Maximum concurrent flow	$\tilde{O}(\omega^{-2}kmn)$ [21]	$\tilde{O}(\omega^{-2}kmn)$	
	$\tilde{O}(\omega^{-2}km^2)$ [14]	$\tilde{O}(\omega^{-2}(k+m)m)$	$\tilde{O}(\omega^{-2}m^2)$ [15]
Max. cost-bd. conc. flow	$\tilde{O}(\omega^{-2}kmn)$ [12]	$\tilde{O}(\omega^{-2}kmn)$	
	$\tilde{O}(\omega^{-2}km^2)$ [14]	$\tilde{O}(\omega^{-2}(k+m)m)$	$\tilde{O}(\omega^{-2}m^2)$ [15]

The framework introduced in this paper for multicommodity flow problems was extended by Fleischer and Wayne [8] to generalized flow. The book by Bienstock [3] is a good survey of the theoretical issues and computational studies done on this topic.

2. Maximum multicommodity flow. Given a graph $G = (V, E)$ with edge capacities $c : E \to \mathbb{R}^+$ and k pairs of terminals $(s_1, t_1), \ldots, (s_k, t_k)$, with one commodity associated with each pair, we want to find a multicommodity flow such that the sum of the flows of all commodities is maximized. Let $\mathcal{P}_j$ be the set of s_j, t_j-paths in G for all $1 \leq j \leq k$, and define $\mathcal{P}$ to be the union of $\mathcal{P}_1, \ldots, \mathcal{P}_k$. Also, let $\mathcal{P}_e$ be the set of paths in $\mathcal{P}$ that use edge e for all $e \in E$. The path-flow linear programming formulation for the maximum multicommodity flow problem has a variable $x(p)$ for the flow sent along each path $p \in \mathcal{P}$:

$$(P_{\text{mmc}}) \qquad \begin{aligned} \max \quad & \sum_{p \in \mathcal{P}} x(p) \\ \text{s.t.} \quad & \sum_{p \in \mathcal{P}_e} x(p) \leq c(e) \quad \forall e \in E, \\ & x \geq 0. \end{aligned}$$

The dual to this linear program associates a length $l(e)$ with each of the edges $e \in E$:

$$(D_{\text{mmc}}) \qquad \begin{aligned} \min \quad & D(l) \overset{\text{def}}{=} \sum_{e \in E} c(e) \cdot l(e) \\ \text{s.t.} \quad & \sum_{e \in p} l(e) \geq 1 \quad \forall p \in \mathcal{P}, \\ & l \geq 0. \end{aligned}$$

Observe that the above two linear programs (P_{mmc}) and (D_{mmc}) have exponential size. Optimal solutions can, however, be found in polynomial time as equivalent polynomial-size edge-flow formulations exist (e.g., see [1]).

In the following let $\texttt{dist}_j(l)$ be the length of the shortest s_j, t_j-path with respect to length l for $1 \leq j \leq k$. Also let $\alpha(l) \overset{\text{def}}{=} \min_j \texttt{dist}_j(l)$ be the minimum length path between any pair of terminals. Then (D_{mmc}) is equivalent to finding a length function $l : E \to \mathbb{R}^+$ such that $\frac{D(l)}{\alpha(l)}$ is minimized. Let $\beta \overset{\text{def}}{=} \min_l D(l)/\alpha(l)$.

The algorithm proceeds in iterations. Let l_{i-1} be the length function at the beginning of the ith iteration and f_{i-1} be the total flow routed in iterations $1 \ldots i-1$. Let P be a path of length $\alpha(l_{i-1})$ between a pair of terminals, and let c be the capacity of the minimum capacity edge on P. In the ith iteration we route c units of flow along P. Thus $f_i = f_{i-1} + c$. The function l_i differs from l_{i-1} only in the lengths of the edges along P; these are modified as $l_i(e) = l_{i-1}(e)(1 + \epsilon c/c(e))$, where ϵ is a constant to be chosen later.

Initially every edge e has length δ, i.e., $l_0(e) = \delta$ for some constant δ to be chosen later. For brevity we denote $\alpha(l_i), D(l_i)$ by $\alpha(i), D(i)$, respectively. The procedure stops after t iterations, where t is the smallest number such that $\alpha(t) \geq 1$.

2.1. Analysis. For every iteration $i \geq 1$

$$\begin{aligned} D(i) &= \sum_e l_i(e)c(e) \\ &= \sum_e l_{i-1}(e)c(e) + \epsilon \sum_{e \in P} l_{i-1}(e)c \\ &= D(i-1) + \epsilon(f_i - f_{i-1})\alpha(i-1), \end{aligned}$$

which implies that

$$D(i) = D(0) + \epsilon \sum_{j=1}^{i} (f_j - f_{j-1})\alpha(j-1). \tag{1}$$

Consider the length function $l_i - l_0$. Note that $D(l_i - l_0) = D(i) - D(0)$ and $\alpha(l_i - l_0) \geq \alpha(i) - \delta L$, where L is the maximum number of edges on any simple path in G. Hence

$$\beta \leq \frac{D(l_i - l_0)}{\alpha(l_i - l_0)} \leq \frac{D(i) - D(0)}{\alpha(i) - \delta L}. \tag{2}$$

Substituting this bound on $D(i) - D(0)$ into (1), we get

$$\alpha(i) \leq \delta L + \frac{\epsilon}{\beta} \sum_{j=1}^{i} (f_j - f_{j-1})\alpha(j-1).$$

To solve the above recurrence we first note that the sequence $x(0), x(1), \ldots x(i), \ldots,$ where $x(i) = \delta L + \frac{\epsilon}{\beta} \sum_{j=1}^{i} (f_j - f_{j-1})x(j-1)$, dominates the sequence

$$\alpha(0), \alpha(1), \ldots, \alpha(i), \ldots,$$

where $x(0) = \alpha(0)$. Now

$$\begin{aligned} x(i) &= \delta L + \frac{\epsilon}{\beta} \sum_{j=1}^{i-1} (f_j - f_{j-1})x(j-1) + \frac{\epsilon}{\beta}(f_j - f_{j-1})x(i-1) \\ &= x(i-1)(1 + \epsilon(f_i - f_{i-1})/\beta) \\ &\leq x(i-1)e^{\epsilon(f_i - f_{i-1})/\beta}. \end{aligned}$$

Since $x(0) = \alpha(0) \leq \delta L$ we have $x(i) \leq \delta L e^{\epsilon f_i/\beta}$, and this implies

$$\alpha(i) \leq \delta L e^{\epsilon f_i/\beta}.$$

By our stopping condition

$$1 \leq \alpha(t) \leq \delta L e^{\epsilon f_t/\beta}, \tag{3}$$

and hence

$$\frac{\beta}{f_t} \leq \frac{\epsilon}{\ln(\delta L)^{-1}}. \tag{4}$$

CLAIM 2.1. *There is a feasible flow of value* $\frac{f_t}{\log_{1+\epsilon}\frac{1+\epsilon}{\delta}}$.

Proof. Consider an edge e. For every $c(e)$ units of flow routed through e the length of e increases by a factor of at least $1 + \epsilon$. The last time its length was increased, e was on a path of length strictly less than 1. Since every increase in edge-length is by a factor of at most $1 + \epsilon$, $l_t(e) < 1 + \epsilon$. Since $l_0(e) = \delta$ it follows that the total flow through e is at most $c(e)\log_{1+\epsilon}\frac{1+\epsilon}{\delta}$. Scaling the flow, f_t, by $\log_{1+\epsilon}\frac{1+\epsilon}{\delta}$ then gives a feasible flow of claimed value. □

Thus the ratio of the values of the optimum dual and the primal solutions, γ, is $\frac{\beta}{f_t}\log_{1+\epsilon}\frac{1+\epsilon}{\delta}$. By substituting the bound on β/f_t from (4) we obtain

$$\gamma \leq \frac{\epsilon \log_{1+\epsilon}\frac{1+\epsilon}{\delta}}{\ln(\delta L)^{-1}} = \frac{\epsilon}{\ln(1+\epsilon)} \frac{\ln\frac{1+\epsilon}{\delta}}{\ln(\delta L)^{-1}}.$$

The ratio $\frac{\ln(1+\epsilon)\delta^{-1}}{\ln(\delta L)^{-1}}$ equals $(1-\epsilon)^{-1}$ for $\delta = (1+\epsilon)((1+\epsilon)L)^{-1/\epsilon}$. Hence with this choice of δ we have

$$\gamma \leq \frac{\epsilon}{(1-\epsilon)\ln(1+\epsilon)} \leq \frac{\epsilon}{(1-\epsilon)(\epsilon-\epsilon^2/2)} \leq (1-\epsilon)^{-2}.$$

Since this quantity should be no more than our approximation ratio $(1+w)$, we choose ϵ appropriately.

2.2. Running time. In the ith iteration we increase the length of the minimum capacity edge along P by a factor of $1+\epsilon$. Since, for any edge e, $l_0(e) = \delta$ and $l_t(e) < 1+\epsilon$, the number of iterations in which e is the minimum capacity edge on the path chosen in that iteration is at most $\lceil \frac{1}{\epsilon}\log_{1+\epsilon} L\rceil$. Using the fact that there are m edges we get the following theorem.

THEOREM 2.1. *There is an algorithm that computes a $(1+\omega)$-approximation to the maximum multicommodity flow in time $O(\omega^{-2}km\log L\cdot T_{sp})$, where L is the maximum number of edges on a path between any source-sink pair and T_{sp} is the time required to compute the shortest s-t path in a graph with nonnegative edge-weights.*

2.3. Subsequent improvements. Fleischer [7] made the interesting observation that it suffices to route flow along an approximate shortest path and that if the path chosen at each step is an a approximation to the shortest path then the approximation guarantee worsens only by a multiplicative factor a. Her algorithm proceeds in phases, each of which is composed of k iterations. If at the start of the ith phase the shortest path between each pair has length at least α, then in the jth iteration of this phase we route the jth commodity along any path of length at most $\alpha(1+\epsilon)$ and move to the next iteration only when the shortest path between s_j, t_j is at least $\alpha(1+\epsilon)$. This ensures that at the end of the ith phase every (s_j, t_j) pair is at least $\alpha(1+\epsilon)$ apart. Hence the number of phases is at most $\log_{1+\epsilon}\delta^{-1}$. The algorithm performs one shortest path computation in each iteration that does not result in flow being routed. Hence the total number of shortest path computations is $(m+k)\lceil\frac{1}{\epsilon}\log_{1+\epsilon} L\rceil$. Since k can be as large as $\mathrm{O}(n^2)$, Fleischer eliminates the dependence of the running time on k by routing all commodities with the same source in an iteration. It is possible to do this without additional effort since Dijkstra's algorithm for computing shortest paths gives the shortest path to every node in the graph.

3. Packing LP.

A packing **LP** is a linear program of the kind

$$\max\left\{c^T x | Ax \leq b, x \geq 0\right\},$$

where A, b, and c are $(m\times n), (m\times 1)$, and $(n\times 1)$ matrices, all of whose entries are positive. We also assume that for all i, j, the (i,j)th entry of A, $A(i,j)$, is at most $b(i)$. The dual of this **LP** is $\min\left\{b^T y | A^T y \geq c, y \geq 0\right\}$.

We view the rows of A as edges and the columns as paths. $b(i)$ is the capacity of edge i, and every unit of flow routed along the jth column consumes $A(i,j)$ units of capacity of edge i while providing a benefit of $c(j)$ units.

The dual variable $y(i)$ corresponds to the length of edge i. Define the *length* of a column j with respect to the dual variables y as $\mathtt{length}_y(j) \overset{\text{def}}{=} \sum_i A(i,j)y(i)/c(j)$. Finding a shortest path now corresponds to finding a column whose length is minimum; define $\alpha(y) \overset{\text{def}}{=} \min_j \mathtt{length}_y(j)$. Also define $D(y) \overset{\text{def}}{=} b^T y$. Then the dual program is equivalent to finding a variable assignment y such that $D(y)/\alpha(y)$ is minimized.

Once again our procedure will be iterative. Let y_{k-1} be the dual variables and f_{k-1} the value of the primal solution at the beginning of the kth iteration. Let q be the minimum length column of A, i.e., $\alpha(y_{k-1}) = \mathtt{length}_{y_{k-1}}(q)$—this corresponds to the path along which we route flow in this iteration. The minimum capacity edge is the row for which $b(i)/A(i,q)$ is minimum; let this be row p. Thus in this iteration we will increase the primal variable $x(q)$ by an amount $b(p)/A(p,q)$ so that $f_k = f_{k-1} + c(q)b(p)/A(p,q)$. The dual variables are modified as

$$y_k(i) = y_{k-1}(i)\left(1 + \epsilon\frac{b(p)/A(p,q)}{b(i)/A(i,q)}\right),$$

where ϵ is a constant to be chosen later.

The initial values of the dual variables are given by $y_0(i) = \delta/b(i)$ for some constant δ to be chosen later. For brevity we denote $\alpha(y_k), D(y_k)$ by $\alpha(k), D(k)$, respectively. Thus $D(0) = m\delta$. The procedure stops at the first iteration t such that $D(t) \geq 1$.

3.1. Analysis. The analysis here proceeds almost exactly as in the case of maximum multicommodity flow. For every iteration $k \geq 1$

$$\begin{aligned} D(k) &= \sum_i b(i)y_k(i) \\ &= \sum_i b(i)y_{k-1}(i) + \epsilon\frac{b(p)}{A(p,q)}\sum_i A(i,q)y_{k-1}(i) \\ &= D(k-1) + \epsilon(f_k - f_{k-1})\alpha(k-1), \end{aligned} \tag{5}$$

which, as before, implies that

$$D(k) = D(0) + \epsilon\sum_{l=1}^{k}(f_l - f_{l-1})\alpha(l-1).$$

Let $\beta \overset{\text{def}}{=} \min_y D(y)/\alpha(y)$. Then $\beta \leq D(l-1)/\alpha(l-1)$, and so

$$D(k) \leq m\delta + \frac{\epsilon}{\beta}\sum_{l=1}^{k}(f_l - f_{l-1})D(l-1).$$

In order to solve this recurrence, we first define

$$x(i) = m\delta + \frac{\epsilon}{\beta}\sum_{l=1}^{k}(f_l - f_{l-1})x(l-1)$$

for all $i \geq 0$. We note that the sequence $(x(i))_{i\geq 0}$ dominates the sequence $(D(i))_{i\geq 0}$. Now

$$\begin{aligned} x(k) &= m\delta + \frac{\epsilon}{\beta}\sum_{l=1}^{k-1}(f_l - f_{l-1})x(l-1) + \frac{\epsilon}{\beta}(f_k - f_{k-1})x(k-1) \\ &= \left(1 + \frac{\epsilon}{\beta}(f_k - f_{k-1})\right)x(k-1) \\ &\leq e^{\epsilon(f_k - f_{k-1})/\beta}x(k-1) \\ &\leq e^{\epsilon f_k/\beta}x(0) = m\delta \cdot e^{\epsilon f_k/\beta}. \end{aligned}$$

Using $D(k) \leq x(k)$ we therefore obtain

$$D(k) \leq m\delta e^{\epsilon f_k/\beta},$$

and by our stopping condition

$$1 \leq D(t) \leq m\delta e^{\epsilon f_t/\beta}, \tag{6}$$

and hence

$$\frac{\beta}{f_t} \leq \frac{\epsilon}{\ln(m\delta)^{-1}}.$$

CLAIM 3.1. *There is a feasible solution to the packing* **LP** *of value* $\frac{f_t}{\log_{1+\epsilon}\frac{1+\epsilon}{\delta}}$.

Proof. The primal solution x we constructed has value f_t. However, it may not be feasible since some packing constraint $(\sum_j A(i,j)x(j))/b(i) \leq 1$ may be violated. When we pick column q and increase $x(q)$ by $b(p)/A(p,q)$ we increase the left-hand side (LHS) of the ith constraint by $\frac{A(i,q)b(p)}{b(i)A(p,q)}$ ($= z$, say). Simultaneously we increase the dual variable $y(i)$ by a multiplicative factor of $1 + \epsilon z$. By our definition of p it follows that $z \leq 1$, and hence increasing the LHS of the ith constraint by 1 causes an increase in $y(i)$ by a multiplicative factor of at least $1+\epsilon$. Note that $y_{t-1}(i) < 1/b(i)$, and so $y_t(i) < (1+\epsilon)/b(i)$. Since $y_0(i) = \delta/b(i)$ it follows that the final value of the LHS of the ith constraint is no more than $\log_{1+\epsilon}\frac{1+\epsilon}{\delta}$. Since this is true for every i, scaling the primal solution by $\log_{1+\epsilon}\frac{1+\epsilon}{\delta}$ gives a feasible solution of value as in the claim. ☐

The rest of the analysis is exactly the same as in section 2.1 with m replacing L. Thus $\delta = (1+\epsilon)((1+\epsilon)m)^{-1/\epsilon}$.

3.2. Running time. In the kth iteration we increase the dual variable of the "minimum capacity" row by a factor of $(1+\epsilon)$. Since for any row i, $y_0(i) = \delta/b(i)$ and $y_t(i) < (1+\epsilon)/b(i)$ and there are m rows in all, the total number of iterations is at most $m\lceil\frac{1}{\epsilon}\log_{1+\epsilon} m\rceil$. For explicitly given packing programs one requires $\mathrm{O}(D)$ time to compute the minimum length column, where D is the number of nonzero entries in the matrix A. This implies a running time of $mD\lceil\frac{1}{\epsilon}\log_{1+\epsilon} m\rceil$ for computing a $(1-\epsilon)^{-2}$-approximation to the packing **LP**.

THEOREM 3.1. *There is an algorithm that computes a* $(1+\omega)$*-approximation to the packing* **LP** *in time* $O(\omega^{-2}mD\log m)$*, where* m *is the number of rows and* D *is the number of nonzero entries in the given constraint matrix.*

3.3. Subsequent improvements. Young [26] observed that with Fleischer's technique this running time improves to $\tilde{O}(\omega^{-2}(mL + D))$, where L is the maximum number of nonzero entries in a column and D is the number of nonzero entries in the constraint matrix.

4. Spreading metrics. Given a graph $G = (V, E)$ with edge costs $c : E \rightarrow \mathbb{R}^+$, a spreading metric is an assignment of lengths to the edges, $l : E \rightarrow \mathbb{R}^+$, so as to minimize $\sum_e l(e)c(e)$ subject to the constraint that for any set $S \subseteq V$ and vertex $r \in S$, $\sum_{v \in S} \texttt{dist}_{r,v}(l) \geq f(S)$, where $\texttt{dist}_{r,v}(l)$ is the distance from r to v under the length function l and $f()$ is a function only of the size of S. For the linear arrangement problem $f(S) = (|S| - 1)(|S| - 3)/4$ [6], while for the problem of computing a ρ-separator[1] $f(S)$ is defined as $|S| - \rho|V|$ [5].

Since the length function l is positive, the shortest paths from r to the other vertices in S form a tree—the shortest path tree rooted at r. Thus the above constraints can be equivalently stated as follows: for any tree T, for any subset S of vertices in T, and for any vertex $r \in S$

$$\sum_{v \in S} \texttt{dist}_{r,v}(l, T) \geq f(S),$$

where $\texttt{dist}_{r,v}(l, T)$ denotes the distance from r to v in tree T under the length function l.

Let $u_e(T, S, r)$ be the number of vertices of S in the subtree below edge e when T is rooted at r. Then the above constraint can be rewritten again to obtain the **LP**

$$\begin{aligned} \min \quad & \sum_e l(e)c(e) \\ \text{s.t.} \quad & \sum_{e \in T} l(e)u_e(T, S, r) \geq f(S) \quad \forall T, \forall S \subseteq T, \forall r \in S, \\ & l \geq 0. \end{aligned}$$

The dual of this program, which is a packing **LP**, has a nonnegative variable $x(T, S, r)$ for every tree T, subset $S \subseteq T$, and vertex $r \in S$ and is as follows:

$$\begin{aligned} \max \quad & \sum_{T,S,r} x(T, S, r)f(S) \\ \text{s.t.} \quad & \sum_{T: e \in T} x(T, S, r)u_e(T, S, r) \leq c(e) \quad \forall e \in E, \\ & x \geq 0. \end{aligned}$$

Note that the packing **LP** has exponentially many variables. However, the $(1+w)$-approximation to the optimum fractional solution, in the previous section, needed an oracle that returned only the "most violated constraint" of the dual **LP**. In this setting, this oracle is a subroutine, which, given a length function l, finds a triple (T, S, r) for which $(\sum_{e \in T} l(e)u_e(T, S, r))/f(S)$ or, equivalently, $(\sum_{v \in S} \texttt{dist}_{r,v}(l, T))/f(S)$ is minimum.

[1] A minimum cost set of edges whose removal disconnects the graph into connected components, each of which has at most $\rho|V|$ vertices.

Our subroutine will try out all n choices for vertex r and for each of these it will determine the best choice of T, S. For a given r and every subset S, the expression $\sum_{v\in S} \mathtt{dist}_{r,v}(l,T)$ is minimized when T is the tree of shortest paths from r and under the length function l. Therefore, for a given r, our choice of T will be the shortest path tree rooted at r. Since $f(S)$ depends only on $|S|$, given that $|S| = k$, the ratio $(\sum_{v\in S} \mathtt{dist}_{r,v}(l,T))/f(S)$ is minimized when S is the set of k nearest vertices to r. Amongst the n different choices for k, and hence for S, we choose the set for which the above ratio is minimum. Having found the best triple (T, S, r), we now determine the extent to which $x(T, S, r)$ is increased by considering all edges in T and finding the edge for which $c(e)/u_e(T, S, r)$ is minimum.

The subroutine thus requires n single-source shortest path computations. The running time of the procedure is obtained by noting that the subroutine is invoked once in each of the $m\lceil \frac{1}{\epsilon} \log_{1+\epsilon} m\rceil$ iterations.

Theorem 4.1. *There is an algorithm that computes a* $(1+\omega)$*-approximation to spreading metrics in time* $O(\omega^{-2} mn \log m \cdot T_{\mathrm{sp}})$*, where* T_{sp} *is the time required to compute single-source shortest paths in a graph with nonnegative edge-weights.*

It is easy to improve the running time by a factor n by using Fleischer's idea. After computing the shortest path tree from a certain root vertex and finding the best set S we continue with the same root vertex until the ratio is at least $(1+\epsilon)$ times the ratio at the start of the phase. Our analysis is now almost exactly the same as for the maximum multicommodity flow problem and leads to a running time of $\tilde{O}(\omega^{-2} m^2)$.

5. Maximum concurrent flow.

Once again we are given a graph with edge capacities $c : E \to \mathbb{R}^+$ and k commodities with s_j, t_j being the source and sink, respectively, for commodity j. Now each commodity has a demand $d(j)$ associated with it, and we want to find the largest λ such that there is a multicommodity flow which routes $\lambda d(j)$ units of commodity j.

In the following, let $\mathcal{F}_j$ be the set of flows that transport $d(j)$ units of flow from s_j to t_j for all $1 \le j \le k$. Similar to section 2, we use $\mathcal{F}$ to denote the union of $\mathcal{F}_1, \dots, \mathcal{F}_k$. For a flow $f \in \mathcal{F}$ and an edge $e \in E$, we let f_e be the amount of flow sent across e. We can then formulate the maximum concurrent flow problem as the following LP:

$$(P_{\mathrm{mcf}}) \qquad \begin{aligned} \max \quad & \lambda \\ \text{s.t.} \quad & \sum_{f\in\mathcal{F}} f_e \cdot x(f) \le c(e) \quad \forall e \in E, \\ & \sum_{f\in\mathcal{F}_j} x(f) \ge \lambda \quad \forall 1 \le j \le k, \\ & x \ge 0, \lambda \ge 0. \end{aligned}$$

Its dual has a length $l(e)$ for each edge $e \in E$, and a variable $z(j)$ for each commodity

$1 \le j \le k$:

$$(D_{\mathrm{mcf}}) \qquad \min \quad D(l) \overset{\mathrm{def}}{=} \sum_{e \in E} c(e) l(e)$$

$$\text{s.t.} \quad \sum_{e \in E} f_e \cdot l(e) \ge z(j) \quad \forall 1 \le j \le k, \forall f \in \mathcal{F}_j,$$

$$\sum_{j=1}^{k} z(j) \ge 1,$$

$$l, z \ge 0.$$

For a given $l : E \rightarrow \mathbb{R}^+$, $z(j)$ is the minimum cost of shipping $d(j)$ units of flow from s_j to t_j under cost function l, henceforth denoted by $\mathtt{min\_cost}_j(l)$. Let

$$\alpha(l) \overset{\mathrm{def}}{=} \sum_{j=1}^{k} \mathtt{min\_cost}_j(l).$$

LP (D_{mcf}) can now be recast as finding an assignment of lengths to the edges, $l : E \rightarrow \mathbb{R}^+$, such that $D(l)/\alpha(l)$ is minimized. Let β be this minimum. For now we assume that $\beta \ge 1$ and shall remove this assumption later.

The algorithm now proceeds in phases; each phase is composed of k iterations. Consider the jth iteration of the ith phase and let $l_{i,j-1}$ be the length function before this iteration. In this iteration we route $d(j)$ units of commodity j along the paths given by $\mathtt{min\_cost}_j(l_{i,j-1})$. Let $f_{i,j}(e)$ be the flow through edge e. The length function is modified as $l_{i,j}(e) = l_{i,j-1}(e)(1 + \epsilon f_{i,j}(e)/c(e))$. Then

$$\begin{aligned} D(l_{i,j}) &= \sum_e l_{i,j}(e) c(e) \\ &= D(l_{i,j-1}) + \epsilon \sum_e l_{i,j-1}(e) f_{i,j}(e) \\ &= D(l_{i,j-1}) + \epsilon \cdot \mathtt{min\_cost}_j(l_{i,j-1}). \end{aligned}$$

The lengths at the start of the $(i+1)$th phase are the same as those at the end of the ith phase, i.e., $l_{i+1,0} = l_{i,k}$. Initially, for any edge e, $l_{1,0}(e) = \delta/c(e) = l_{0,k}(e)$.

5.1. The analysis. We shall be interested in the values of the functions $D(), \alpha()$ only for the length functions $l_{i,k}, i \ge 0$. For brevity we denote $D(l_{i,k}), \alpha(l_{i,k})$ by $D(i), \alpha(i)$, respectively. With this new notation we have for $i \ge 1$

$$D(i) = D(l_{i,k}) = D(l_{i,0}) + \epsilon \sum_{j=1}^{k} \mathtt{min\_cost}_j(l_{i,j-1}).$$

Since the edge-lengths are monotonically increasing, $\mathtt{min\_cost}_j(l_{i,j-1}) \le \mathtt{min\_cost}_j(l_{i,k})$, and hence

$$D(i) \le D(l_{i,0}) + \epsilon \sum_{j=1}^{k} \mathtt{min\_cost}_j(l_{i,k}) = D(i-1) + \epsilon\alpha(i). \tag{7}$$

Since $\frac{D(i)}{\alpha(i)} \geq \beta$ we have

$$D(i) \leq \frac{D(i-1)}{1-\epsilon/\beta}.$$

Since $D(0) = m\delta$ we have for $i \geq 1$

$$\begin{aligned} D(i) &\leq \frac{m\delta}{(1-\epsilon/\beta)^i} \\ &= \frac{m\delta}{1-\epsilon/\beta}\left(1+\frac{\epsilon}{\beta-\epsilon}\right)^{i-1} \\ &\leq \frac{m\delta}{1-\epsilon/\beta} e^{\frac{\epsilon(i-1)}{\beta-\epsilon}} \\ &\leq \frac{m\delta}{1-\epsilon} e^{\frac{\epsilon(i-1)}{\beta(1-\epsilon)}}, \end{aligned}$$

where the last inequality uses our assumption that $\beta \geq 1$.

The procedure stops at the first phase t for which $D(t) \geq 1$. Therefore,

$$1 \leq D(t) \leq \frac{m\delta}{1-\epsilon} e^{\frac{\epsilon(t-1)}{\beta(1-\epsilon)}},$$

which implies

$$\frac{\beta}{t-1} \leq \frac{\epsilon}{(1-\epsilon)\ln\frac{1-\epsilon}{m\delta}}. \tag{8}$$

In the first $t-1$ phases, for every commodity j, we have routed $(t-1)d(j)$ units. However, this flow may violate capacity constraints.

CLAIM 5.1. $\lambda > \frac{t-1}{\log_{1+\epsilon} 1/\delta}$.

Proof. Consider an edge e. For every $c(e)$ units of flow routed through e, we increase its length by at least a factor $1+\epsilon$. Initially, its length is $\delta/c(e)$, and after $t-1$ phases, since $D(t-1) < 1$, the length of e satisfies $l_{t-1,k}(e) < 1/c(e)$. Therefore the total amount of flow through e in the first $t-1$ phases is strictly less than $\log_{1+\epsilon}\frac{1/c(e)}{\delta/c(e)} = \log_{1+\epsilon} 1/\delta$ times its capacity. Scaling the flow by $\log_{1+\epsilon} 1/\delta$ implies the claim. □

Thus the ratio of the values of the dual and primal solutions, γ, is strictly less than $\frac{\beta}{t-1}\log_{1+\epsilon} 1/\delta$. Substituting the bound on $\beta/(t-1)$ from (8), we get

$$\gamma < \frac{\epsilon \log_{1+\epsilon} 1/\delta}{(1-\epsilon)\ln\frac{1-\epsilon}{m\delta}} = \frac{\epsilon}{(1-\epsilon)\ln(1+\epsilon)} \frac{\ln 1/\delta}{\ln\frac{1-\epsilon}{m\delta}}.$$

For $\delta = (m/(1-\epsilon))^{-1/\epsilon}$ the ratio $\frac{\ln 1/\delta}{\ln\frac{1-\epsilon}{m\delta}}$ equals $(1-\epsilon)^{-1}$, and hence

$$\gamma \leq \frac{\epsilon}{(1-\epsilon)^2\ln(1+\epsilon)} \leq \frac{\epsilon}{(1-\epsilon)^2(\epsilon-\epsilon^2/2)} \leq (1-\epsilon)^{-3}.$$

Now it remains to choose ϵ suitably so that $(1-\epsilon)^{-3}$ is at most our desired approximation ratio $1+w$.

5.2. Running time. By weak-duality we have

$$1 \leq \gamma < \frac{\beta}{t-1} \log_{1+\epsilon} \frac{1}{\delta},$$

and hence the number of phases in the above procedure, t, is strictly less than $1 + \beta \log_{1+\epsilon} 1/\delta$, which implies that $t = \lceil \frac{\beta}{\epsilon} \log_{1+\epsilon} \frac{m}{1-\epsilon} \rceil$.

The running time of our computation depends on β, which can be reduced/increased by multiplying the demands/capacities appropriately. Let z_i be the maximum possible flow of commodity i, and let $z \overset{\text{def}}{=} \min_i z_i/d(i)$. Then z denotes the maximum fraction of the demands that can be routed independently, and hence $z/k \leq \beta \leq z$. We scale the capacities/demands so that $z/k = 1$ thus satisfying our assumption that $\beta \geq 1$. Note, however, that β could now be as large as k.

If our procedure does not stop within $2\lceil \frac{1}{\epsilon} \log_{1+\epsilon} \frac{m}{1-\epsilon} \rceil$ ($= T$, say) phases, then we know that $\beta \geq 2$. We double the demands of all commodities and continue the procedure. Note that β is now half its value in the previous phase and is at least 1. We run the procedure for an additional T phases and if it does not halt we again double demands. Since we halve the value of β after every T phases, the total number of phases is at most $T \log k$.

THEOREM 5.1. *There is an algorithm that computes a $(1+\omega)$-approximation to the maximum concurrent flow in time $O(\omega^{-2} k \log k \log m \cdot T_{\text{mcf}})$, where T_{mcf} is the time required to compute a minimum cost s-t flow in a graph with nonnegative edge-costs.*

The number of phases can be reduced further using an idea from [20]. We first compute a 2-approximation to β using the procedure outlined above. This requires $O(\log k \log m)$ phases and returns $\hat{\beta}$, $\beta \leq \hat{\beta} \leq 2\beta$. Now create a new instance by multiplying demands by $\hat{\beta}/2$; this instance has $1 \leq \beta \leq 2$. Therefore, we need at most an additional T phases to obtain a $(1+w)$-approximation. Thus the number of phases is $O(\log m(\log k + (\epsilon \ln 1 + \epsilon)^{-1}))$, which multiplied by k gives the number of single-commodity min-cost flow computations required.

5.3. Avoiding min-cost flow computations. We now show how min-cost flow computations can be avoided in the above algorithm for the maximum concurrent flow problem. Using the notation introduced in section 2, an alternate path-flow LP formulation of the maximum concurrent flow problem is as follows:

$$(P^2_{\text{mcf}}) \qquad \begin{aligned} \max \quad & \lambda \\ \text{s.t.} \quad & \sum_{p \in \mathcal{P}_e} x(p) \leq c(e) \quad \forall e \in E, \\ & \sum_{p \in \mathcal{P}_j} x(p) \geq \lambda \cdot d(j) \quad \forall 1 \leq j \leq k, \\ & x \geq 0, \lambda \geq 0. \end{aligned}$$

Its linear programming dual has a length $l(e)$ for each edge $e \in E$ and a variable $z(j)$ for each commodity j:

$$(D^2_{\mathrm{mcf}}) \qquad \begin{aligned} \min \quad & D(l) \overset{\mathrm{def}}{=} \sum_{e \in E} c(e) l(e) \\ \text{s.t.} \quad & \sum_{e \in p} l(e) \geq z(j) \quad \forall 1 \leq j \leq k, \forall p \in \mathcal{P}_j, \\ & \sum_{j=1}^{k} d(j) \cdot z(j) \geq 1, \\ & l, z \geq 0. \end{aligned}$$

For a given $l : E \rightarrow \mathbb{R}^+$, $z(j)$ is the shortest path between s_j and t_j under length function l. Define

$$\alpha(l) \overset{\mathrm{def}}{=} \sum_{j} d(j) \mathtt{dist}_j(l),$$

where $\mathtt{dist}_j(l)$ denotes the shortest path distance between s_j and t_j under the length function l. The dual (D^2_{mcf}) can then be viewed as an assignment of lengths to edges, $l : E \rightarrow \mathbb{R}^+$, such that $D(l)/\alpha(l)$ is minimized. Let β be this minimum.

The structure of this new algorithm is similar to that in the previous section. Thus the algorithm runs in phases, each of which is composed of k iterations. In the jth iteration of the ith phase, we route $d(j)$ units of commodity j in a sequence of steps. Let $l^{s-1}_{i,j}$ be the length function before the sth step, and let $P^s_{i,j}$ be the shortest path between s_j and t_j; i.e., $P^s_{i,j}$ has length $\mathtt{dist}_j(l^{s-1}_{i,j})$. In this step we route $f^s_{i,j} = \min\left\{c, d^{s-1}_{i,j}\right\}$ units of flow along $P^s_{i,j}$, where c is the capacity of the minimum capacity edge on this path. We now set $d^s_{i,j}$ to $d^{s-1}_{i,j} - f^s_{i,j}$; the iteration ends after p steps, where $d^p_{i,j} = 0$.

Thus at each step we perform a shortest path computation instead of a min-cost flow computation as in section 6. The length functions are modified in exactly the same manner as before and the analysis is almost exactly the same. Thus after routing all flow of commodity j we have

$$D(l^p_{i,j}) \leq D(l^0_{i,j}) + \epsilon \cdot d(j) \mathtt{dist}_j(l^p_{i,j}),$$

and after routing all commodities in the ith phase we have

$$D(l_{i,k}) \leq D(l_{i,0}) + \epsilon \sum_{j=1}^{k} d(j) \mathtt{dist}_j(l_{i,k}).$$

Using the same abbreviations as before we again obtain

$$D(i) \leq D(i-1) + \epsilon \alpha(i).$$

Beyond this point we follow the analysis of section 5.1 to argue that we have a $(1+\omega)$-approximation for the same choice of ϵ and δ.

For the running time we again note that in each step, except the last one in an iteration, we increase the length of at least one edge by a factor $1+\epsilon$. Since each edge has an initial length of δ and a final length less than $1+\epsilon$, the number of steps exceeds the number of iterations by at most $m \log_{1+\epsilon} \frac{1+\epsilon}{\delta}$. Thus the total number of steps is at most $(2k \log k + m)\lceil \frac{1}{\epsilon} \log_{1+\epsilon} \frac{m}{1-\epsilon} \rceil$ and each of these involves one shortest path computation.

Recall from the previous section that we needed k initial maximum flow computations to compute an approximate interval for the optimum throughput. We now describe a technique introduced by Grigoriadis and Khachiyan [14] to compute a slightly larger interval using k shortest path computations.

Define a length $l(e) = 1/c(e)$ for each edge $e \in E$. For $1 \leq i \leq k$, let P_i be a shortest s_i, t_i-path with respect to this length. Then let f be the flow obtained by sending $d(i)$ units of flow along path P_i for all $1 \leq i \leq k$ concurrently. Let f^* be an optimum concurrent flow that feasibly routes $\beta \cdot d(i)$ units of flow from s_i to t_i for each commodity i and define $\bar{f}$ as $(1/\beta)f^*$. Flow $\bar{f}$ routes the full demand of $d(i)$ units of flow between s_i and t_i for each $1 \leq i \leq k$ while sending at most $(1/\beta) \cdot c(e)$ of flow on each edge $e \in E$. The total length of flow $\bar{f}$ under l is

$$\sum_{e \in E} \frac{1}{c(e)} \cdot \bar{f}_e \leq \frac{m}{\beta}.$$

It is not hard to see that the total length of flow f is at most that of $\bar{f}$, and hence

$$\frac{1}{c(\bar{e})} \cdot f_{\bar{e}} \leq \sum_{e \in E} \frac{f_e}{c(e)} \leq \frac{m}{\beta}$$

for all edges $\bar{e} \in E$. Equivalently, the flow $f \cdot (\beta/m)$ is feasible. The maximum congestion of f is given by

$$\lambda = \max_{e \in E} \frac{f_e}{c(e)} \leq \frac{m}{\beta}.$$

Thus, the flow f/λ is feasible and has a throughput of at least β/m, i.e., $\beta \in [1/\lambda, m/\lambda]$. Using this interval for β, the total number of phases used in our algorithm becomes $T \log m$.

THEOREM 5.2. *There is an algorithm that computes a $(1+\omega)$-approximation to the maximum concurrent flow in time $O(\omega^{-2}(k \log m + m) \log m \cdot T_{\text{sp}})$ where T_{sp} is the time required to compute the shortest s-t path in a graph with nonnegative edge-weights.*

5.4. Subsequent improvements. Karakostas [15] improved the running time of the above algorithm by removing the dependence on k. This was done in a manner similar to the approach followed for maximum multicommodity flow. Thus in an iteration all commodities with the same source are considered together. The shortest path to all sinks are computed with one call to Dijkstra's algorithm and flow is routed along these paths in ratio of the demands of the various commodities. As before, in each step of the iteration, except the last, the length of at least one edge increases by a factor $1+\epsilon$. However, the number of iterations in a phase is now at most $\min(k,n)$, and hence the overall running time is $\tilde{O}(\omega^{-2}m^2)$.

6. Minimum cost multicommodity flow. Given an instance of the multicommodity flow problem, as in the previous section, edge costs $b : E \to \mathbb{R}^+$, where $b(e)$ represents the cost incurred in shipping one unit of flow along edge e, and a bound B, we consider the problem of maximizing λ subject to the additional constraint that the cost of the flow is no more than B. In the following LP formulation we use the notation introduced in section 5 and we let $b(f)$ denote the cost of a flow $f \in \mathcal{F}$ under cost function b:

$$(P_{\rm mcmcf}) \qquad \begin{aligned} \max \quad & \lambda \\ \text{s.t.} \quad & \sum_{f \in \mathcal{F}} f_e \cdot x(f) \le c(e) \quad \forall e \in E, \\ & \sum_{f \in \mathcal{F}_j} x(f) \ge \lambda \quad \forall 1 \le j \le k, \\ & \sum_{f \in \mathcal{F}} b(f) \cdot x(f) \le B, \\ & x \ge 0, \lambda \ge 0. \end{aligned}$$

Its dual has a length $l(e)$ for each edge $e \in E$, variables $z(1), \dots, z(k)$ for the throughput constraints, and a variable ϕ for the cost constraint. We will view ϕ as the *length* of the cost constraint.

$$(D_{\rm mcmcf}) \qquad \begin{aligned} \min \quad & D(l,\phi) \overset{\rm def}{=} \sum_{e \in E} c(e) l(e) + B \cdot \phi \\ \text{s.t.} \quad & \sum_{e \in E} f_e \cdot (l(e) + b(e)\phi) \ge z(j) \quad \forall 1 \le j \le k, \forall f \in \mathcal{F}_j, \\ & \sum_{j=1}^{k} z(j) \ge 1, \\ & l, z \ge 0. \end{aligned}$$

For a given $l : E \to \mathbb{R}^+$, $z(j)$ is the minimum cost of shipping $d(j)$ units of flow from s_j to t_j under cost function $l + b\phi$. Define $\alpha(l,\phi) \overset{\rm def}{=} \sum_j \texttt{min\_cost}_j(l + \phi b)$. Then $(D_{\rm mcmcf})$ can be restated as finding a length function (l, ϕ) such that $D(l,\phi)/\alpha(l,\phi)$ is minimum; let β denote this minimum value. As in the case of maximum concurrent flow we begin by assuming that $\beta \ge 1$.

Once again the algorithm proceeds in phases, each of which is composed of k iterations. In the jth iteration of the ith phase we begin with length functions $(l_{i,j-1}, \phi_{i,j-1})$ and route $d(j)$ units of commodity j. As before, for all edges e, define $l_{i+1,0}(e) = l_{i,k}(e)$ and $l_{1,0}(e) = l_{0,k}(e) = \delta/c(e)$. Similarly, $\phi_{i+1,0} = \phi_{i,k}$ and $\phi_{1,0} = \delta/B$.

The flow in each iteration is routed in a sequence of steps; in each step we route only so much flow that its cost does not exceed the bound B. Let $(l_{i,j}^{s-1}, \phi_{i,j}^{s-1})$ be the length functions at the start of the sth step (see Figure 1); the lengths at the start of the first step are given by $l_{i,j}^0 = l_{i,j-1}$ and $\phi_{i,j}^0 = \phi_{i,j-1}$. Further, let $d_{i,j}^{s-1}$ be the flow of commodity j that remains to be routed in this iteration. We compute $f_{i,j}^s \overset{\rm def}{=} \texttt{min\_cost}_j(l_{i,j}^{s-1} + b\phi_{i,j}^{s-1})$, which routes $d(j)$ units of flow of commodity j. Since we need to route only $d_{i,j}^{s-1}$ units of flow, we multiply the flow function $f_{i,j}^s$ by $d_{i,j}^{s-1}/d(j)$. If $B_{i,j}^s$ is the cost of flow $f_{i,j}^s$, then the cost of the scaled flow is

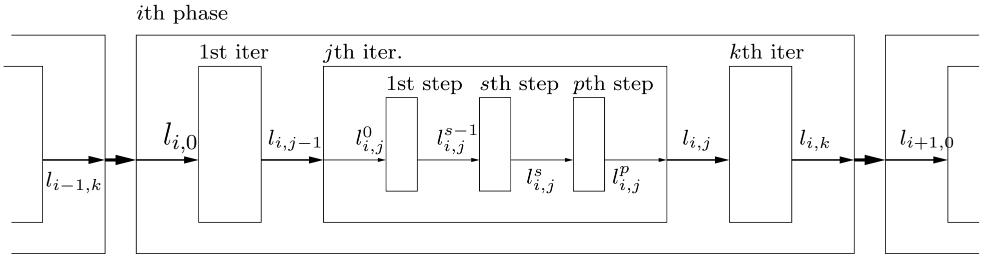


FIG. 1. *The notation used in section* 6. *The length functions above the central axis are the lengths* before *the box on the right, and the ones below are the lengths* after *the box on the left.*

$B^s_{i,j}d^{s-1}_{i,j}/d(j)$. If this quantity exceeds B, then we multiply the original flow function $f^s_{i,j}$ by $B^s_{i,j}/B$. We reuse notation and denote the final scaled flow and its cost by $f^s_{i,j}, B^s_{i,j}$, respectively. Now $f^s_{i,j}$ routes at most $d^{s-1}_{i,j}$ units of flow at cost $B^s_{i,j} \leq B$.

The length functions are modified in a similar manner as before. Thus $l^s_{i,j} = l^{s-1}_{i,j}(1+\epsilon f^s_{i,j}(e)/c(e))$ and $\phi^s_{i,j} = \phi^{s-1}_{i,j}(1+\epsilon B^s_{i,j}/B)$. Further, only $d^s_{i,j} = d^{s-1}_{i,j} - f^s_{i,j}$ more units of commodity j remain to be routed in this iteration. The iteration ends at the step p for which $d^p_{i,j} = 0$. The procedure stops at the first step at which $D()$ exceeds 1; let this happen in the tth phase.

6.1. Analysis. Note that now

$$\begin{aligned} &D(l^s_{i,j}, \phi^s_{i,j}) \\ &= D(l^{s-1}_{i,j}, \phi^{s-1}_{i,j}) + \epsilon \cdot \texttt{min\_cost}_j(l^{s-1}_{i,j} + b\phi^{s-1}_{i,j}) f^s_{i,j}/d(j) \\ &\leq D(l^{s-1}_{i,j}, \phi^{s-1}_{i,j}) + \epsilon \cdot \texttt{min\_cost}_j(l^{p}_{i,j} + b\phi^{p}_{i,j}) f^s_{i,j}/d(j), \end{aligned}$$

where the last inequality holds because the edge-lengths are monotonically increasing over steps. The total flow routed in the p steps equals the demand of commodity j, i.e., $\sum_{s=1}^p f^s_{i,j} = d(j)$. Summing over all p steps, we get

$$D(l^p_{i,j}, \phi^p_{i,j}) \leq D(l^0_{i,j}, \phi^0_{i,j}) + \epsilon \cdot \texttt{min\_cost}_j(l^p_{i,j} + b\phi^p_{i,j}).$$

The length functions at the start of the $(j+1)$th iteration are given by $l_{i,j} = l^p_{i,j}$ and $\phi_{i,j} = \phi^p_{i,j}$. Moving from steps to iterations we have

$$\begin{aligned} &D(l_{i,j}, \phi_{i,j}) \\ &\leq D(l_{i,j-1}, \phi_{i,j-1}) + \epsilon \cdot \texttt{min\_cost}_j(l_{i,j} + b\phi_{i,j}) \\ &\leq D(l_{i,j-1}, \phi_{i,j-1}) + \epsilon \cdot \texttt{min\_cost}_j(l_{i,k} + b\phi_{i,k}), \end{aligned}$$

where the last inequality uses the fact that the edge-lengths are monotonically increasing over iterations. Summing over all iterations in the ith phase, we have

$$\begin{aligned} D(l_{i,k}, \phi_{i,k}) &\leq D(l_{i,0}, \phi_{i,0}) + \epsilon \sum_{j=1}^{k} \texttt{min\_cost}_j(l_{i,k} + b\phi_{i,k}) \\ &= D(l_{i-1,k}, \phi_{i-1,k}) + \epsilon\alpha(l_{i,k}, \phi_{i,k}). \end{aligned}$$

As before we abbreviate $D(l_{i,k}, \phi_{i,k}), \alpha(l_{i,k}, \phi_{i,k})$ to $D(i), \alpha(i)$, respectively, to obtain

$$D(i) \leq D(i-1) + \epsilon\alpha(i).$$

The remainder of the analysis is exactly as in section 5.1. The only modification is in the claim about the throughput of the flow routed. Now we need to argue that the cost of the flow after we scale it by $\log_{1+\epsilon} 1/\delta$ is at most B or, equivalently, that the cost of the flow routed in the first $t-1$ iterations is at most $B \log_{1+\epsilon} 1/\delta$. This follows from the facts that $\phi_{t-1,k} < 1/B$ (since $D(t-1) < 1$), that $\phi_{1,0} = \delta/B$, and that in our procedure every time we route flow whose total cost is B we increase ϕ by at least a factor $1+\epsilon$.

6.2. Running time.

Note that except for the last step in each iteration, in all other steps we increase the length function ϕ by a factor $1+\epsilon$. This implies that the total number of steps exceeds the number of iterations by at most $\log_{1+\epsilon} 1/\delta$.

Now define z_i as the maximum possible flow of commodity i of cost no more than B. Again $z \stackrel{\text{def}}{=} \min_i z_i/d(i)$ denotes the maximum fraction of the demands that can be routed if the capacity constraints and the bound B on the cost of the flow are applied independently to each commodity. Thus $z/k \leq \beta \leq z$, and we multiply demands suitably so that for the new instance $1 \leq \beta \leq k$. As before we double the demands, thereby halving β, after every $T = 2\lceil \frac{1}{\epsilon} \log_{1+\epsilon} \frac{m}{1-\epsilon} \rceil$ phases. Thus the number of iterations is $kT \log k$, and our procedure for minimum cost multicommodity flow needs at most $(2k \log k + 1)\lceil \frac{1}{\epsilon} \log_{1+\epsilon} \frac{m}{1-\epsilon} \rceil$ single-commodity min-cost flow computations.

THEOREM 6.1. *There is an algorithm that computes a $(1+\omega)$-approximation to the maximum cost-bounded concurrent flow in time $O(\omega^{-2} k \log k \log m \cdot T_{\text{mcf}} + kT_{\text{mcbf}})$, where T_{mcf} is the time required to compute a minimum-cost s-t flow in a graph with nonnegative edge-costs and T_{mcbf} is the time required to compute the maximum s-t flow of cost at most B in a capacitated network with nonnegative edge costs.*

6.3. Avoiding min-cost flow computations.

Much like in section 5.3 we can give an alternate path-flow formulation for the minimum-cost multicommodity flow problem. In the following we let $b(p)$ denote the cost of path $p \in \mathcal{P}$:

$$(P^2_{\text{mcmcf}}) \qquad \max \quad \lambda$$

$$\text{s.t.} \quad \sum_{p \in \mathcal{P}_e} x(p) \leq c(e) \quad \forall e \in E,$$

$$\sum_{p \in \mathcal{P}_j} x(p) \geq \lambda \cdot d(j) \quad \forall 1 \leq j \leq k,$$

$$\sum_{p \in \mathcal{P}} b(p) \cdot x(p) \leq B,$$

$$x \geq 0, \lambda \geq 0.$$

Its linear programming dual has a length $l(e)$ for each edge $e \in E$, a length ϕ for the cost constraint, and a variable $z(j)$ for each commodity j:

$$(D^2_{\mathrm{mcmcf}}) \qquad \begin{aligned} \min \quad & D(l,\phi) \stackrel{\mathrm{def}}{=} \sum_{e \in E} c(e)l(e) + B \cdot \phi \\ \text{s.t.} \quad & \sum_{e \in p} (l(e) + b(e)\phi) \geq z(j) \quad \forall 1 \leq j \leq k, \forall p \in \mathcal{P}_j, \\ & \sum_{j=1}^{k} d(j) \cdot z(j) \geq 1, \\ & l, z \geq 0. \end{aligned}$$

For a given $l : E \to \mathbb{R}^+$, $z(j)$ is the shortest path between s_j and t_j under length function $l + b\phi$. We now define $\alpha(l,\phi) \stackrel{\mathrm{def}}{=} \sum_j d(j)\mathtt{dist}_j(l + b\phi)$. The dual to the min-cost multicommodity flow problem is an assignment of lengths to edges, $l : E \to \mathbb{R}^+$, and a scalar ϕ such that $D(l)/\alpha(l)$ is minimized. Let β be this minimum.

The algorithm differs from the one developed in section 6 in that at any step we route flow along only one path, which, if this is the sth step of the jth phase of the ith iteration, is the shortest path between s_j and t_j under the length function $l^{s-1}_{i,j} + b\phi^{s-1}_{i,j}$. If the minimum capacity edge on this path has capacity c, then the flow function at this step, $f^s_{i,j}$, corresponds to routing c units of flow along this path. If $c \leq d^{s-1}_{i,j}$ and the cost of this flow is less than B, we route this flow completely. Else we scale it so that the flow routed in this step has cost no more than B and the total flow routed in this iteration does not exceed $d(j)$.

The analysis of the algorithm proceeds as in section 6.1 with the only modification that $\mathtt{min\_cost}_j(.)$ is replaced with $d(j)\mathtt{dist}_j(.)$. For the running time we need only observe that in each step, except the last step in an iteration, we increase either the length of some edge or the value of ϕ by a factor $1+\epsilon$. The lengths of the edges and ϕ can each be increased by a factor $1+\epsilon$ at most $\log_{1+\epsilon} \frac{1+\epsilon}{\delta}$ times. Hence the number of steps exceeds the number of iterations by at most $(m+1)\lceil \frac{1}{\epsilon} \log_{1+\epsilon} \frac{m}{1-\epsilon} \rceil$.

Similar to section 5.3, we now describe how an idea proposed by Grigoriadis and Khachiyan [14] can be adapted to find a good estimate on the maximum throughput β subject to capacity and cost bounds. Once again, we define the length l_e of each edge $e \in E$ as $b(e)/B + 1/c(e)$. For each $1 \leq i \leq k$, let P_i be the shortest s_i, t_i-path for this length. Then define f to be the flow obtained by routing $d(i)$ units of flow along P_i for all commodities i simultaneously. As before, let f^* be an optimum cost-bounded flow with throughput β and define $\bar{f}$ as $f^* \cdot (1/\beta)$. The flow $\bar{f}$ routes $d(i)$ units of flow between the terminals of each of the k commodities.

The total length of flow $\bar{f}$ under length l is

$$\sum_{e \in E} (b(e)/B + 1/c(e)) \cdot \bar{f}_e \leq \frac{m+1}{\beta}.$$

The total length of flow f is at most that of $\bar{f}$, and hence

$$\frac{1}{c(e)} \cdot f_e \leq \frac{m+1}{\beta}$$

for all edges $e \in E$ and

$$\frac{1}{B} \cdot \sum_{e \in E} b(e) f_e \leq \frac{m+1}{\beta}.$$

The flow $(\beta/(m+1))f$ is therefore a feasible cost-bounded flow with throughput $\beta/(m+1)$.

Define the congestion of f as

$$\lambda = \max \left\{ \frac{\sum_{e \in E} b(e) f_e}{B}, \max_{e \in E} \frac{f_e}{c(e)} \right\} \leq \frac{m+1}{\beta}.$$

From the above we conclude that the optimal throughput β must be in the interval $[1/\lambda, (m+1)/\lambda]$. Using this interval for β, the total number of phases used in the algorithm becomes $T \log(m+1)$.

Theorem 6.2. *There is an algorithm that computes a $(1+\omega)$-approximation to the maximum concurrent flow in time $O(\omega^{-2}(m + k \log m) \log m \cdot T_{\text{sp}})$, where T_{sp} is the time required to compute the shortest s-t path in a graph with nonnegative edge-weights.*

6.4. Subsequent improvements. Karakostas [15] showed how to remove the dependence of the running time on k by grouping commodities with a common source. The shortest paths are now computed with respect to the length function $l + \phi b$ and only so much flow is routed that the cost of flow routed is no more than B. This leads to a $(1+\omega)$-approximation algorithm for computing the maximum cost-bounded concurrent flow in time $\tilde{O}(\omega^{-2} m^2)$.

7. Integrality. A multicommodity flow has integrality q if the flow of every commodity on every edge is a nonnegative integer multiple of q. In this section we show how small modifications to the algorithms discussed in previous sections lead to flows that have small integrality.

Our algorithm for maximum multicommodity flow routes flow along a path P in the ith iteration. If c is the minimum capacity of an edge on P, then we require that the flow routed in this iteration be no more than c. However, note that if we route $q < c$ units along P and increase the length of an edge e on P by a factor $(1+\epsilon q/c(e))$, then the algorithm still delivers a $(1-\epsilon)^{-2}$-approximation to the maximum multicommodity flow, albeit with a worse running time. To obtain a feasible flow we eventually scale the flow constructed in this manner by $\log_{1+\epsilon} 1/\delta$. Thus if we were routing q units in a certain iteration, then only $\frac{q}{\log_{1+\epsilon} 1/\delta}$ units would "appear" in the feasible solution.

Theorem 7.1. *Let e be the minimum capacity edge in G and $q \leq c(e)$. Then one can in polynomial time compute a flow f which is a $(1-\epsilon)^{-2}$-approximation to the maximum multicommodity flow and has integrality $\frac{q\epsilon}{\log_{1+\epsilon} L}$.*

Corollary 7.2. *If all edges in G have capacity at least $\frac{1}{\epsilon} \log_{1+\epsilon} L$, then there is an integral flow which is a $(1-\epsilon)^{-2}$-approximation to the maximum multicommodity flow.*

For maximum concurrent flow we use the algorithm from section 5.3. Recall that in the sth step of the jth iteration in the ith phase we route $f_{i,j}^s = \min\{c, d_{i,j}^{s-1}\}$ units of flow along path $P_{i,j}^s$, where c is the minimum capacity of an edge on this path and $d_{i,j}^{s-1}$ is the residual demand of the jth commodity. As in the case of maximum multicommodity flow we route $q < f_{i,j}^s$ units of flow in this step and increase the length of an edge e on P by a factor $(1+\epsilon q/c(e))$. To ensure that exactly q units of flow can be routed in each step of the jth iteration we require that $d(j)$ be an integral multiple of q. To obtain a feasible flow we scale the flow constructed by $\log_{1+\epsilon} 1/\delta$. Hence in the final solution the flow appears in units of $\frac{q\epsilon}{\log_{1+\epsilon} m/(1-\epsilon)}$.

THEOREM 7.3. *Let e be the minimum capacity edge in G and $q \leq c(e)$. If all demands are integral multiples of q, then one can, in polynomial time, compute a flow f which is a $(1-\epsilon)^{-3}$-approximation to the maximum concurrent flow and f has integrality $\frac{q\epsilon}{\log_{1+\epsilon} m/(1-\epsilon)}$.*

COROLLARY 7.4. *If all edges in G have capacity at least $\frac{1}{\epsilon}\log_{1+\epsilon}\frac{m}{1-\epsilon}$ and all demands are integral multiples of $\frac{1}{\epsilon}\log_{1+\epsilon}\frac{m}{1-\epsilon}$, then there is an integral flow which is a $(1-\epsilon)^{-3}$-approximation to the maximum concurrent flow.*

The above theorem and its corollary also hold for the setting of min-cost multicommodity flows.

8. Improvements in practice. In this section we propose a heuristic for our algorithms that turns out to improve running times greatly in practice. The idea is best explained with the example of the maximum multicommodity algorithm from section 2. To route $(f_i - f_{i-1})$ units of flow in iteration i, we computed k shortest paths which was later improved to one shortest path computation by Fleischer.

The idea now is to allow flow to be routed along paths which have lengths greater than the shortest path. More precisely, let l be the vector of current edge lengths and let f be the total flow routed so far. Let $\widehat{\beta}$ be an upper bound on β. We allow flow to be routed along a path P if its length is at most $L\delta e^{\epsilon f/\widehat{\beta}}$, where L, ϵ, and δ are defined as in section 2. The amount of flow routed along P equals the minimum capacity of an edge on P. The edge-lengths are updated in the same manner as before. The procedure stops when

$$1 \leq L\delta e^{\epsilon f/\widehat{\beta}}.$$

We first show that in the modified algorithm we can always find a path whose length is at most the given bound. Observe that the $\alpha(j-1)$ on the right side of (1) really denotes the length of the path along which flow was routed in the jth iteration. As our induction hypothesis we assume that this quantity is at most $\delta L e^{\epsilon f_{j-1}/\widehat{\beta}}$, which in turn is at most $\delta L e^{\epsilon f_{j-1}/\beta}$; we denote this last expression by $y(j-1)$. This implies that the length of the shortest path at the ith iteration, $\alpha(i)$, is bounded as

$$\alpha(i) \leq \delta L + \frac{\epsilon}{\beta}\sum_{j=1}^{i}(f_j - f_{j-1})y(j-1).$$

Recall the solution of the recurrence for the sequence x in section 2.1. It follows that the expression on the right is at most $\delta L e^{\epsilon f_i/\beta}$, which shows that the shortest path between any pair has length less than the specified bound.

In the original algorithm in section 5 we used the stopping condition in two ways. We argued that the length of any edge is no more than $1+\epsilon$ and that $\delta L e^{\epsilon f_t/\beta} \geq 1$. The termination condition of the modified algorithm is the same as the second property. The first property also holds since all paths along which flow was ever routed had length at most 1.

This modification to the algorithm allows one to continue sending flow on a path until its length exceeds the specified bound. Thus we can now route more flow for every shortest path computation performed. This same heuristic can be adapted to the other problems considered in this paper to obtain better running times in practice.

Acknowledgments. The authors thank Philip Klein, Cliff Stein, and Neal Young for useful discussions.

REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows*, Prentice–Hall, Englewood Cliffs, NJ, 1993.

[2] B. Awerbuch and F. T. Leighton, *Improved approximation algorithms for the multicommodity flow problem and local competitive routing in dynamic networks*, in Proceedings of the ACM Symposium on Theory of Computing, ACM, New York, 1994, pp. 487–496.

[3] D. Bienstock, *Potential Function Methods for Approximately Solving Linear Programming Problems: Theory and Practice*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.

[4] D. Bienstock and G. Iyengar, *Approximating fractional packings and coverings in $O(1/\epsilon)$ iterations*, SIAM J. Comput., 35 (2006), pp. 825–854.

[5] G. Even, J. Naor, S. Rao, and B. Schieber, *Fast approximate graph partitioning algorithms*, SIAM J. Comput., 28 (1999), pp. 2187–2214.

[6] G. Even, S. Naor, S. Rao, and B. Schieber, *Divide-and-conquer approximation algorithms via spreading metrics*, J. ACM, 47 (2000), pp. 585–616.

[7] L. K. Fleischer, *Approximating fractional multicommodity flow independent of the number of commodities*, SIAM J. Discrete Math., 13 (2000), pp. 505–520.

[8] L. K. Fleischer and K. D. Wayne, *Fast and simple approximation schemes for generalized flow*, Math. Program., 91 (2002), pp. 215–238.

[9] N. Garg and J. Könemann, *Faster and Simpler Algorithms for Multicommodity Flow and Other Fractional Packing Problems*, Technical report 97-1-025, Max-Planck Institut für Informatik, Saarbrücken, Germany, 1997.

[10] N. Garg and J. Könemann, *Faster and simpler algorithms for multicommodity flow and other fractional packing problems*, in Proceedings of the IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 1998, pp. 300–309.

[11] A. V. Goldberg, *A natural randomization strategy for multicommodity flow and related algorithms*, Inform. Process. Lett., 42 (1992), pp. 249–256.

[12] M. Grigoriadis and L. G. Khachiyan, *Approximate minimum-cost multicommodity flows in $\tilde{O}(\epsilon^{-2}knm)$ time*, Math. Programming, 75 (1996), pp. 477–482.

[13] M. D. Grigoriadis and L. G. Khachiyan, *Fast approximation schemes for convex programs with many blocks and coupling constraints*, SIAM J. Optim., 4 (1994), pp. 86–107.

[14] M. D. Grigoriadis and L. G. Khachiyan, *Coordination complexity of parallel price-directive decomposition*, Math. Oper. Res., 21 (1996), pp. 321–340.

[15] G. Karakostas, *Faster approximation schemes for fractional multicommodity flow problems*, in Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, SIAM, Philadelphia, 2002, pp. 166–173.

[16] D. Karger and S. Plotkin, *Adding multiple cost constraints to combinatorial optimization problems, with applications to multicommodity flows*, in Proceedings of the ACM Symposium on Theory of Computing, ACM, New York, 1995, pp. 18–25.

[17] P. Klein, S. Plotkin, C. Stein, and É. Tardos, *Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts*, SIAM J. Comput., 23 (1994), pp. 466–487.

[18] T. Leighton, F. Makedon, S. Plotkin, C. Stein, S. Tragoudas, and É. Tardos, *Fast approximation algorithms for multicommodity flow problems*, J. Comput. System Sci., 50 (1995), pp. 228–243.

[19] Yu. Nesterov, *Smooth minimization of non-smooth functions*, Math. Program., 103 (2005), pp. 127–152.

[20] S. Plotkin, D. Shmoys, and É. Tardos, *Fast approximation algorithms for fractional packing and covering problems*, Math. Oper. Res., 20 (1995), pp. 257–301.

[21] T. Radzik, *Fast deterministic approximation for the multicommodity flow problem*, in Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, SIAM, Philadelphia, 1995, pp. 486–492.

[22] F. Shahrokhi and D. Matula, *The maximum concurrent flow problem*, J. ACM, 37 (1990), pp. 318–334.

[23] C. Stein, *Approximation Algorithms for Multicommodity Flow and Scheduling Problems*, Ph.D. thesis, MIT, Cambridge, MA, 1992.

[24] P. M. VAIDYA, *Speeding up linear programming using fast matrix multiplication*, in Proceedings of the IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 1989, pp. 332–337.

[25] N. YOUNG, *Randomized rounding without solving the linear program*, in Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, SIAM, Philadelphia, 1995, pp. 170–178.

[26] N. E. YOUNG, *Sequential and parallel algorithms for mixed packing and covering*, in Proceedings of the IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 2001, pp. 538–546.

SIAM J. COMPUT.
Vol. 37, No. 2, pp. 653–670

APPROXIMATION ALGORITHMS FOR ORIENTEERING AND DISCOUNTED-REWARD TSP[*]

AVRIM BLUM[†], SHUCHI CHAWLA[‡], DAVID R. KARGER[§], TERRAN LANE[¶], ADAM MEYERSON[‖], AND MARIA MINKOFF[**]

Abstract. In this paper, we give the first constant-factor approximation algorithm for the rooted ORIENTEERING problem, as well as a new problem that we call the DISCOUNTED-REWARD traveling salesman problem (TSP), motivated by robot navigation. In both problems, we are given a graph with lengths on edges and rewards on nodes, and a start node s. In the ORIENTEERING problem, the goal is to find a path starting at s that maximizes the reward collected, subject to a hard limit on the total length of the path. In the DISCOUNTED-REWARD TSP, instead of a length limit we are given a discount factor γ, and the goal is to maximize the total discounted reward collected, where the reward for a node reached at time t is discounted by γ^t. This problem is motivated by an approximation to a planning problem in the Markov decision process (MDP) framework under the commonly employed infinite horizon discounted reward optimality criterion. The approximation arises from a need to deal with exponentially large state spaces that emerge when trying to model one-time events and nonrepeatable rewards (such as for package deliveries). We also consider tree and multiple-path variants of these problems and provide approximations for those as well. Although the unrooted ORIENTEERING problem, where there is no fixed start node s, has been known to be approximable using algorithms for related problems such as k-TSP (in which the amount of reward to be collected is fixed and the total length is approximately minimized), ours is the first to approximate the rooted question, solving an open problem in [E. M. Arkin, J. S. B. Mitchell, and G. Narasimhan, *Proceedings of the* 14*th ACM Symposium on Computational Geometry*, 1998, pp. 307–316] and [B. Awerbuch, Y. Azar, A. Blum, and S. Vempala, *SIAM J. Comput.*, 28 (1998), pp. 254–262]. We complement our approximation result for ORIENTEERING by showing that the problem is APX-hard.

Key words. approximation algorithms, traveling salesman problem, prize-collecting traveling salesman problem, orienteering, robot navigation, Markov decision processes

AMS subject classifications. 68W25, 68W40

DOI. 10.1137/050645464

1. Introduction.

Consider a robot with a map of its environment that needs to visit a number of sites to drop off packages, collect samples, search for a lost item, etc. One classic model of such a scenario is the traveling salesman problem (TSP), in which we ask for the tour that visits all the sites and whose length is as short as possible.

[*] Received by the editors November 14, 2005; accepted for publication (in revised form) February 2, 2007; published electronically June 15, 2007. A preliminary version of this paper appeared in *Proceedings of the* 44*th Annual IEEE Symposium on Foundations of Computer Science* (FOCS 2003), Cambridge, MA, 2003, pp. 46–55.

http://www.siam.org/journals/sicomp/37-2/64546.html

[†] Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213 (avrim@cs.cmu.edu). This author's research was supported by NSF grants CCR-0105488, IIS-0121678, and CCR-0122581.

[‡] Computer Science Department, University of Wisconsin-Madison, Madison, WI 53706 (shuchi@cs.wisc.edu). This work was done while this author was a graduate student at CMU and was supported by NSF grants CCR-0105488, IIS-0121678, and CCR-0122581.

[§] MIT Computer Science and Artificial Intelligence Laboratory (CSAIL), Cambridge, MA 02139 (karger@theory.csail.mit.edu).

[¶] Department of Computer Science, University of New Mexico, Albuquerque, NM 87131 (terran@cs.unm.edu).

[‖] Computer Science Department, University of California, Los Angeles, Los Angeles, CA 90095 (awm@cs.ucla.edu). This work was done while this author was visiting Carnegie Mellon University.

[**] Max-Planck-Institut für Informatik, Saarbrucken 66123, Germany (mariam@theory.csail.mit.edu). This work was done while this author was a graduate student at MIT.

However, what if this robot cannot visit everything? For example, it might have a limited supply of battery power. In that case, a natural problem is to find the tour that visits the maximum total reward of sites (where the reward might correspond to the value of a package being delivered or the probability that some lost item we are searching for is located there), subject to a constraint that the total length is at most some given bound B. This is called the (rooted) ORIENTEERING problem ("rooted" because we are fixing the starting location of the robot). Interestingly, while there have been a number of algorithms that given a desired reward can approximately minimize the distance traveled (which yield approximations to the unrooted ORIENTEERING problem), approximating the reward for the case of a *fixed* starting location and *fixed* hard length limit has been an open problem.

Alternatively, suppose that battery power is not the limiting consideration, but we simply want to give the robot a penalty for taking too long to visit high-value sites. For example, if we are searching for a lost item, and at each time step there is some possibility the item will be taken (or, if we are searching for a trapped individual in a dangerous environment, and at each time step there is some probability the individual might die), then we would want to discount the reward for a site reached at time t by γ^t, where γ is a known discount factor. We call this the DISCOUNTED-REWARD TSP. This problem is motivated by an approximation to a planning problem in the *Markov decision process* (MDP) framework [26, 25] under the commonly employed *infinite horizon discounted reward* optimality criterion. The approximation arises from a need to deal with exponentially large state spaces that emerge when trying to model one-time events and nonrepeatable rewards (such as for package deliveries).

In this paper, we provide the first constant-factor approximations to both the (rooted) ORIENTEERING and the DISCOUNTED-REWARD TSP problems, as well as a number of variants that we discuss below. We also prove that ORIENTEERING is APX-hard, or NP-hard to approximate within an arbitrarily small constant factor.

1.1. Motivation and background. Robot navigation and path planning problems can be modeled in many ways. In the theoretical computer science and optimization communities, these are typically modeled as kinds of prize-collecting TSPs [19, 4, 17, 3]. In the artificial intelligence community, problems of this sort are often modeled as MDPs [6, 7, 21, 25, 26]. Below we give some background and motivation for our work from each perspective.

1.1.1. MDPs and time-dependent rewards. An MDP consists of a state space S, a set of actions A, a probabilistic transition function T, and a reward function R. For this work, it is sufficient to consider discrete, finite S and A. At any given time step, an agent (such as a robot) acting in an MDP will be located at some state $s \in S$, where it can choose an action $a \in A$. The agent is subsequently relocated to a new state s' drawn from the transition probability distribution $T(s'|s,a) \equiv \Pr[q_{t+1} = s'|q_t = s, a]$, where q_t is a random variable indicating the agent's state at time step t. The transition function captures both the agent's stochastic dynamics (e.g., unreliable actuators) and structure and characteristics of the environment such as walls, pits, friction of the surface, etc. Associated with each state is a real-valued reward, given by the function $R(s)$, which the agent receives upon entering state s.[1] For example, a

[1] It is also possible to model rewards associated with actions or transitions by writing more general reward functions such as $R(s,a)$ or $R(s,a,s')$, but such extensions do not fundamentally change the nature of the MDP. Any such functions can be rewritten into a model of the form we give here with an appropriate modification to the state and action sets.

package-delivery robot might get a reward every time it correctly delivers a package.

The goal of planning in an MDP framework is to formulate a policy, $\psi : S \rightarrow A$, that guides the agent to optimal long-term aggregate reward. In order to encourage the agent to perform the tasks that we want, and to do so in a timely manner, a commonly employed aggregate reward objective function is the *infinite horizon discounted reward* [21, 25, 26]. Specifically, for a given *discount factor* $\gamma \in (0, 1)$, the value of the reward collected at time t is discounted by a factor γ^t. Thus the total discounted reward, which we aim to maximize, is $R_{tot} = \sum_{t=0}^{\infty} R(s_t)\gamma^t$. Because the agent's actions are stochastic, in practice we must settle for optimizing the expected value of this quantity, $V^{\psi}(s) = \mathbf{E}_{\psi}[R_{tot}|q_0 = s]$, where the expectation is taken with respect to all possible trajectories through the state space rooted at state s, weighted by their probability of occurring under policy ψ. Note that because a fixed (s, a) pair yields a fixed probability distribution over next states, the combination of an MDP with a fixed policy produces a Markov chain over S. The expectation, therefore, is simply the expected discounted reward accumulated by a random walk on the corresponding Markov chain. This optimality criterion guides the agent to accumulate as much reward as possible as early as possible, and produces what in practice turns out to be good behavior.

One can also motivate exponential discounting by imagining that, at each time step, there is some fixed probability the game will end (the robot loses power, a catastrophic failure occurs, the objectives change, etc.). The quantity $V^{\psi}(s)$ then gives the expected (undiscounted) reward collected by the robot before the game ends. Exponential discounting also has the nice mathematical property that it is *time-independent*, meaning that an optimal strategy is *stationary* and can be completely described by the mapping from states to actions given by ψ.[2] The overall goal of planning, then, is to locate ψ^*, the policy that maximizes $V^{\psi}(s)$, the expected infinite horizon discounted reward. A fundamental theorem of MDP planning states that for this optimality criterion, there is guaranteed to be a stationary ψ^* that dominates all other policies at all states: $V^{\psi^*}(s) \geq V^{\psi}(s)$ for all $s \in S$ and all ψ [25].

There are well-known algorithms for solving MDPs in time polynomial in the cardinality of the state space [6, 25, 26]. However, one drawback of the MDP model is that the agent receives $R(s)$ every time that state s is visited. Thus, in order to model a package-delivery or search-and-rescue robot, one would need a state representing not only the current location of the robot but also a record of all packages (victims) it has already delivered (rescued). For example, one could write $S = L \times 2^d$, where L is a set of discrete locations that the robot could occupy, and the list of d bits tracks whether the agent has achieved each of d subgoals (packages or rescues). Then the reward function can be $R(\langle l, b_1, \ldots, b_d\rangle) = 1$ iff l is a location containing subgoal i and $b_i = 0$, or $R(s) = 0$ otherwise. When the robot reaches the location containing subgoal i, b_i is set to 1 and remains so thereafter. This formulation yields an exponential increase in the size of the state space over the raw cardinality of L and prevents a direct, exact solution of the MDP. Instead, it would be preferable to directly model the case of rewards that are given only the *first* time a state is visited [22, 23].

As a first step towards tackling this general problem, we abandon the stochastic element and restrict the model to deterministic, reversible actions. This model is a reasonable approximation to many robot-navigation style MDP domains, in which we can formulate subpolicies for navigating between pairs of locations in the environ-

[2] Under other objective functions, an optimal policy could require dependence on the number of steps remaining in the game or other functions of the history of states encountered to date.

ment. Often, such subpolicies, or macros, can be "nearly deterministic" (failing with probability $\leq \epsilon$) because they average out the stochasticity of atomic actions over many steps [23]. We can to a good approximation, therefore, treat such a domain as a deterministic planning problem over the set of subgoal locations (nodes) and location-to-location macros (arcs). This leads us to study the DISCOUNTED-REWARD TSP, in which we assume we have an undirected weighted graph (edge weights represent the time to traverse a given edge), with a reward value π_v on each vertex v, and our goal is to find a path visiting each vertex v at time t_v so as to maximize $\sum \pi_v \gamma^{t_v}$.

1.1.2. Prize-collecting TSP and ORIENTEERING problems. A different way to model the goal of collecting as much reward as possible as early as possible is to impose a hard deadline on the time the robot may spend delivering its packages. The robot gets a reward equal to the value of the package on a delivery but only if the delivery is made before a deadline D. If the deadline is exceeded, he gets no reward. This problem has been studied previously as the ORIENTEERING problem [19] or bank robber problem [3].

ORIENTEERING belongs to the family of the prize-collecting traveling salesman problems (PCTSPs). Given a set of cities with nonnegative "prize" values associated with them and a table of pairwise distances, a salesman needs to pick a subset of the cities to visit so as to minimize the total distance traveled while maximizing the total amount of prize collected. Note that there is a trade-off between the cost of a tour and how much prize it spans. The original version of the PCTSP introduced by Balas [4] deals with these two conflicting objectives by combining them: one seeks a tour that minimizes the *sum* of the total distance traveled and the penalties (prizes) on cities skipped, while collecting at least a given quota amount of prize. Bienstock et al. [8] subsequently focused on a special case of this problem in which the quota requirement is dropped, and provided a constant-factor approximation for it. This was further improved to a 2-approximation via a primal-dual approach by Goemans and Williamson [18].

An alternative approach to the bicriterion optimization is to optimize just one of the objectives while enforcing a fixed bound on the other. For example, in a quota-driven version of the PCTSP, called k-TSP, every node has a prize of one unit and the goal is to minimize the total length of the tour, while visiting at least k nodes. Similarly, ORIENTEERING can be viewed as a budget-driven version of the PCTSP, since we are maximizing the total amount of prize collected, while keeping the distance traveled below a certain threshold.[3]

There are several constant-factor approximations known for the k-TSP problem [2, 15, 9, 3], the best being a recent 2-approximation due to Garg [16]. Most of these results are based on a classic primal-dual algorithm for the PCTSP due to Goemans and Williamson [18] (mentioned above).

The algorithms for k-TSP extend easily to the *unrooted* version of the ORIENTEERING problem in which we do not fix the starting location [3, 16]. In particular, suppose that we want to find a tour (cycle) of length D that spans value Π, and using a c-approximation to the k-TSP, for some $c > 1$, we obtain a cycle of length cD spanning value Π. Then we can just break this cycle into $2c$ pieces of length at most $D/2$ each, take the piece spanning the most value, and convert it into a cycle of length at most D (by traversing the path forward and then back). Noting that the cycle

[3] Strictly speaking, a budget-driven version of the PCTSP would require a tour, e.g., a path that ends at the start node, whereas the ORIENTEERING problem is content with a path that ends at an arbitrary node. We consider both versions of the problem.

spans at least $\Pi/2c$ value, we get a $2c$-approximation to unrooted ORIENTEERING. However, this does not work for the rooted problem because the "best piece" in the above reduction might be far from the start. In contrast to this simple result, there is no previously known $O(1)$-approximation algorithm for the rooted ORIENTEERING problem in general graphs. Arkin, Mitchell, and Narasimhan [1] give a constant-factor approximation to the rooted ORIENTEERING problem for the special case of points in the plane.

1.2. Summary of results. In this paper, we give the first constant-factor approximation algorithms for both of the above problems. A key contribution of our work is the introduction of the *min-excess* objective. The excess of a path is defined to be the difference between the length of a prize-collecting s-t path and the length of the shortest path between the two endpoints. Informally, any path must spend a minimum amount of time equal to the shortest distance between s and t, just to get to the destination t; the excess of the path is the extra time spent by it to gather reward along the way. We consider the following MIN-EXCESS-PATH problem: given a weighted graph with rewards, endpoints s and t, and a reward quota k, find a minimum excess path from s to t collecting reward at least k.

Approximating the excess of a path turns out to be a crucial component in our algorithms for ORIENTEERING and DISCOUNTED-REWARD TSP. Note that an approximation for the excess is a strictly better guarantee than what is implied by an approximation algorithm for the k-TSP; the latter would return a path that has length at most a constant multiple times the *total* optimal length from s to t.

Our algorithm for approximating MIN-EXCESS-PATH uses as a subroutine an approximation to a variant of the k-TSP, the min-cost s-t path problem (k-PATH problem in [10]). In particular, an α_{CP}-approximation to the k-PATH, when used as a subroutine in our algorithm, implies an ($\alpha_{EP} = \frac{3}{2}\alpha_{CP} - \frac{1}{2}$)-approximation for the MIN-EXCESS-PATH problem. The best currently known approximation for the k-PATH problem is a $(2+\delta)$-approximation (for any fixed $\delta > 0$), which follows from a $(2+\delta)$-approximation to the k-TSP due to Chaudhuri et al. [10].[4] Then via our reduction, this implies a $(2.5+\delta)$-approximation to excess, for any fixed $\delta > 0$. We also present an improved analysis of our algorithm based on the Chaudhuri et al. k-TSP algorithm obtaining a $(2+\delta)$-approximation for the MIN-EXCESS-PATH problem.

An α_{EP}-approximation to MIN-EXCESS-PATH further implies a $(1 + \lceil\alpha_{EP}\rceil)$-approximation for ORIENTEERING and a roughly $e(\alpha_{EP} + 1)$-approximation for DISCOUNTED-REWARD TSP. Our final approximation factors for the latter two problems are 4 and $6.75 + \delta$, respectively.

Finally, using the APX-hardness of the TSP on bounded metrics [14], we prove that MIN-EXCESS-PATH and ORIENTEERING are APX-hard.

1.3. Subsequent work. Following the initial publication of our work, Bansal et al. [5] obtained a 3-approximation for a stronger version of ORIENTEERING called "point-to-point ORIENTEERING," in which the starting location s as well as the terminal location t are fixed. They also consider the vehicle routing problem with time-windows, a generalization of ORIENTEERING in which each reward has a time-window ("release time" and "deadline") associated with it, and reward is earned only if the location is visited within the corresponding time-window. Bansal et al. obtain an

[4] Garg's 2-approximation algorithm for the k-TSP, although better that Chaudhuri et al.'s $(2+\delta)$-approximation, implies only a 5-approximation to the k-PATH problem.

$O(\log^2 n)$-approximation for this problem, as well as an $O(\log n)$-approximation when all the release times are zero.

Organization. The rest of this paper is organized as follows. We begin with some definitions in section 2. Then we give an algorithm for MIN-EXCESS-PATH in section 3, followed by algorithms for DISCOUNTED-REWARD TSP and ORIENTEERING in sections 4 and 5, respectively. In section 6, we extend some of the algorithms to tree and multiple-path versions of the problems. We present some hardness of approximation results in section 7 and conclude in section 8.

2. Notation and definitions.

Our work encompasses a variety of problems. In this section, we introduce the notation to be used throughout the paper, provide formal problem statements, and describe a uniform naming scheme for them.

Let $G = (V, E)$ be a weighted undirected graph, with a distance function on edges, $d : E \to \Re^+$, and a *prize* or *reward* function on nodes, $\pi : V \to \Re^+$. Let $\pi_v = \pi(v)$ be the reward on node v. Let $s \in V$ denote a special node called the *start* or *root*.

For a path P visiting u before v, let $d^P(u, v)$ denote the length along P from u to v. Let $d(u, v)$ denote the length of the *shortest* path from node u to node v. For ease of notation, let $d_v = d(s, v)$ and $d^P(v) = d^P(s, v)$. For a set of nodes $V' \subseteq V$, let $\Pi(V') = \sum_{v \in V'} \pi_v$. For a set of edges $E' \subseteq E$, let $d(E') = \sum_{e \in E'} d(e)$.

Our problems aim to construct a certain subgraph—a path, tree, or cycle, possibly with additional constraints. Most of the problems attempt a trade-off between two objective functions: the *cost* (distance) of the path (or tree, or cycle), and the *total prize* spanned by it. From the point of view of exact algorithms, we need simply to specify the cost we are willing to tolerate and the prize we wish to span. Most variants of this problem, however, are $\mathcal{NP}$-hard, and so we focus on approximation algorithms. We must then specify our willingness to approximate the two distinct objectives. We refer to a *min-cost* problem when our goal is to *approximately* minimize the cost of our objective subject to a fixed lower bound on prize (thus, prize is a feasibility constraint, while our approximated objective is cost). Conversely, we refer to a *max-prize* problem when our goal is to *approximately* maximize the prize collected subject to a fixed upper bound on cost (thus, cost is a feasibility constraint, while our approximated objective is prize). For example, the min-cost tree problem is the traditional k-MST: it requires spanning k units of prize and aims to minimize the cost of doing so. Both the rooted and unrooted min-cost tree problems have constant-factor approximations [20, 2, 15, 9, 3]. The max-prize path problem, which aims to find a path of length at most D from the start node s that visits a maximum amount of prize, is the ORIENTEERING problem.

The main subroutine in our algorithms also requires introducing a variation on approximate cost. Define the *excess* of a path P from s to t to be $d^P(s, t) - d(s, t)$, that is, the difference between that path's length and the distance between s and t in the graph. Obviously, the minimum excess path of total prize Π is also the minimum *cost* path of total prize Π; however, a path of a constant factor times minimum cost may have more than a constant factor times the minimum excess. We therefore consider separately the *minimum excess path* problem. Note that an (s, t) path approximating the optimum excess ϵ by a factor α will have length $d(s, t) + \alpha\epsilon \leq \alpha(d(s, t) + \epsilon)$ and therefore approximates the minimum cost path by a factor α as well. Achieving a good approximation to this MIN-EXCESS-PATH problem will turn out to be a key ingredient in our approximation algorithms.

Finally, as discussed earlier, we consider a different means of combining length and cost motivated by applications of MDPs. We introduce a *discount factor* $\gamma < 1$.

Table 2.1
Approximation factors and reductions for our problems.

Problem	Best approx.	Source/Reduction	Hardness of approx.
min-cost s-t path (k-Path)	$\alpha_{CP} = 2 + \delta$	[10]	1.0046
min-excess path (Min-Excess-Path)	$\alpha_{EP} = 2.5 + \delta$ $\alpha_{EP} = 2 + \delta$	$\frac{3}{2}(\alpha_{CP}) - \frac{1}{2}$ algo. based on [10]	1.0046
max-discounted-prize path (Disctd-Reward TSP)	$\alpha_{DP} = 6.75 + \delta$	$\alpha_{EP}^{-\alpha_{EP}}(1 + \alpha_{EP})^{1+\alpha_{EP}}$	?
max-prize path (Orienteering)	$\alpha_{PP} = 4$	$1 + \lceil \alpha_{EP} \rceil$	1.00068
max-prize tree	$\alpha_{PT} = 8$	$2\alpha_{PP}$	?
max-prize cycle	$\alpha_{PC} = 8$	$2\alpha_{PP}$	1.00068
max-prize multiple-path	$\alpha_{kPP} = 5$	$\alpha_{PP} + 1$	1.00068

Given a path P rooted at s, let the *discounted reward* collected at node v by path P be defined as $\rho_v^P = \pi_v \gamma^{d^P(s,v)}$. That is, the prize gets discounted exponentially by the amount of time it takes for the path to reach node v. The *max-discounted-reward* problem is to find a path P rooted at s that maximizes $\rho^P = \sum_{v \in P} \rho_v^P$. We call this the Discounted-Reward TSP. Note that the length of the path is not specifically bounded in this problem, though of course shorter paths produce less discounting.

2.1. Reductions and approximation factors. We present a constant-factor approximation algorithm for the max-prize path (rooted Orienteering) problem, solving an open problem of [3, 1], as well as the Discounted-Reward TSP. Central to our results is a constant-factor approximation for the Min-Excess-Path problem defined above, which uses an algorithm for the min-cost s-t path problem as a subroutine. We also give constant-factor approximations to several related problems, including the max-prize tree problem—the "dual" to the k-MST (min-cost tree) problem—and max-prize cycle. Specific constants are given in Table 2.1. For the Min-Excess-Path problem, we derive a tighter analysis and an improved approximation of $2+\delta$ in section 3.2, based on the min-cost s-t path algorithm of [10]. This improvement gives a better approximation factor of $6.75 + \delta$ for the Discounted-Reward TSP.

Our approximation algorithms reflect a series of reductions from one approximation problem to another. Improvements in the approximations for various problems will propagate through. We state approximation factors in the form α_{XY}, where XY denotes the problem being approximated; the first letter denotes the objective (cost, prize, excess, or discounted prize denoted by C, P, E, and D, respectively), and the second letter denotes the structure (path, cycle, or tree denoted by P, C, or T, respectively).

2.2. Preliminaries. To support dynamic programming in the max-prize variants, we assume that all prizes are integers in the range $\{1, \ldots, n^2\}$—this allows us to "guess" the reward collected by the optimal solution by trying out all integer values less than n^3. We can make this assumption by scaling the values such that the maximum value is exactly n^2 (this guarantees that the optimal solution gets at least n^2 reward). We then round each value down to its nearest integer, losing an additive amount of at most n, which is a negligible multiplicative factor. This negligible factor does mean that an approximation algorithm for a max-prize problem with guarantee c on polynomially bounded inputs has (weaker) guarantee "arbitrarily close to c" on arbitrary inputs. For the MIN-EXCESS-PATH problem, we do not make this bounded-value assumption. This implies that the running time of our algorithm is linear in the total value in the graph (with all the values in the graph being integral). Note, however, that in our algorithms for the max-prize problems, we may use a bicriteria version of MIN-EXCESS-PATH in which we approximate the value obtained to within a $(1 + O(1/n))$ factor and excess to within an α_{EP} factor. We may then use the bounded value assumption, and our running time is again bounded by a polynomial in n.

3. MIN-EXCESS-PATH.

Let P^* be the shortest path from s to t with $\Pi(P^*) \geq k$. Let $\epsilon(P^*) = d(P^*) - d(s,t)$. Our algorithm returns a path P with $\Pi(P) \geq k$ and length $d(P) = d(s,t) + \alpha_{EP}\epsilon(P^*)$, where $\alpha_{EP} = \frac{3}{2}\alpha_{CP} - \frac{1}{2}$. Thus we obtain a $(2.5 + \delta)$-approximation to MIN-EXCESS-PATH using an algorithm of Chaudhuri et al. [10] for min-cost s-t path (MCP) with $\alpha_{CP} = 2 + \delta$.

We begin with a brief description of the min-cost path algorithm and approximation. In their paper, Chaudhuri et al. provide a subroutine for constructing a tree containing nodes s and t that spans at least k vertices[5] and has cost at most $(1 + \frac{1}{2}\delta)$ times the cost of the shortest s-t path with k vertices, for any fixed constant δ. To construct an s-t path from the tree obtained by the algorithm of Chaudhuri et al., we can double all the edges, except those along the tree path from s to t. This gives us a partial "Euler tour" of the tree that starts at s and ends at t. Clearly, the cost of such a path is at most $(2 + \delta)$ times the cost of the shortest s-t path spanning prize k, for any fixed constant δ.

Now we return to the harder MIN-EXCESS-PATH problem. The idea for our algorithm for MIN-EXCESS-PATH is as follows. Suppose that the optimum solution path encounters all its vertices in increasing order of distance from s. We call such a path *monotonic*. We can find this optimum monotonic path via a simple dynamic program: for each possible prize value p and for each vertex i in increasing order of distance from s, we compute the minimum excess path that starts at vertex s, ends at i, and collects prize at least p.

We solve the general case by breaking the optimum path into *segments* that are either monotonic (and thus can be found optimally as just described) or "wiggly" (generating a large amount of excess). We show that the total length of the wiggly portions is comparable to the excess of the optimum path; our solution uses the optimum monotonic paths and approximates the length of the wiggly portions by a constant factor, yielding an overall increase proportional to the excess.

Consider the optimal path P^* from s to t. We divide it into segments in the following manner. For any real d, define $f(d)$ as the number of edges on P^* with one endpoint at distance less than d from s and the other endpoint at distance at least d from s. Note that $f(d) \geq 1$ for all $0 \leq t \leq d_t$ (it may also be nonzero for some $d \geq d_t$). Note also that f is piecewise constant, changing only at distances equal to vertex distances. We break the real line into intervals according to f: the *type* 1 *intervals* are the maximal intervals on which $f(d) = 1$; the *type* 2 *intervals* are the maximal intervals on which $f(d) \geq 2$. These intervals partition the real line (out to the maximum distance reached by the optimum solution) and alternate between types 1 and 2. Let the interval boundaries be labeled $0 = b_1 < b_2 \cdots b_m$, where b_m is the maximum distance of any vertex on the path, so that the ith interval is (b_i, b_{i+1}).

[5] The algorithm can be transformed easily to obtain a tree spanning a given target *prize value*—to each node v with a prize π_v, we attach $\pi_v - 1$ leaves and run the algorithm on this new graph.

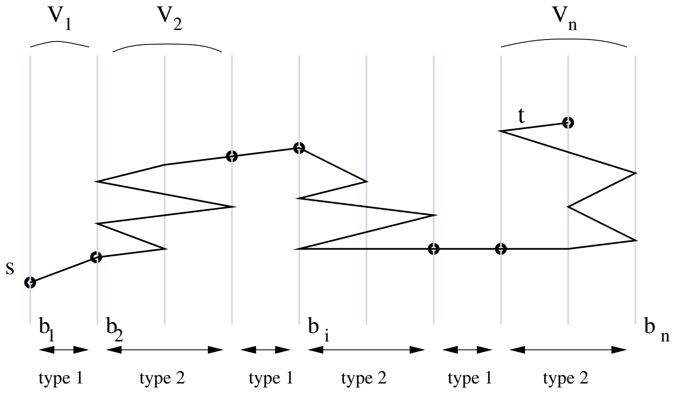


FIG. 3.1. *Segment partition of a path in graph G. Vertices are laid out in order of their distances from s. The vertical dotted lines indicate distances from s.*

Note that each b_i is the distance label for some vertex. Let V_i be the set of vertices whose distance from s falls in the ith interval. Note that the optimum path traverses each set V_i exactly once—once it leaves some V_i, it does not return. One of any two adjacent intervals is of type 1; if the path left this interval and returned to it, then $f(d)$ would exceed 1 within the interval. Thus, the vertices of P^* in set V_i form a contiguous *segment* of the optimum path that we label as $S_i = P^* \cap V_i$.

A segment partition is shown in Figure 3.1.

Note that for each i, there may be (at most) 1 edge crossing from V_i to V_{i+1}. To simplify the next two lemmas, let us split that edge into two with a vertex at distance b_i from s, so that every edge is completely contained in one of the segments (this can be done since one endpoint of the edge has distance exceeding b_i and the other endpoint has distance less than b_i).

LEMMA 3.1. *A segment S_i of type 1 has length at least $b_{i+1} - b_i$. A segment S_i of type 2 has length at least $3(b_{i+1} - b_i)$, unless it is the segment containing t, in which case it has length at least $3(d_t - b_i)$.*

Proof. The length of segment S_i is lower bounded by the integral of $f(d)$ over the ith interval. In a type 1 interval, the result is immediate. For a type 2 interval, note that $f(d) \geq 1$ actually implies that $f(d) \geq 3$ by a parity argument—if the path crosses distance d twice only, it must end up at distance less than d. □

COROLLARY 3.2. *The total length of type 2 segments is at most $3\epsilon(P^*)/2$.*

Proof. Let ℓ_i denote the length of segment i. We know that the length of P^* is $d_t + \epsilon(P^*) = \sum \ell_i$. At the same time, we can write

$$d_t \leq b_m = \sum_{i=1}^{m-1} (b_{i+1} - b_i) \leq \sum_{i \text{ type } 1} \ell_i + \sum_{i \text{ type } 2} \ell_i/3.$$

It follows that

$$\epsilon(P^*) = \sum \ell_i - d_t \geq \sum_{i \text{ type } 2} 2\ell_i/3.$$

Multiplying both sides by 3/2 completes the proof. □

Having completed this analysis, we note that the corollary remains true even if we do not introduce extra vertices on edges crossing interval boundaries. The crossing edges are no longer counted as parts of segments, but this decreases only the total length of type 2 segments.

3.1. A dynamic program. Our algorithm computes, for each interval that might be an interval of the optimum solution, a segment corresponding to the optimum solution in that interval. It then uses a dynamic program to paste these fragments together using (and paying for) edges that cross between segments. The segments we compute are defined by four vertices: the closest-to-s and farthest-from-s vertices, c and f, in the interval (which define the start- and endpoints of the interval: our computation is limited to vertices within that interval), and the first and last vertices, x and y, on the segment within that interval. They are also defined by the amount p of prize we are required to collect within the segment. There are therefore $O(\Pi n^4)$ distinct segments to compute, where Π is the total prize in the graph. For each segment, we find an optimum solution for a type 1 and a type 2 interval. For a type 1 interval, the optimum path is monotonic; we can therefore compute (in linear time) an optimum (shortest) monotonic path from x to y that collects prize p. If the interval is of type 2, the optimum path need not be monotonic. Instead, we use the MCP routine to approximate to within a constant factor the minimum length of a path that starts at x, finishes at y, stays within the boundaries of the interval defined by c and f, and collects prize at least p. Let $A_1(x, y, p)$ and $A_2(x, y, c, f, p)$ denote the lengths of the type 1 and type 2 segments computed above for each interval. (Note that for type 1 segments, $c = x$ and $f = y$.)

Given the optimum type 1 and near-optimum type 2 segment determined for each set of four vertices and prize value, we can find the optimal way to paste some subset of them together monotonically using a dynamic program. Let $B_1(y, p)$ denote the length of the path starting at s, ending at y, spanning value p, and ending in a type 1 segment that is found by our algorithm. Likewise, let $B_2(y, f, p)$ denote the length of a corresponding path ending in a type 2 segment found by our algorithm. Then the following recurrences determine $B_1(y, p)$ and $B_2(y, f, p)$:

$$\begin{aligned}
B_1(y,p) &= \min_{x,z,f,p':d_z<d_f<d_x<d_y,p'<p} \{B_2(z,f,p') + d(z,x) + A_1(x,y,p-p')\},\\
B_2(y,f,p) &= \min_{x,c,f,p':d_z<d_c<d_x<d_y,p'<p} \{B_1(z,p') + d(z,x) + A_2(x,y,c,f,p-p')\}.
\end{aligned}$$

Note that the segments corresponding to the optimum path are considered in this dynamic program, and so our solution will be at least as good as the one we get by using the segments corresponding to the ones on the optimum path (i.e., using the optimum type 1 segments and using the approximately optimum type 2 segments). We need to show only that this solution is good.

We focus on the segments corresponding to the optimum path P^*. Consider the segments S_i of length ℓ_i on the optimum path. If S_i is of type 1, our algorithm will find a (monotonic) segment with the same endpoints collecting the same amount of prize of no greater length. If S_i is of type 2, our algorithm (through its use of subroutine MCP) will find a path with the same endpoints collecting the same prize over length at most $\alpha_{CP}\ell_i$. Let L_1 denote the total length of the optimum type 1 segments, together with the lengths of the edges used to connect between segments. Let L_2 denote the total length of the optimum type 2 segments. Recall that $L_1 + L_2 = d_t + \epsilon(P^*)$ and that (by Corollary 3.2) $L_2 \leq \frac{3}{2}\epsilon(P^*)$. By concatenating the optimum type 1 segments and the approximately optimum type 2 segments, the dynamic program can (and therefore will) find a path collecting the same total prize as P^* of total length at most

$$\begin{aligned} L_1 + \alpha_{CP} L_2 &= L_1 + L_2 + (\alpha_{CP} - 1) L_2 \\ &\leq d_t + \epsilon(P^*) + (\alpha_{CP} - 1)\left(\frac{3}{2}\epsilon(P^*)\right) \\ &= d_t + \left(\frac{3}{2}\alpha_{CP} - \frac{1}{2}\right)\epsilon(P^*). \end{aligned}$$

In other words, we approximate the minimum excess to within a factor of $\frac{3}{2}\alpha_{CP} - \frac{1}{2}$.

3.2. An improved approximation for Min-Excess-Path. Our approximation guarantee for Min-Excess-Path derived above is based on treating the k-path subroutine as a "black box." In this section, we show how to slightly improve our approximation guarantee for the Min-Excess-Path problem by exploiting the details of the min-cost path algorithm derived from the work of Chaudhuri et al. [10].

Recall that Chaudhuri et al. provide an algorithm for constructing a tree containing two nodes s and t that spans at least k reward and has cost at most $(1 + \frac{1}{2}\delta)$ times the cost of the optimal k-path P^* from s to t. Doubling the edges of this tree, we obtain an approximation to the k-path problem with $\alpha_{CP} = (2 + \delta)$.

In fact, if the optimal path P^* has length $\ell = d(s,t) + \epsilon^*$, then the tree has length at most $(1 + \frac{1}{2}\delta)(d(s,t) + \epsilon^*)$. We convert this tree into a path from s to t by doubling all edges, except for the ones on the tree path from s to t. Noting that the total cost of "nondoubled" edges is at least $d(s,t)$, we get a path from s to t of length at most $(2+\delta)(d(s,t)+\epsilon^*) - d(s,t) = (1+\delta)(d(s,t)+\epsilon^*) + \epsilon^* = (1+\delta)\ell + \epsilon^*$. This stronger guarantee gives us an improved guarantee on the performance of the Min-Excess-Path algorithm described above.

In particular, suppose that we apply the k-path algorithm to a segment of type 2 with endpoints u and v, and having an optimum min-excess path with length $\ell = d(u,v) + \epsilon$. Then we get a path from u to v with the same value and length at most $(1+\delta)\ell + \epsilon$, for any fixed small constant δ.

In order to combine the lengths of all the type 2 segments, we rewrite this expression in terms of the excess of the path P^* at u and v. Let $\epsilon_u = d^{P^*}(u) - d_u$ denote the excess of P^* from s to u, and $\epsilon_v = d^{P^*}(v) - d_v$ the excess of P^* from s to v. Note that the ϵ in the above expression is not necessarily equal to $\epsilon_v - \epsilon_u$, but we can relate the two quantities via the triangle inequality—the distance along P^* from u to v can be written as $\ell = (d_v + \epsilon_v) - (d_u + \epsilon_u) \leq d(u,v) + \epsilon_v - \epsilon_u$; therefore, $\epsilon \leq \epsilon_v - \epsilon_u$. This implies that the Chaudhuri et al. algorithm returns a path of length at most $(1+\delta)\ell + \epsilon_v - \epsilon_u$ between u and v.

We can now apply Corollary 3.2 as before, to obtain an approximation in terms of the excess. The dynamic program finds a path collecting the same total value as P^* and of total length at most

$$\begin{aligned} L_1 + (1+\delta)L_2 + \sum_{\text{type 2 segments}} (\epsilon_v - \epsilon_u) &\leq L_1 + (1+\delta)L_2 + \epsilon(P^*) \\ &= d_t + 2\epsilon(P^*) + \delta L_2 \\ &\leq d_t + 2\epsilon(P^*) + \frac{3\delta}{2}\epsilon(P^*), \end{aligned}$$

where the last statement follows from Corollary 3.2. Therefore, we get an approximation ratio of $2 + \delta'$ for the Min-Excess-Path problem, for any small constant δ'.

4. Maximum discounted-prize path. In this section, we present an approximation algorithm for the Discounted-Reward TSP which builds upon our Min-Excess-Path algorithm. Recall that we aim to optimize $\rho(P) = \sum \gamma^{d_v^P} \pi_v$. Assume without loss of generality that the discount factor is $\gamma = 1/2$—we simply rescale each length ℓ to ℓ' such that $\gamma^\ell = (\frac{1}{2})^{\ell'}$, i.e., $\ell' = \ell \log_2(1/\gamma)$.

We first establish a property of an optimal solution that we make use of in our algorithm. Define the *scaled prize* π' of a node v to be the (discounted) reward that a path gets at node v if it follows a shortest path from the root to v. That is, $\pi'_v = \pi_v \gamma^{d_v}$. Let $\Pi'(P) = \sum_{v \in P} \pi'_v$. Note that for any path P, the discounted reward obtained by P is at most $\Pi'(P)$.

Now consider an optimal solution P^*. Fix a parameter ϵ that we will set later. Let t be the last node on the path P^* for which $d_t^{P^*} - d_t \leq \epsilon$; i.e., the excess of path P^* at t is at most ϵ. Consider the portion of P^* from root s to t. Call this path P_t^*.

Lemma 4.1. *Let P_t^* be the part of P^* from s to t. Then, $\rho(P_t^*) \geq \rho(P^*)(1 - \frac{1}{2^\epsilon})$.*

Proof. Assume otherwise. Suppose we shortcut P^* by taking a shortest path from s to the next node visited by P^* after t. This new path collects (discounted) rewards from the vertices of $P^* - P_t^*$, which form more than a $\frac{1}{2^\epsilon}$ fraction of the total discounted reward by assumption. The shortcutting procedure decreases the distance on each of these vertices by at least ϵ, meaning these rewards are "undiscounted" by a factor of at least 2^ϵ over what they would be in path P^*. Thus, the total reward on this path exceeds the optimum, a contradiction. ∎

It follows that we can approximate $\rho(P^*)$ by approximating $\rho(P_t^*)$. Based on the above observation, we give the algorithm of Figure 4.1 for finding an approximately optimal solution. Note that "guess t" and "guess k" are implemented by exhausting all polynomially many possibilities.

```
Algorithm for Discounted-Reward TSP
    1. Rescale all edge lengths so that $\gamma = 1/2$.
    2. Replace the prize value of each node with the prize discounted by the shortest
       path to that node: $\pi'_v = \gamma^{d_v} \pi_v$. Call this modified graph $G'$.
    3. Guess $t$—the last node on optimal path $P^*$ with excess less than $\epsilon$.
    4. Guess $k$—the value of $\Pi'(P_t^*)$.
    5. Apply our Min-Excess-Path approximation algorithm to find a path $P$
       collecting scaled prize $k$ with small excess.
    6. Return this path as the solution.
```

Fig. 4.1. *Approximation for maximum discounted-prize path (*Discounted-Reward *TSP).*

Our analysis below proceeds in terms of $\alpha = \alpha_{EP}$, the approximation factor for our Min-Excess-Path algorithm.

Lemma 4.2. *Our approximation algorithm finds a path P that collects discounted reward $\rho(P) \geq \Pi'(P_t^*)/2^{\alpha\epsilon}$.*

Proof. The prefix P_t^* of the optimum path shows that it is possible to collect scaled prize $k = \Pi'(P_t^*)$ on a path with excess ϵ. Thus, our approximation algorithm finds a path collecting the same scaled prize with excess at most $\alpha\epsilon$. In particular, the excess of any vertex v in P is at most $\alpha\epsilon$. Thus, the discounted reward collected at v is at least

$$\rho(v) \geq \pi_v \left(\frac{1}{2}\right)^{d_v+\alpha\epsilon} = \pi_v \left(\frac{1}{2}\right)^{d_v} \left(\frac{1}{2}\right)^{\alpha\epsilon} = \pi'_v \left(\frac{1}{2}\right)^{\alpha\epsilon}.$$

Summing over all $v \in P$ and observing $\Pi'(P) \geq \Pi'(P^*) \geq \Pi'(P^*_t)$ completes the proof. □

Combining Lemmas 4.2 and 4.1, we get the following.

THEOREM 4.3. *The solution returned by the above algorithm has* $\rho(P) \geq (1 - \frac{1}{2^\epsilon})\rho(P^*)/2^{\alpha\epsilon}$.

Proof.

$$\begin{aligned} \rho(P) &\geq \Pi'(P^*)/2^{\alpha\epsilon} && \text{by Lemma 4.2} \\ &\geq \rho(P^*_t)/2^{\alpha\epsilon} && \text{by definition of } \pi' \\ &\geq \left(1 - \tfrac{1}{2^\epsilon}\right)\rho(P^*)/2^{\alpha\epsilon} && \text{by Lemma 4.1.} \quad \square \end{aligned}$$

We can now set ϵ as we like. Writing $x = 2^{-\epsilon}$, we optimize our approximation factor by maximizing $(1-x)x^\alpha$ to deduce $x = \alpha/(\alpha+1)$. Plugging this into x yields an approximation ratio of $(1+\alpha_{EP})(1+1/\alpha_{EP})^{\alpha_{EP}}$.

5. Orienteering.

In this section, we present an algorithm for computing an approximately max-prize path of length at most D that starts at a specified vertex s. We will use the algorithm for MIN-EXCESS-PATH given in section 3 as a subroutine. Our algorithm for the max-prize problem is given in Figure 5.1. As before, "guess k" is implemented by performing exhaustive enumeration.

```
Algorithm for max-prize path (ORIENTEERING)
    1. Guess $k$, the amount of prize collected by an optimum ORIENTEERING solution. Let $\alpha = \lceil \alpha_{EP} \rceil + 1$.
    2. For each vertex $v$, compute the min-excess path from $s$ to $v$ collecting prize $k/\alpha$.
    3. There exists a $v$ such that the min-excess path returned has length at most $D$; return the corresponding path.
```

FIG. 5.1. *Algorithm for max-prize path (*ORIENTEERING*).*

We analyze this algorithm by showing that any optimum ORIENTEERING solution contains a low-excess path which, in turn, is an approximately optimum ORIENTEERING solution. More precisely, we prove that for some vertex v, there exists a path from s to v with excess at most $\frac{D-d_v}{\alpha_{EP}}$ that collects prize at least $\frac{\Pi^*}{\alpha_{PP}}$, where α_{EP} is the approximation ratio for MIN-EXCESS-PATH, α_{PP} is the desired approximation ratio for max-prize path, and Π^* is the prize of the optimum max-prize path. Assuming this path exists, our min-excess path computation on this vertex v will find a path with total length at most $d_v + \alpha_{EP}\frac{D-d_v}{\alpha_{EP}} = D$ and prize at least $\frac{\Pi^*}{\alpha_{PP}}$, providing an α_{PP}-approximation for ORIENTEERING.

Let t be the last vertex on the optimum ORIENTEERING path. We first consider the case where t is the vertex at maximum distance from s on the optimum path.

LEMMA 5.1. *If there is a path from s to t of length at most D that collects prize Π, such that t is the furthest point from s along this path, then there is a path from s to some node v with excess at most $\frac{D-d_v}{r}$ and prize at least $\frac{\Pi}{r}$ (for any integer $r \geq 1$).*

Proof. For each point a along the original path P, let $\epsilon(a) = d^P_a - d_a$; in other words, $\epsilon(a)$ is the excess in the length of the path to a over the shortest-path distance.

We have $\epsilon(t) \le D - d_t$. Consider mapping the points on the path to a line from 0 to $\epsilon(t)$ according to their excess (we observe that excess can only increase as we traverse path P). Divide this line into r intervals with length $\frac{\epsilon(t)}{r}$. Some such interval must contain at least $\frac{\Pi}{r}$ prize, since otherwise the entire interval from 0 to $\epsilon(t)$ would not be able to collect prize Π. Suppose such an interval starts with node a and ends with node v. We consider a path from s to v that takes the shortest s-a path and then follows path P from a to v. This path collects the prize of the interval from a to v in the original path, which is a prize of at least $\frac{\Pi}{r}$ as desired. The total length of this path is $d_a + d^P(a, v) = d_a + d_v^P - d_a^P = d_v + \epsilon(v) - \epsilon(a) \le d_v + \frac{\epsilon(t)}{r}$. The excess of this path is $\frac{\epsilon(t)}{r} \le \frac{D-d_t}{r} \le \frac{D-d_v}{r}$, where the last inequality follows by using that t is farther from s than v. □

Of course, in general the optimum ORIENTEERING path might have some intermediate node that is farther from s than the terminal node t. We will generalize the above lemma to account for this case.

LEMMA 5.2. *If there is a path from s to t of length at most D that collects prize Π, then there is a path from s to some node v with excess at most $\frac{D-d_v}{r}$ and prize at least $\frac{\Pi}{r+1}$ (for any integer $r \ge 1$).*

Proof. Let f be the furthest point from s along the given path P. We are interested in the case where $f \ne t$. We can break path P into two pieces; first a path from s to f and then a path from f to t. Using the symmetry of our metric, we can produce a second path from s to f by using the shortest path from s to t and then following the portion of our original path from f to t in reverse. We now have two paths from s to f, each of which has length at most D. The total length of these paths is bounded by $D + d_t$. We will call our paths A and B, and let their lengths be $d_f + \delta_A$ and $d_f + \delta_B$, respectively. Note that $\delta_A + \delta_B \le D + d_t - 2d_f < D - d_f$.

We now map path A to the interval from 0 to δ_A according to the excess at each point, much as in Lemma 5.1. We consider dividing this interval into pieces of length $\frac{\delta_A+\delta_B}{r}$ (the last subinterval may have shorter length if δ_A does not divide evenly). We perform the same process on path B. We have created a total of $r + 1$ intervals (this relies on the assumption that r is integral, allowing us to bound the sum of the ceilings of the number of intervals for each path). We conclude that some such interval has prize at least $\frac{\Pi}{r+1}$. We suppose without loss of generality that this interval spans a portion of path A from a to v. We now consider a path that travels from s to a via the shortest path and then from a to v following path A. The length of this path is bounded by $d_v + \frac{\delta_A+\delta_B}{r}$ for an excess of at most $\frac{D-d_f}{r} \le \frac{D-d_v}{r}$ as desired. □

Making use of Lemma 5.2, we can prove that our algorithm for ORIENTEERING obtains a constant approximation. Making use of Chaudhuri et al.'s approximation for min-cost s-t path [10] along with our result on MIN-EXCESS-PATH from section 3, we have a 4-approximation for ORIENTEERING.

THEOREM 5.3. *Our algorithm is an $(\lceil \alpha_{EP} \rceil + 1)$-approximation for the max-prize path (*ORIENTEERING*) problem, where α_{EP} is the approximation factor for* MIN-EXCESS-PATH.

Proof. Lemma 5.2 implies that there exists a path from s to some v with excess $\frac{D-d_v}{\lceil \alpha_{EP} \rceil}$ obtaining prize $\frac{\Pi^*}{\lceil \alpha_{EP} \rceil + 1}$. Such a path has length $d_v + \frac{D-d_v}{\lceil \alpha_{EP} \rceil}$, implying that the approximation algorithm for MIN-EXCESS-PATH will find a path from s to v with length at most $d_v + (D - d_v) = D$ and at least the same prize. The algorithm described will eventually try the proper values of k and v and find such a path in polynomial time. □

6. Extensions.

6.1. Max-prize tree and max-prize cycle.

In this section, we consider the tree and cycle variants of the ORIENTEERING problem. In max-prize tree, given a graph G with root r, prize function Π, and lengths d, we are required to output a tree $\mathcal{T}$ rooted at r with $d(\mathcal{T}) \leq \mathcal{D}$ and maximum possible reward $\Pi(\mathcal{T})$. This problem is also called the budget prize-collecting Steiner tree problem [20]. Although the unrooted version of the problem can be approximated to within a factor of $5 + \epsilon$ via a 3-approximation for k-MST [20], the version of the problem in which a tree is required to contain a specified vertex has remained open until recently.

Let the optimal solution for this problem be a tree $\mathcal{T}^*$. Double the edges of this tree to obtain an Euler tour of length at most $2D$. Now, divide this tour into two paths, each starting from the root r and having length at most D. Among them, let P' be the path that has greater reward. Now consider the max-prize path problem on the same graph with distance limit D. Clearly, the optimal solution P^* to this problem has $\Pi(P^*) \geq \Pi(P') \geq \frac{\Pi(\mathcal{T}^*)}{2}$. Thus, we can use the α_{PP}-approximation for ORIENTEERING to get a $2\alpha_{PP}$-approximation to T^*.

Finally, we note that we can use our algorithm for the ORIENTEERING problem to approximate max-prize cycle. Namely, we can find an approximately max-prize cycle of length at most D that contains a specified vertex s. To this end, we apply our algorithm to an instance of the ORIENTEERING problem with the starting node s and the length constraint $D/2$. To obtain a cycle from the resulting path, we connect its endpoints by a shortest path. Clearly, the length of the resulting cycle is at most D. Now, notice that an optimal max-prize cycle of length D can span at most twice the amount of prize as an optimal max-prize path of length $D/2$. Thus, using α_{PP}-approximation to ORIENTEERING, we get $2\alpha_{PP}$-approximation to the max-prize cycle problem.

6.2. Multiple-path ORIENTEERING.

In this section, we consider a variant of the ORIENTEERING problem in which we are allowed to construct up to k paths, each having length at most D.

We approximate this problem by applying the algorithm in section 4 successively k times, to construct the k paths. At the ith step, we set the prizes of all points visited in the first $i - 1$ paths to 0, and constructed the ith path on the new graph, using the ORIENTEERING algorithm in section 5. Using a set-cover-like argument, we get the following approximation guarantees for the cases when all paths have the same starting point and when different paths have different starts.[6]

THEOREM 6.1. *If all the paths have a common start node, the above algorithm gives a* $1/(1 - e^{-\alpha_{PP}})$*-approximation to multiple-path* ORIENTEERING. *If the paths have different start nodes, the above algorithm gives a* $(\alpha_{PP} + 1)$*-approximation to multiple-path* ORIENTEERING.

Proof. Consider first the case when all the paths have the same starting point. Let the difference in the reward collected by the optimal solution and the reward collected by our solution up to stage i be Π_i. At the beginning, this is the total reward of the optimal solution. At step i, at least one of the paths in the optimal solution collects reward, not collected by the algorithm by stage i, of value at least $\frac{1}{k}\Pi_i$. Then, using the approximation guarantee of the algorithm for ORIENTEERING, our solution collects at least a $\frac{1}{k\alpha_{PP}}$ fraction of this reward. That is, $\Pi_{i+1} \leq (1 - \frac{1}{k\alpha_{PP}})\Pi_i$. By the end of

[6] Subsequent to the initial publication of our work, Chekuri and Kumar [12] independently employed a similar analysis for other "maximum-coverage" problems.

k rounds, the total reward collected by optimal solution, but not collected by us, is at most $(1-\frac{1}{k\alpha_{PP}})^k\Pi(P^*) \le e^{-\alpha_{PP}}\Pi(P^*)$, and the result follows.

Next, consider the case when different paths have different starting locations. Let O_i be the set of points visited by the ith path in the optimal solution and A_i be the corresponding set of points visited by our algorithm. Let Δ_i be the set of points that are visited by the ith path in the optimal solution and some other path in our solution. Let $O = \cup_i O_i$, $A = \cup_i A_i$, and $\Delta = \cup_i \Delta_i$. Now, in the ith stage, there is a valid path starting at the ith source that visits all points in $O_i \setminus \Delta_i$. Thus we have $\Pi(A_i) \ge \frac{1}{\alpha_{PP}}(\Pi(O_i) - \Pi(\Delta_i))$. Summing over i, we get $\alpha_{PP}\Pi(A) \ge (\Pi(O) - \Pi(\Delta))$. But $\Pi(\Delta) \le \Pi(A)$. Thus $\Pi(A) \ge \frac{1}{\alpha_{PP}+1}\Pi(O)$. □

7. Hardness of approximation. All the problems discussed in this paper are NP-hard, as they are generalizations of the TSP. In this section, we show that the Min-Excess-Path problem and Orienteering are APX-hard; that is, it is NP-hard to approximate these problems to within an arbitrary constant factor.

The hardness of approximating the Min-Excess-Path problem follows from the APX-hardness of the TSP [24]. In particular, we can reduce the TSP to an instance of Min-Excess-Path on the same graph, with any one vertex as the start and the end point, and a reward quota of n. Then an α-approximation to Min-Excess-Path on this instance is also an α-approximation to the TSP. We therefore get the following theorem.

Theorem 7.1. *The* Min-Excess-Path *problem is NP-hard to approximate to within a factor of* $\frac{220}{219}$.

Theorem 7.2. Orienteering *is NP-hard to approximate to within a factor of* $\frac{1481}{1480}$.

Proof. We reduce the TSP on $\{1,2\}$-metrics to Orienteering. In particular, let $G = (V, E)$ be a complete graph on n nodes, with edges lengths in the set $\{1,2\}$. Engebretsen and Karpinski [14] show that the TSP is NP-hard to approximate within a factor of $1+\alpha = \frac{741}{740}$ on such graphs.

Our reduction is as follows. Let the length of the optimal TSP solution be $L = n + \delta n$. (We simply try all values of L between n and $2n$.) Suppose that there is an algorithm that approximates Orienteering within a factor of $1+\beta$, where $\beta \le \frac{\alpha}{2} = \frac{1}{1480}$. We apply this algorithm to the graph G with distance limit L. Note that the optimal solution (which is the optimal TSP path) collects $n-1$ nodes within distance L (all nodes, except the start, assuming a reward of 0 on the start node). Therefore, the solution returned by our algorithm collects $\frac{1}{1+\beta}(n-1)$ nodes. We augment this solution to a tour containing all the nodes by using $(1-\frac{1}{1+\beta})(n-1)+1$ edges of length at most 2. Therefore, the length of our solution is at most

$$\begin{aligned} L + 2(1-\tfrac{1}{1+\beta})(n-1)+2 &= L + \tfrac{2\beta}{1+\beta}(n-1)+2 \\ &< L + 2\beta n \\ &= L + \alpha n \le (1+\alpha)L, \end{aligned}$$

where the second inequality follows from assuming that $n > \frac{1}{\beta^2}$.

Therefore, we get a $(1+\alpha)$-approximation to the TSP on G, contradicting the fact that TSP is NP-hard to approximate to within a $(1+\alpha)$ factor on $\{1,2\}$-metrics. □

Using a similar argument as for Orienteering, we get a $\frac{1481}{1480}$ hardness of approximation for the max-prize cycle problem as well.

8. Conclusions. In this paper, we give constant-factor algorithms for the Orienteering problem, Discounted-Reward TSP, and some of their variants. We also prove that it is NP-hard to obtain a polynomial time approximation scheme (PTAS) for the Orienteering and Min-Excess-Path problems. An interesting open problem is to obtain better approximations, or even a PTAS, for these problems when the underlying metric is planar. Another interesting open problem is to consider the directed versions of the problems, although we believe that it may be hard to approximate these to within constant or even logarithmic factors. Some recent progress has been made in this direction by Chekuri and Pál [13], who developed quasi-polynomial time log-approximation algorithms for Orienteering and several related problems on directed graphs.

Even more ambitiously, returning to the MDP motivation for this work, one would like to generalize these results to probabilistic transition functions. However, this has the additional complication that the optimum solution may not even have a short description (it is no longer just a path). Still, perhaps some sort of nontrivial approximation bound, or a result holding in important special cases, can be found. The Ph.D. thesis of the second author [11] contains preliminary results in this direction.

REFERENCES

[1] E. M. Arkin, J. S. B. Mitchell, and G. Narasimhan, *Resource-constrained geometric network optimization*, in Proceedings of the 14th ACM Symposium on Computational Geometry, 1998, pp. 307–316.

[2] S. Arora and G. Karakostas, *A $2+\epsilon$ approximation algorithm for the k-MST problem*, in Proceedings of the 11th ACM Symposium on Discrete Algorithms, 2000, pp. 754–759.

[3] B. Awerbuch, Y. Azar, A. Blum, and S. Vempala, *New approximation guarantees for minimum-weight k-trees and prize-collecting salesmen*, SIAM J. Comput., 28 (1998), pp. 254–262.

[4] E. Balas, *The prize collecting traveling salesman problem*, Networks, 19 (1989), pp. 621–636.

[5] N. Bansal, A. Blum, S. Chawla, and A. Meyerson, *Approximation algorithms for deadline-TSP and vehicle routing with time-windows*, in Proceedings of the 35th ACM Symposium on Theory of Computing, 2004, pp. 166–174.

[6] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, Athena Scientific, Belmont, MA, 1995.

[7] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, Belmont, MA, 1996.

[8] D. Bienstock, M. X. Goemans, D. Simchi-Levi, and D. Williamson, *A note on the prize-collecting traveling salesman problem*, Math. Programming, 59 (1993), pp. 413–420.

[9] A. Blum, R. Ravi, and S. Vempala, *A constant-factor approximation algorithm for the k-MST problem*, J. Comput. System Sci., 58 (1999), pp. 101–108.

[10] K. Chaudhuri, B. Godfrey, S. Rao, and K. Talwar, *Paths, trees, and minimum latency tours*, in Proceedings of the 44th IEEE Symposium on Foundations of Computer Science, 2003, pp. 36–45.

[11] S. Chawla, *Graph Algorithms for Planning and Partitioning*, Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, 2005.

[12] C. Chekuri and A. Kumar, *Maximum coverage problem with group budget constraints and applications*, in Proceedings of the 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, Springer-Verlag, Berlin, 2004, pp. 72–83.

[13] C. Chekuri and M. P'al, *A recursive greedy algorithm for walks in directed graphs*, in Proceedings of the 46th IEEE Symposium on Foundations of Computer Science, 2005, pp. 245–253.

[14] L. Engebretsen and M. Karpinski, *Approximation hardness of TSP with bounded metrics*, in Proceedings of the 28th International Colloquium on Automata, Languages and Programming, Springer-Verlag, Berlin, 2001, pp. 201–212.

[15] N. Garg, *A 3-approximation for the minimum tree spanning k vertices*, in Proceedings of the 37th IEEE Symposium on Foundations of Computer Science, 1996, pp. 302–309.

[16] N. Garg, *Saving an epsilon: A 2-approximation for the k-MST problem in graphs*, in Proceedings of the 36th ACM Symposium on Theory of Computing, 2005, pp. 396–402.

[17] M. Goemans and D. Williamson, *A general approximation technique for constrained forest problems*, in Proceedings of the 3rd ACM Symposium on Discrete Algorithms, 1992, pp. 307–315.

[18] M. X. Goemans and D. P. Williamson, *A general approximation technique for constrained forest problems*, SIAM J. Comput., 24 (1995), pp. 296–317.

[19] B. L. Golden, L. Levy, and R. Vohra, *The orienteering problem*, Naval Res. Logist., 34 (1987), pp. 307–318.

[20] D. Johnson, M. Minkoff, and S. Phillips, *The prize collecting Steiner tree problem: Theory and practice*, in Proceedings of the 11th ACM Symposium on Discrete Algorithms, 2000, pp. 760–769.

[21] L. P. Kaelbling, M. L. Littman, and A. W. Moore, *Reinforcement learning: A survey*, J. Artificial Intelligence Res., 4 (1996), pp. 237–285.

[22] T. Lane and L. P. Kaelbling, *Approaches to macro decompositions of large Markov decision process planning problems*, in Proceedings of the 2001 SPIE Conference on Mobile Robotics, 2001, pp. 104–113.

[23] T. Lane and L. P. Kaelbling, *Nearly deterministic abstractions of Markov decision processes*, in Proceedings of the 18th AAAI National Conference on Artificial Intelligence, 2002, pp. 260–266.

[24] C. Papadimitriou and S. Vempala, *On the approximability of the traveling salesman problem*, in Proceedings of the 32nd ACM Symposium on Theory of Computing, 2000, pp. 126–133.

[25] M. L. Puterman, *Markov Decision Processes*, Wiley, New York, 1994.

[26] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, 1998.

SIAM J. Comput.
Vol. 37, No. 3, pp. 671–705

EFFECTIVE STRONG DIMENSION IN ALGORITHMIC INFORMATION AND COMPUTATIONAL COMPLEXITY*

KRISHNA B. ATHREYA[†], JOHN M. HITCHCOCK[‡], JACK H. LUTZ[§], AND ELVIRA MAYORDOMO[¶]

Abstract. The two most important notions of fractal dimension are *Hausdorff dimension*, developed by Hausdorff [*Math. Ann.*, 79 (1919), pp. 157–179], and *packing dimension*, developed independently by Tricot [*Math. Proc. Cambridge Philos. Soc.*, 91 (1982), pp. 57–74] and Sullivan [*Acta Math.*, 153 (1984), pp. 259–277]. Both dimensions have the mathematical advantage of being defined from measures, and both have yielded extensive applications in fractal geometry and dynamical systems. Lutz [*Proceedings of the* 15*th IEEE Conference on Computational Complexity*, Florence, Italy, 2000, IEEE Computer Society Press, Piscataway, NJ, 2000, pp. 158–169] has recently proven a simple characterization of Hausdorff dimension in terms of *gales*, which are betting strategies that generalize martingales. Imposing various computability and complexity constraints on these gales produces a spectrum of effective versions of Hausdorff dimension, including constructive, computable, polynomial-space, polynomial-time, and finite-state dimensions. Work by several investigators has already used these effective dimensions to shed significant new light on a variety of topics in theoretical computer science. In this paper we show that packing dimension can also be characterized in terms of gales. Moreover, even though the usual definition of packing dimension is considerably more complex than that of Hausdorff dimension, our gale characterization of packing dimension is an exact dual of—and every bit as simple as—the gale characterization of Hausdorff dimension. Effectivizing our gale characterization of packing dimension produces a variety of *effective strong dimensions*, which are exact duals of the effective dimensions mentioned above. In general (and in analogy with the classical fractal dimensions), the effective strong dimension of a set or sequence is at least as great as its effective dimension, with equality for sets or sequences that are sufficiently regular. We develop the basic properties of effective strong dimensions and prove a number of results relating them to fundamental aspects of randomness, Kolmogorov complexity, prediction, Boolean circuit-size complexity, polynomial-time degrees, and data compression. Aside from the above characterization of packing dimension, our two main theorems are the following. 1. If $\vec{\beta} = (\beta_0, \beta_1, \dots)$ is a computable sequence of biases that are bounded away from 0 and R is random with respect to $\vec{\beta}$, then the dimension and strong dimension of R are the lower and upper average entropies, respectively, of $\vec{\beta}$. 2. For each pair of Δ^0_2-computable real numbers $0 < \alpha \leq \beta \leq 1$, there exists $A \in \mathrm{E}$ such that the polynomial-time many-one degree of A has dimension α in E and strong dimension β in E. Our proofs of these theorems use a new large deviation theorem for self-information with respect to a bias sequence $\vec{\beta}$ that need not be convergent.

Key words. effective dimension, packing dimension, Hausdorff dimension, Kolmogorov complexity, computational complexity, Martin-Löf randomness

AMS subject classifications. 68Q15, 68Q30, 03D99, 28A78, 11K55

*Received by the editors January 23, 2003; accepted for publication (in revised form) January 4, 2007; published electronically June 21, 2007.

http://www.siam.org/journals/sicomp/37-3/44691.html

[†] Departments of Mathematics and Statistics, Iowa State University, Ames, IA 50011 (kba@iastate.edu). This author's research was supported in part by Air Force Office of Scientific Research grant ITSI F 49620-01-1-0076 and National Science Foundation grant 0344187.

[‡] Department of Computer Science, University of Wyoming, Laramie, WY 82072 (jhitchco@cs.uwyo.edu). This author's research was supported in part by National Science Foundation grants 9988483 and 0515313.

[§] Department of Computer Science, Iowa State University, Ames, IA 50011 (lutz@cs.iastate.edu). This author's research was supported in part by National Science Foundation grants 9988483 and 0344187, and by Spanish Government MEC project TIN 2005-08832-C03-02.

[¶] Departamento de Informática e Ingeniería de Sistemas, Universidad de Zaragoza, 50015 Zaragoza, Spain (elvira@unizar.es). This author's research was supported in part by Spanish Government MEC projects TIC2002-04019-C03-03 and TIN 2005-08832-C03-02, and by National Science Foundation grants 9988483 and 0344187. It was done while visiting Iowa State University.

DOI. 10.1137/S0097539703446912

1. Introduction.

Hausdorff dimension—a powerful tool of fractal geometry developed by Hausdorff [12] in 1919—was effectivized in 2000 by Lutz [22, 23, 24]. This has led to a spectrum of effective versions of Hausdorff dimension, including constructive, computable, polynomial-space, polynomial-time, and finite-state dimensions. Work by several investigators has already used these effective dimensions to illuminate a variety of topics in algorithmic information theory and computational complexity [23, 24, 1, 7, 27, 13, 16, 11, 14, 15, 10]. (See [28] for a survey of some of these results.) This work has also underscored and renewed the importance of earlier work by Ryabko [29, 30, 31, 32], Staiger [38, 39, 40], and Cai and Hartmanis [5] relating Kolmogorov complexity to classical Hausdorff dimension. (See section 6 of [24] for a discussion of this work.)

The key to all these effective dimensions is a simple characterization of classical Hausdorff dimension in terms of *gales*, which are betting strategies that generalize martingales. (Martingales, introduced by Lévy [18] and Ville [46], have been used extensively by Schnorr [33, 34, 35] and others in the investigation of randomness and by Lutz [20, 21] and others in the development of resource-bounded measure.) Given this characterization, it is a simple matter to impose computability and complexity constraints on the gales to produce the above-mentioned spectrum of effective dimensions.

In the 1980s, a new concept of fractal dimension, called the packing dimension, was introduced independently by Tricot [43] and Sullivan [41]. Packing dimension shares with Hausdorff dimension the mathematical advantage of being based on a measure. Over the past two decades, despite its greater complexity (requiring an extra optimization over all countable decompositions of a set in its definition), packing dimension has become, next to Hausdorff dimension, the most important notion of fractal dimensions, yielding extensive applications in fractal geometry and dynamical systems [9, 8].

The main result of this paper is a proof that packing dimension can also be characterized in terms of gales. Moreover, notwithstanding the greater complexity of packing dimension's definition (and the greater complexity of its behavior on compact sets, as established by Mattila and Mauldin [26]), our gale characterization of packing dimension is an exact dual of—and every bit as simple as—the gale characterization of Hausdorff dimension. (This duality and simplicity are in the *statement* of our gale characterization; its proof is perforce more involved than its counterpart for Hausdorff dimension.)

Effectivizing our gale characterization of packing dimension produces for each of the effective dimensions above an *effective strong dimension* that is its exact dual. Just as the Hausdorff dimension of a set is bounded above by its packing dimension, the effective dimension of a set is bounded above by its effective strong dimension. Moreover, just as in the classical case, the effective dimension coincides with the strong effective dimension for sets that are sufficiently regular.

After proving our gale characterization and developing the effective strong dimensions and some of their basic properties, we prove a number of results relating them to fundamental aspects of randomness, Kolmogorov complexity, prediction, Boolean circuit-size complexity, polynomial-time degrees, and data compression. Our two main theorems along these lines are the following.

1. If $\delta > 0$ and $\vec{\beta} = (\beta_0, \beta_1, \dots)$ is a computable sequence of biases with each

$\beta_i \in [\delta, \frac{1}{2}]$, then every sequence R that is random with respect to $\vec{\beta}$ has dimension

$$\dim(R) = \liminf_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \mathcal{H}(\beta_i)$$

and strong dimension

$$\mathrm{Dim}(R) = \limsup_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \mathcal{H}(\beta_i),$$

where $\mathcal{H}(\beta_i)$ is the Shannon entropy of β_i.

2. For every pair of Δ^0_2-computable real numbers $0 < \alpha \leq \beta \leq 1$ there is a decision problem $A \in \mathrm{E}$ such that the polynomial-time many-one degree of A has dimension α in E and strong dimension β in E.

In order to prove these theorems, we prove a new large deviation theorem for the self-information $\log \frac{1}{\mu^{\vec{\beta}}(w)}$, where $\vec{\beta}$ is as in 1 above. Note that $\vec{\beta}$ need not be convergent here.

A corollary of theorem 1 above is that, if the average entropies $\frac{1}{n}\sum_{i=0}^{n-1} \mathcal{H}(\beta_i)$ converge to a limit $\overline{H}(\vec{\beta})$ as $n \to \infty$, then $\dim(R) = \mathrm{Dim}(R) = \overline{H}(\vec{\beta})$. Since the convergence of these average entropies is a much weaker condition than the convergence of the biases β_n as $n \to \infty$, this corollary substantially strengthens Theorem 7.7 of [24].

Our remaining results are much easier to prove, but their breadth makes a strong prima facie case for the utility of effective strong dimension. They in some cases explain dual concepts that had been curiously neglected in earlier work, and they are likely to be useful in future applications. It is to be hoped that we are on the verge of seeing the full force of fractal geometry applied fruitfully to difficult problems in the theory of computing.

2. Preliminaries.

We use the set $\mathbb{Z}$ of integers, the set $\mathbb{Z}^+$ of (strictly) positive integers, the set $\mathbb{N}$ of natural numbers (i.e., nonnegative integers), the set $\mathbb{Q}$ of rational numbers, the set $\mathbb{R}$ of real numbers, and the set $[0, \infty)$ of nonnegative reals. All logarithms in this paper are base 2. We use the slow-growing function $\log^* n = \min\{j \in \mathbb{N} \mid t_j \geq n\}$, where $t_0 = 0$ and $t_{j+1} = 2^{t_j}$, and Shannon's binary entropy function $\mathcal{H} : [0,1] \to [0,1]$ defined by

$$\mathcal{H}(\beta) = \beta \log \frac{1}{\beta} + (1-\beta) \log \frac{1}{1-\beta},$$

where $0 \log \frac{1}{0} = 0$.

A *string* is a finite, binary string $w \in \{0,1\}^*$. We write $|w|$ for the length of a string w and λ for the empty string. For $i, j \in \{0, \ldots, |w|-1\}$, we write $w[i..j]$ for the string consisting of the ith through the jth bits of w, and $w[i]$ for $w[i..i]$, the ith bit of w. Note that the 0th bit $w[0]$ is the leftmost bit of w and that $w[i..j] = \lambda$ if $i > j$. A *sequence* is an infinite binary sequence. If S is a sequence and $i, j \in \mathbb{N}$, then the notations $S[i..j]$ and $S[i]$ are defined exactly as for strings. We work in the *Cantor space* $\mathbf{C}$ consisting of all sequences. A string $w \in \{0,1\}^*$ is a *prefix* of a sequence $S \in \mathbf{C}$, and we write $w \sqsubseteq S$, if $S[0..|w|-1] = w$. The *cylinder generated by* a string $w \in \{0,1\}^*$ is $\mathbf{C}_w = \{S \in \mathbf{C} | w \sqsubseteq S\}$. Note that $\mathbf{C}_\lambda = \mathbf{C}$.

A *language*, or *decision problem*, is a set $A \subseteq \{0,1\}^*$. We usually identify a language A with its characteristic sequence $\chi_A \in \mathbf{C}$ defined by $\chi_A[n] = \textbf{if } s_n \in A \textbf{ then } 1 \textbf{ else } 0$, where $s_0 = \lambda$, $s_1 = 0$, $s_2 = 1$, $s_3 = 00, \ldots$ is the standard enumeration of $\{0,1\}^*$. That is, we usually (but not always) use A to denote both the set $A \subseteq \{0,1\}^*$ and the sequence $A = \chi_A \in \mathbf{C}$.

Given a set $A \subseteq \{0,1\}^*$ and $n \in \mathbb{N}$, we use the abbreviations $A_{=n} = A \cap \{0,1\}^n$ and $A_{\leq n} = A \cap \{0,1\}^{\leq n}$. A *prefix set* is a set $A \subseteq \{0,1\}^*$ such that no element of A is a prefix of another element of A.

For each $i \in \mathbb{N}$ we define a class G_i of functions from $\mathbb{N}$ into $\mathbb{N}$ as follows.

$$\begin{aligned} G_0 &= \{f \mid (\exists k)(\forall^\infty n) f(n) \leq kn\}, \\ G_{i+1} &= 2^{G_i(\log n)} = \{f \mid (\exists g \in G_i)(\forall^\infty n) f(n) \leq 2^{g(\log n)}\}. \end{aligned}$$

We also define the functions $\hat{g}_i \in G_i$ by $\hat{g}_0(n) = 2n$, $\hat{g}_{i+1}(n) = 2^{\hat{g}_i(\log n)}$. We regard the functions in these classes as growth rates. In particular, G_0 contains the linearly bounded growth rates, and G_1 contains the polynomially bounded growth rates. It is easy to show that each G_i is closed under composition, that each $f \in G_i$ is $o(\hat{g}_{i+1})$, and that each $\hat{g}_i$ is $o(2^n)$. Thus G_i contains superpolynomial growth rates for all $i > 1$, but all growth rates in the G_i-hierarchy are subexponential.

Let CE be the class of computably enumerable languages. Within the class DEC of all decidable languages, we are interested in the exponential complexity classes $\mathrm{E}_i = \mathrm{DTIME}(2^{G_{i-1}})$ and $\mathrm{E}_i\mathrm{SPACE} = \mathrm{DSPACE}(2^{G_{i-1}})$ for $i \geq 1$. The much-studied classes $\mathrm{E} = \mathrm{E}_1 = \mathrm{DTIME}(2^{\mathrm{linear}})$, $\mathrm{E}_2 = \mathrm{DTIME}(2^{\mathrm{polynomial}})$, and $\mathrm{ESPACE} = \mathrm{E}_1\mathrm{SPACE} = \mathrm{DSPACE}(2^{\mathrm{linear}})$ are of particular interest.

We use the following classes of functions:

$$\begin{aligned} \mathrm{all} &= \{f \mid f : \{0,1\}^* \to \{0,1\}^*\}, \\ \mathrm{comp} &= \{f \in \mathrm{all} \mid f \text{ is computable}\}, \\ \mathrm{p}_i &= \{f \in \mathrm{all} \mid f \text{ is computable in } G_i \text{ time}\} \ (i \geq 1), \\ \mathrm{p}_i\mathrm{space} &= \{f \in \mathrm{all} \mid f \text{ is computable in } G_i \text{ space}\} \ (i \geq 1). \end{aligned}$$

(The length of the output *is* included as part of the space used in computing f.) We write p for p_1 and pspace for $\mathrm{p}_1$space.

A *constructor* is a function $\delta : \{0,1\}^* \to \{0,1\}^*$ that satisfies $x \sqsubsetneq \delta(x)$ for all x. The *result* of a constructor δ (i.e., the language *constructed* by δ) is the unique language $R(\delta)$ such that $\delta^n(\lambda) \sqsubseteq R(\delta)$ for all $n \in \mathbb{N}$. Intuitively, δ constructs $R(\delta)$ by starting with λ and then iteratively generating successively longer prefixes of $R(\delta)$. We write $R(\Delta)$ for the set of languages $R(\delta)$ such that δ is a constructor in Δ. The following facts are the reason for our interest in the above-defined classes of functions:

$R(\mathrm{all}) = \mathbf{C}$.
$R(\mathrm{comp}) = \mathrm{DEC}$.
For $i \geq 1$, $R(\mathrm{p}_i) = \mathrm{E}_i$.
For $i \geq 1$, $R(\mathrm{p}_i\mathrm{space}) = \mathrm{E}_i\mathrm{SPACE}$.

If D is a discrete domain (such as $\mathbb{N}, \{0,1\}^*, \mathbb{N} \times \{0,1\}^*$, etc.), then a function $f : D \longrightarrow [0,\infty)$ is *Δ-computable* if there is a function $\hat{f} : \mathbb{N} \times D \longrightarrow \mathbb{Q} \cap [0,\infty)$ such that $|\hat{f}(r,x) - f(x)| \leq 2^{-r}$ for all $r \in \mathbb{N}$ and $x \in D$ and $\hat{f} \in \Delta$ (with r coded in unary and the output coded in binary). We say that f is *exactly Δ-computable* if $f : D \longrightarrow \mathbb{Q} \cap [0,\infty)$ and $f \in \Delta$. We say that f is *lower semicomputable* if there is a computable function $\hat{f} : D \times \mathbb{N} \to \mathbb{Q}$ such that

(a) for all $(x,t) \in D \times \mathbb{N}$, $\hat{f}(x,t) \leq \hat{f}(x,t+1) < f(x)$, and

(b) for all $x \in D$, $\lim_{t\to\infty} \hat{f}(x,t) = f(x)$.

Finally, we say that f is Δ^0_2-computable if f is computable (i.e., comp-computable) relative to the halting oracle.

A real number $\alpha \in [0,\infty)$ is *computable* (respectively, Δ^0_2-*computable*) if the function $f : \{0\} \to [0,\infty)$ defined by $f(0) = \alpha$ is computable (respectively, Δ^0_2-computable).

Let k be a positive integer. A *k-account finite-state gambler (k-account FSG)* is a tuple $G = (Q, \delta, \beta, q_0, \vec{c_0})$, where

- Q is a nonempty, finite set of states,
- $\delta : Q \times \{0,1\} \to Q$ is the transition function,
- $\beta : \{1,\ldots,k\} \times Q \times \{0,1\} \to \mathbb{Q} \cap [0,1]$ is the betting function,
- $q_0 \in Q$ is the initial state, and
- $\vec{c_0}$ is the initial capital vector, a sequence of k nonnegative rational numbers.

The betting function satisfies $\beta(i,q,0) + \beta(i,q,1) = 1$ for each $q \in Q$ and $1 \leq i \leq k$. We use the standard extension $\delta^* : \Sigma^* \to Q$ of δ defined recursively by $\delta^*(\lambda) = q_0$ and $\delta^*(wb) = \delta(\delta^*(w), b)$ for all $w \in \{0,1\}^*$ and $b \in \{0,1\}$.

3. Fractal dimensions. In this section we briefly review the classical definitions of some fractal dimensions and the relationships among them. Since we are primarily interested in binary sequences and (equivalently) decision problems, we focus on fractal dimension in the Cantor space $\mathbf{C}$.

For each $k \in \mathbb{N}$, we let $\mathcal{A}_k$ be the collection of all prefix sets A such that $A_{<k} = \emptyset$. For each $X \subseteq \mathbf{C}$, we then define the families

$$\mathcal{A}_k(X) = \left\{ A \in \mathcal{A}_k \,\middle|\, X \subseteq \bigcup_{w\in A} \mathbf{C}_w \right\},$$
$$\mathcal{B}_k(X) = \{ A \in \mathcal{A}_k \,|(\forall w \in A)\mathbf{C}_w \cap X \neq \emptyset \}.$$

If $A \in \mathcal{A}_k(X)$, then we say that the prefix set A *covers* the set X. If $A \in \mathcal{B}_k(X)$, then we call the prefix set A a *packing* of X. For $X \subseteq \mathbf{C}$, $s \in [0,\infty)$, and $k \in \mathbb{N}$, we then define

$$H^s_k(X) = \inf_{A\in\mathcal{A}_k(X)} \sum_{w\in A} 2^{-s|w|},$$
$$P^s_k(X) = \sup_{A\in\mathcal{B}_k(X)} \sum_{w\in A} 2^{-s|w|}.$$

Since $H^s_k(X)$ and $P^s_k(X)$ are monotone in k, the limits

$$H^s(X) = \lim_{k\to\infty} H^s_k(X),$$
$$P^s_\infty(X) = \lim_{k\to\infty} P^s_k(X)$$

exist, though they may be infinite. We then define

$$P^s(X) = \inf \left\{ \sum_{i=0}^{\infty} P^s_\infty(X_i) \,\middle|\, X \subseteq \bigcup_{i=0}^{\infty} X_i \right\}. \tag{3.1}$$

The set functions H^s and P^s have the technical properties of an outer measure [9], and the (possibly infinite) quantities $H^s(X)$ and $P^s(X)$ are thus known as the *s-dimensional Hausdorff (outer) cylinder measure* of X and the *s-dimensional packing* *(outer) cylinder measure* of X, respectively. The set function P^s_∞ is *not* an outer measure; this is the reason for the extra optimization (3.1) in the definition of the packing measure.

Definition. *Let* $X \subseteq \mathbf{C}$.

1. *The* Hausdorff dimension *of* X *is* $\dim_{\mathrm{H}}(X) = \inf\{s \in [0,\infty) | H^s(X) = 0\}$.
2. *The* packing dimension *of* X *is* $\dim_{\mathrm{P}}(X) = \inf\{s \in [0,\infty) | P^s(X) = 0\}$.

The proof of our main result uses a well-known characterization of packing dimension as a modified box dimension. For each $X \subseteq \mathbf{C}$ and $n \in \mathbb{N}$, let

$$N_n(X) = \Big|\{w \in \{0,1\}^n | (\exists S \in X) w \sqsubseteq S\}\Big|.$$

Then the *upper box dimension* of X is

$$\overline{\dim}_{\mathrm{B}}(X) = \limsup_{n\to\infty} \frac{\log N_n(X)}{n}. \tag{3.2}$$

The lower box dimension $\underline{\dim}_{\mathrm{B}}(X)$, which we do not use here, is obtained by using a limit inferior in place of the limit superior in (3.2). When $\underline{\dim}_{\mathrm{B}}(X) = \overline{\dim}_{\mathrm{B}}(X)$, this quantity, written $\dim_{\mathrm{B}}(X)$, is called the box dimension of X.

Box dimensions are over 60 years old, have been reinvented many times, and have been named many things, including Minkowski dimension, Kolmogorov entropy, Kolmogorov dimension, topological entropy, metric dimension, logarithmic density, and information dimension. Box dimensions are often used in practical applications of fractal geometry because they are easy to estimate, but they are not well-behaved mathematically. The *modified upper box dimension*

$$\overline{\dim}_{\mathrm{MB}}(X) = \inf\left\{\sup_i \overline{\dim}_{\mathrm{B}}(X_i) \,\middle|\, X \subseteq \bigcup_{i=0}^{\infty} X_i\right\} \tag{3.3}$$

is much better behaved. (Note that (3.3), like (3.1), is an optimization over all countable decompositions of X.) In fact, the following relations are well known [9].

Theorem 3.1. *For all* $X \subseteq \mathbf{C}$, $0 \le \dim_{\mathrm{H}}(X) \le \overline{\dim}_{\mathrm{MB}}(X) = \dim_{\mathrm{P}}(X) \le \overline{\dim}_{\mathrm{B}}(X) \le 1$.

The above dimensions are *monotone*, i.e., $X \subseteq Y$ implies $\dim(X) \le \dim(Y)$, and *stable*, i.e., $\dim(X \cup Y) = \max\{\dim(X), \dim(Y)\}$. The Hausdorff and packing dimensions are also *countably stable*, i.e., $\dim(\cup_{i=0}^{\infty} X_i) = \sup\{\dim(X_i) | i \in \mathbb{N}\}$.

4. Gale characterizations.

In this section we review the gale characterization of Hausdorff dimension and prove our main theorem, which is the dual gale characterization of packing dimension.

Definition. *Let* $s \in [0,\infty)$.

1. *An* s-supergale *is a function* $d : \{0,1\}^* \longrightarrow [0,\infty)$ *that satisfies the condition*

$$d(w) \ge 2^{-s}[d(w0) + d(w1)] \tag{4.1}$$

for all $w \in \{0,1\}^*$.

2. *An* s-gale *is an* s*-supergale that satisfies* (4.1) *with equality for all* $w \in \{0,1\}^*$.
3. *A* supermartingale *is a* 1*-supergale.*
4. *A* martingale *is a* 1*-gale.*

Intuitively, we regard a supergale d as a strategy for betting on the successive bits of a sequence $S \in \mathbf{C}$. More specifically $d(w)$ is the amount of capital that d has after betting on the prefix w of S. If $s = 1$, then the right-hand side of (4.1) is the conditional expectation of $d(wb)$, given that w has occurred (when b is a uniformly distributed binary random variable). Thus a martingale models a gambler's capital when the payoffs are fair. (The expected capital after the bet is the actual capital before the bet.) In the case of an s-gale, if $s < 1$, the payoffs are less than fair; if $s > 1$, the payoffs are more than fair.

We use the following known generalization of the Kraft inequality.

Lemma 4.1 (Lutz [23]). *Let $s \in [0, \infty)$. If d is an s-supergale and $B \subseteq \{0, 1\}^*$ is a prefix set, then for all $w \in \{0, 1\}^*$, $\sum_{u \in B} 2^{-s|u|} d(wu) \leq d(w)$.*

We now define two criteria for the success of a gale or supergale.

Definition. *Let d be an s-supergale, where $s \in [0, \infty)$.*

1. *We say that d* succeeds *on a sequence $S \in \mathbf{C}$ if*

$$\limsup_{n \to \infty} d(S[0..n-1]) = \infty. \tag{4.2}$$

The success set *of d is $S^\infty[d] = \{S \in \mathbf{C} | d \text{ succeeds on } S\}$.*

2. *We say that d* succeeds strongly *on a sequence $S \in \mathbf{C}$ if*

$$\liminf_{n \to \infty} d(S[0..n-1]) = \infty. \tag{4.3}$$

The strong success set *of d is $S^\infty_{\mathrm{str}}[d] = \{S \in \mathbf{C} | d \text{ succeeds strongly on } S\}$.*

We have written conditions (4.2) and (4.3) in a fashion that emphasizes their duality. Condition (4.2) says simply that the set of values $d(S[0..n-1])$ is unbounded, while condition (4.3) says that $d(S[0..n-1]) \to \infty$ as $n \to \infty$.

Notation. Let $X \subseteq \mathbf{C}$.

1. $\mathcal{G}(X)$ is the set of all $s \in [0, \infty)$ for which there exists an s-gale d such that $X \subseteq S^\infty[d]$.
2. $\mathcal{G}^{\mathrm{str}}(X)$ is the set of all $s \in [0, \infty)$ for which there exists an s-gale d such that $X \subseteq S^\infty_{\mathrm{str}}[d]$.
3. $\widehat{\mathcal{G}}(X)$ is the set of all $s \in [0, \infty)$ for which there exists an s-supergale d such that $X \subseteq S^\infty[d]$.
4. $\widehat{\mathcal{G}}^{\mathrm{str}}(X)$ is the set of all $s \in [0, \infty)$ for which there exists an s-supergale d such that $X \subseteq S^\infty_{\mathrm{str}}[d]$.

Note that $s' \geq s \in \mathcal{G}(X)$ implies that $s' \in \mathcal{G}(X)$, and similarly for the classes $\mathcal{G}^{\mathrm{str}}(X)$, $\widehat{\mathcal{G}}(X)$, and $\widehat{\mathcal{G}}^{\mathrm{str}}(X)$. The following fact is also clear.

Observation 4.2. *For all $X \subseteq \mathbf{C}$, $\mathcal{G}(X) = \widehat{\mathcal{G}}(X)$ and $\mathcal{G}^{\mathrm{str}}(X) = \widehat{\mathcal{G}}^{\mathrm{str}}(X)$.*

For Hausdorff dimension, we have the following known fact.

Theorem 4.3 (gale characterization of Hausdorff dimension; see Lutz [23]). *For all $X \subseteq \mathbf{C}$, $\dim_{\mathrm{H}}(X) = \inf \mathcal{G}(X)$.*

Our main result is the following dual of Theorem 4.3.

Theorem 4.4 (gale characterization of packing dimension). *For all $X \subseteq \mathbf{C}$, $\dim_{\mathrm{P}}(X) = \inf \mathcal{G}^{\mathrm{str}}(X)$.*

By Observation 4.2, we could equivalently use $\widehat{\mathcal{G}}(X)$ and $\widehat{\mathcal{G}}^{\mathrm{str}}(X)$ in Theorems 4.3 and 4.4, respectively. We will use the following lemma to prove Theorem 4.4.

Lemma 4.5. *For each family of sets $\{X_k \subseteq \mathbf{C} \,|\, k \in \mathbb{N}\}$, $\inf \mathcal{G}^{\mathrm{str}}\left(\bigcup_k X_k\right) = \sup_k \inf \mathcal{G}^{\mathrm{str}}(X_k)$.*

Proof. The inequality $\inf \mathcal{G}^{\mathrm{str}}(\bigcup_k X_k) \geq \sup_k \inf \mathcal{G}^{\mathrm{str}}(X_k)$ holds trivially.

To prove that $\inf \mathcal{G}^{\rm str}(\bigcup_k X_k) \le \sup_k \inf \mathcal{G}^{\rm str}(X_k)$, let $s > \sup_k \inf \mathcal{G}^{\rm str}(X_k)$. Then for each $k \in \mathbb{N}$ there is an s-gale d_k such that $X_k \subseteq S^{\infty}_{\rm str}[d_k]$. We define an s-gale d by

$$d(w) = \sum_{k\in\mathbb{N}} \frac{2^{-k}}{d_k(\lambda)} \cdot d_k(w)$$

for all $w \in \{0,1\}^*$. Then for each k, for any $S \in X_k$, we have

$$d(S[0..n-1]) \ge \frac{2^{-k}}{d_k(\lambda)} \cdot d_k(S[0..n-1])$$

for all n, so $S \in S^{\infty}_{\rm str}[d]$. Therefore $\bigcup_k X_k \subseteq S^{\infty}_{\rm str}[d]$, and the lemma follows. □

Proof of Theorem 4.4. Let $X \subseteq \mathbf{C}$. By Theorem 3.1, it suffices to show that $\overline{\dim}_{\rm MB}(X) = \inf \mathcal{G}^{\rm str}(X)$.

To see that $\overline{\dim}_{\rm MB}(X) \le \inf \mathcal{G}^{\rm str}(X)$, let $s > \inf \mathcal{G}^{\rm str}(X)$. It suffices to show that $\overline{\dim}_{\rm MB}(X) \le s$.

By our choice of s, there is an s-gale d such that $X \subseteq S^{\infty}_{\rm str}[d]$. For each $n \in \mathbb{N}$, let

$$B_n = \{w \in \{0,1\}^n | d(w) > d(\lambda)\}$$

and

$$Y_n = \{S \in \mathbf{C} | S[0..n-1] \in B_n\}.$$

For each $i \in \mathbb{N}$, let

$$X_i = \bigcap_{n=i}^{\infty} Y_n,$$

and note that

$$X \subseteq \bigcup_{i=0}^{\infty} X_i. \tag{4.4}$$

For all $n \ge i \in \mathbb{N}$, we have $X_i \subseteq Y_n$, whence the generalized Kraft inequality (Lemma 4.1) tells us that

$$N_n(X_i) \le N_n(Y_n) = |B_n| < 2^{sn}.$$

It follows that, for all $i \in \mathbb{N}$,

$$\overline{\dim}_{\rm B}(X_i) = \limsup_{n\to\infty} \frac{\log N_n(X_i)}{n} \le s,$$

whence by (4.4),

$$\overline{\dim}_{\rm MB}(X) \le \sup_{i\in\mathbb{N}} \overline{\dim}_{\rm B}(X_i) \le s.$$

To see that $\inf \mathcal{G}^{\rm str}(X) \le \overline{\dim}_{\rm MB}(X)$, let $s > s' > s'' > \overline{\dim}_{\rm MB}(X)$. It suffices to show that $\inf \mathcal{G}^{\rm str}(X) \le s$. Since $s'' > \overline{\dim}_{\rm MB}(X)$, there exist sets $X_0, X_1, \ldots \subseteq \mathbf{C}$ such that $X = \bigcup_{i=0}^{\infty} X_i$ and $\overline{\dim}_{\rm B}(X_i) < s''$ for all $i \in \mathbb{N}$. By Lemma 4.5, it suffices to show that $s \in \mathcal{G}^{\rm str}(X_i)$ for all $i \in \mathbb{N}$.

Fix $i \in \mathbb{N}$. Since $\overline{\dim}_{\mathrm{B}}(X_i) < s''$, there exists $n_0 \in \mathbb{N}$ such that, for all $n \geq n_0$, $\frac{\log N_n(X_i)}{n} < s''$, i.e., $N_n(X_i) < 2^{s''n}$. For each $n \geq n_0$, let

$$A_n = \{S[0..n-1] | S \in X_i\}$$

(noting that $|A_n| = N_n(X_i)$), and define $d_n : \{0,1\}^* \to [0,\infty)$ by

$$d_n(w) = \begin{cases} 2^{(s-s')|w|} \displaystyle\sum_{\substack{u \\ wu \in A_n}} 2^{-s'|u|} & \text{if } |w| \leq n, \\ 2^{(s-1)(|w|-n)} d_n(w[0..n-1]) & \text{if } |w| > n. \end{cases}$$

It is routine to verify that d_n is an s-gale for each $n \geq n_0$. Note also that $d_n(w) = 2^{(s-s')n}$ for all $n \geq n_0$ and $w \in A_n$. Let $d = \sum_{n=n_0}^{\infty} d_n$. Then

$$\begin{aligned} d(\lambda) = \sum_{n=n_0}^{\infty} d_n(\lambda) &= \sum_{n=n_0}^{\infty} |A_n| 2^{-s'n} = \sum_{n=n_0}^{\infty} N_n(X_i) 2^{-s'n} \\ &< \sum_{n=n_0}^{\infty} 2^{(s''-s')n} < \infty, \end{aligned}$$

so d is an s-gale by linearity. Let $S \in X_i$. Then, for all $n \geq n_0$, $S[0..n-1] \in A_n$, so

$$d(S[0..n-1]) \geq d_n(S[0..n-1]) \geq 2^{(s-s')n}.$$

Thus $S \in S_{\mathrm{str}}^{\infty}[d]$. This shows that $X_i \subseteq S_{\mathrm{str}}^{\infty}[d]$, whence $s \in \mathcal{G}^{\mathrm{str}}(X_i)$. □

5. Effective strong dimensions. Theorem 4.3 has been used to effectivize Hausdorff dimension at a variety of levels. In this section we review these effective dimensions while using Theorem 4.4 to develop the dual effective strong dimensions.

We define a gale or supergale to be *constructive* if it is lower semicomputable. For any $s \in [0,\infty)$ and any k-account FSG G an s-gale $d_G^{(s)}$ is defined as follows [11]. (Recall that finite-state gamblers were defined in section 2.) For each $1 \leq i \leq k$ we define an s-gale $d_{G,i}^{(s)}$ by the recursion

$$d_{G,i}^{(s)}(\lambda) = c_{0,i},$$

$$d_{G,i}^{(s)}(wb) = 2^s d_{G,i}^{(s)}(w) \beta(i, \delta^*(w), b)$$

for all $w \in \{0,1\}^*$ and $b \in \{0,1\}$. Then

$$d_G^{(s)} = \sum_{i=1}^{k} d_{G,i}^{(s)}.$$

We define an s-gale d to be *finite-state* if there is an FSG G such that $d_G^{(s)} = d$. For the rest of this paper, Δ denotes one of the classes all, comp, p, pspace, p_2, $\mathrm{p}_2$space, etc., defined in section 2.

For each $\Gamma \in \{\mathrm{constr}, \Delta, \mathrm{FS}\}$ and $X \subseteq \mathbf{C}$, we define the sets $\mathcal{G}_\Gamma(X)$, $\mathcal{G}_\Gamma^{\mathrm{str}}(X)$, $\widehat{\mathcal{G}}_\Gamma(X)$, and $\widehat{\mathcal{G}}_\Gamma^{\mathrm{str}}(X)$ just as the classes $\mathcal{G}(X)$, $\mathcal{G}^{\mathrm{str}}(X)$, $\widehat{\mathcal{G}}(X)$, and $\widehat{\mathcal{G}}^{\mathrm{str}}(X)$ were defined in section 4, but with the following modifications:

(i) If $\Gamma = \mathrm{constr}$, then d is required to be constructive.

(ii) If $\Gamma = \Delta$, then d is required to be Δ-computable.

(iii) In $\mathcal{G}_{\mathrm{FS}}(X)$ and $\mathcal{G}^{\mathrm{str}}_{\mathrm{FS}}(X)$, d is required to be finite-state.

(iv) $\widehat{\mathcal{G}}_{\mathrm{FS}}(X)$ and $\widehat{\mathcal{G}}^{\mathrm{str}}_{\mathrm{FS}}(X)$ are not defined.

The following effectivizations of Hausdorff and packing dimension are motivated by Theorems 4.3 and 4.4.

Definition. *Let* $X \subseteq \mathbf{C}$ *and* $S \in \mathbf{C}$*.*

1. [24] *The* constructive dimension *of* X *is* $\mathrm{cdim}(X) = \inf \mathcal{G}_{\mathrm{constr}}(X)$*.*
2. *The* constructive strong dimension *of* X *is* $\mathrm{cDim}(X) = \inf \mathcal{G}^{\mathrm{str}}_{\mathrm{constr}}(X)$*.*
3. [24] *The* dimension *of* S *is* $\dim(S) = \mathrm{cdim}(\{S\})$*.*
4. *The* strong dimension *of* S *is* $\mathrm{Dim}(S) = \mathrm{cDim}(\{S\})$*.*
5. [23] *The* Δ-dimension *of* X *is* $\dim_{\Delta}(X) = \inf \mathcal{G}_{\Delta}(X)$*.*
6. *The* Δ-strong dimension *of* X *is* $\mathrm{Dim}_{\Delta}(X) = \inf \mathcal{G}^{\mathrm{str}}_{\Delta}(X)$*.*
7. [23] *The* dimension *of* X in $R(\Delta)$ *is* $\dim(X|R(\Delta)) = \dim_{\Delta}(X \cap R(\Delta))$*.*
8. *The* strong dimension *of* X in $R(\Delta)$ *is* $\mathrm{Dim}(X|R(\Delta)) = \mathrm{Dim}_{\Delta}(X \cap R(\Delta))$*.*
9. [11] *The* finite-state dimension *of* X *is* $\dim_{\mathrm{FS}}(X) = \inf \mathcal{G}_{\mathrm{FS}}(X)$*.*
10. *The* finite-state strong dimension *of* X *is* $\mathrm{Dim}_{\mathrm{FS}}(X) = \inf \mathcal{G}^{\mathrm{str}}_{\mathrm{FS}}(X)$*.*
11. [11] *The* finite-state dimension *of* S *is* $\dim_{\mathrm{FS}}(S) = \dim_{\mathrm{FS}}(\{S\})$*.*
12. *The* finite-state strong dimension *of* S *is* $\mathrm{Dim}_{\mathrm{FS}}(S) = \mathrm{Dim}_{\mathrm{FS}}(\{S\})$*.*

In parts 1, 2, 5, and 6 of the above definition, we could equivalently use the "hatted" sets $\widehat{\mathcal{G}}_{\mathrm{constr}}(X)$, $\widehat{\mathcal{G}}^{\mathrm{str}}_{\mathrm{constr}}(X)$, $\widehat{\mathcal{G}}_{\Delta}(X)$, and $\widehat{\mathcal{G}}^{\mathrm{str}}_{\Delta}(X)$ in place of their unhatted counterparts. In the case of parts 5 and 6, this follows from Lemma 4.7 of [23]. In the case of parts 1 and 2, it follows from the main theorem in [15] (which answered an open question in [24], where $\widehat{\mathcal{G}}_{\mathrm{constr}}(X)$ was in fact used in defining $\mathrm{cdim}(X)$).

The polynomial-time dimensions $\dim_{\mathrm{p}}(X)$ and $\mathrm{Dim}_{\mathrm{p}}(X)$ are also called the feasible dimension and the feasible strong dimension, respectively. The notation $\dim_{\mathrm{p}}(X)$ for the p-dimension is all too similar to the notation $\dim_{\mathrm{P}}(X)$ for the classical packing dimension, but confusion is unlikely because these dimensions typically arise in quite different contexts.

Note that the classical Hausdorff and packing dimensions can each now be written in three different ways, i.e.,

$$\dim_{\mathrm{H}}(X) = \dim_{\mathrm{all}}(X) = \dim(X|\mathbf{C})$$

and

$$\dim_{\mathrm{P}}(X) = \mathrm{Dim}_{\mathrm{all}}(X) = \mathrm{Dim}(X|\mathbf{C}).$$

Observations 5.1.

1. *Each of the dimensions that we have defined is* monotone *(e.g.,* $X \subseteq Y$ *implies* $\mathrm{cdim}(X) \leq \mathrm{cdim}(Y)$*).*
2. *Each of the effective strong dimensions is bounded below by the corresponding effective dimension (e.g.,* $\mathrm{cdim}(X) \leq \mathrm{cDim}(X)$*).*
3. *Each of the dimensions that we have defined is nonincreasing as the effectivity constraint is relaxed (e.g.,* $\dim_{\mathrm{H}}(X) \leq \mathrm{cdim}(X) \leq \dim_{\mathrm{pspace}}(X) \leq \dim_{\mathrm{FS}}(X)$*).*
4. *Each of the dimensions that we have defined is nonnegative and assigns* $\mathbf{C}$ *the dimension* 1*.*

Lemma 5.2. *The finite-state dimensions are* stable*; i.e., for all* $X, Y \subseteq \mathbf{C}$*,*

$$\dim_{\mathrm{FS}}(X \cup Y) = \max\{\dim_{\mathrm{FS}}(X), \dim_{\mathrm{FS}}(Y)\}$$

and

$$\mathrm{Dim}_{\mathrm{FS}}(X \cup Y) = \max\{\mathrm{Dim}_{\mathrm{FS}}(X), \mathrm{Dim}_{\mathrm{FS}}(Y)\}.$$

Proof. The stability of finite-state dimension was proved in [11]. The same arguments establish stability for finite-state strong dimension. □

DEFINITION. *Let $X, X_0, X_1, X_2, \ldots \subseteq \mathbf{C}$.*

1. *We say that X is a* Δ-union of the Δ-dimensioned sets $\{X_k | k \in \mathbb{N}\}$ *if $X = \bigcup_{k=0}^{\infty} X_k$ and for each $s > \sup_{k\in\mathbb{N}} \dim_\Delta(X_k)$ with 2^s rational there is a function $d : \mathbb{N} \times \{0,1\}^* \to [0,\infty)$ with the following three properties:*
   - (i) *d is Δ-computable.*
   - (ii) *For each $k \in \mathbb{N}$, if we write $d_k(w) = d(k, w)$, then the function d_k is an s-gale.*
   - (iii) *For each $k \in \mathbb{N}$, $X_k \subseteq S^\infty[d_k]$.*

   Analogously, X is a Δ-union of the Δ-strong dimensioned sets $\{X_k | k \in \mathbb{N}\}$ *if there is a d with the above properties that also satisfies*
   - (iv) *For each $k \in \mathbb{N}$, $X_k \subseteq S^\infty_{\mathrm{str}}[d_k]$.*
2. *We say that X is a* Δ-union of the sets $\{X_k | k \in \mathbb{N}\}$ dimensioned in $R(\Delta)$ *if $X = \bigcup_{k=0}^{\infty} X_k$ and $X \cap R(\Delta)$ is a Δ-union of the Δ-dimensioned sets $\{X_k \cap R(\Delta) | k \in \mathbb{N}\}$.*

   Analogously, X is a Δ-union of the sets $\{X_k | k \in \mathbb{N}\}$ strong dimensioned in $R(\Delta)$ *if $X = \bigcup_{k=0}^{\infty} X_k$ and $X \cap R(\Delta)$ is an Δ-union of the Δ-strong dimensioned sets $\{X_k \cap R(\Delta) | k \in \mathbb{N}\}$.*

LEMMA 5.3. *The dimensions defined from Δ are* Δ-countably stable*; i.e., if X is a Δ-union of the Δ-dimensioned sets $X_0, X_1, X_2, \ldots,$ then*

$$\dim_\Delta(X) = \sup_{k\in\mathbb{N}} \dim_\Delta(X_k),$$

and if X is a Δ-union of the Δ-strong dimensioned sets $X_0, X_1, X_2, \ldots,$ then

$$\mathrm{Dim}_\Delta(X) = \sup_{k\in\mathbb{N}} \mathrm{Dim}_\Delta(X_k),$$

and similarly for dimension and strong dimension in $R(\Delta)$.

Proof. The stability of $\dim_\Delta$ over Δ-unions was proven in [23]. The proof for strong dimension is analogous. □

LEMMA 5.4. *The constructive dimensions are* absolutely stable*; i.e., for all $X \subseteq \mathbf{C}$,*

$$\mathrm{cdim}(X) = \sup_{S\in X} \dim(S)$$

and

$$\mathrm{cDim}(X) = \sup_{S\in X} \mathrm{Dim}(S).$$

Proof. The absolute stability of constructive dimension was proven in [24] using optimal constructive supergales. The same argument works for constructive strong dimension. □

In the following two sections, we use Martin-Löf's definition of randomness [25] as reformulated in terms of martingales by Schnorr [33] as follows.

A *probability measure* on $\mathbf{C}$ is a function $\nu : \{0,1\}^* \to [0,\infty)$ such that $\nu(\lambda) = 1$ and $\nu(w) = \nu(w0) + \nu(w1)$ for all $w \in \{0,1\}^*$. (Intuitively, $\nu(w)$ is the probability that $w \sqsubseteq S$ when the sequence S is "chosen according to ν.")

A *bias* is a real number $\beta \in [0,1]$. Intuitively, if we toss a 0/1-valued coin with bias β, then β is the probability of the outcome 1. A *bias sequence* is a sequence $\vec{\beta} = (\beta_0, \beta_1, \beta_2, \dots)$ of biases. If $\vec{\beta}$ is a bias sequence, then the *$\vec{\beta}$-coin-toss probability measure* is the probability $\mu^{\vec{\beta}}$ on $\mathbf{C}$ defined by

$$\mu^{\vec{\beta}}(w) = \prod_{i=0}^{|w|-1} \beta_i(w), \tag{5.1}$$

where $\beta_i(w) = (2\beta_i - 1)w[i] + (1-\beta_i)$, i.e., $\beta_i(w) = \textbf{if } w[i] \textbf{ then } \beta_i \textbf{ else } 1 - \beta_i$. That is, $\mu^{\vec{\beta}}$ is the probability that $S \in \mathbf{C}_w$ when $S \in \mathbf{C}$ is chosen according to a random experiment in which for each i, independently of all other j, the ith bit of S is decided by tossing a 0/1-valued coin whose probability of 1 is β_i. In the case where the biases β_i are all the same, i.e., $\vec{\beta} = (\beta, \beta, \beta, \dots)$ for some $\beta \in [0,1]$, we write μ^{β} for $\mu^{\vec{\beta}}$, and (5.1) simplifies to

$$\mu^{\beta}(w) = (1-\beta)^{\#(0,w)}\beta^{\#(1,w)}, \tag{5.2}$$

where $\#(b,w)$ is the number of times the bit b appears in the string w. The *uniform probability measure* on $\mathbf{C}$ is the probability measure $\mu = \mu^{\frac{1}{2}}$, for which (5.2) simplifies to

$$\mu(w) = 2^{-|w|}$$

for all $w \in \{0,1\}^*$.

DEFINITION. *Let ν be a probability measure on $\mathbf{C}$.*

1. *A* ν-martingale *is a function $d : \{0,1\}^* \to [0,\infty)$ that satisfies the condition*

$$d(w)\nu(w) = d(w0)\nu(w0) + d(w1)\nu(w1)$$

for all $w \in \{0,1\}^*$.

2. *A ν-martingale is* constructive *if it is lower semicomputable.*

Note that a μ-martingale is a martingale. If $\vec{\beta}$ is a bias sequence, then we call a $\mu^{\vec{\beta}}$-martingale simply a $\vec{\beta}$-martingale.

DEFINITION. *Let ν be a probability measure on $\mathbf{C}$, and let $X \subseteq \mathbf{C}$.*

1. *X has* constructive ν-measure 0*, and we write $\nu_{\text{constr}}(X) = 0$ if there is a constructive ν-martingale d such that $X \subseteq S^{\infty}[d]$.*
2. *X has* constructive ν-measure 1*, and we write $\nu_{\text{constr}}(X) = 1$ if $\nu_{\text{constr}}(\mathbf{C} - X) = 0$.*

DEFINITION. *If ν is a probability measure on $\mathbf{C}$, then a sequence $R \in \mathbf{C}$ is* ν-random*, and we write $R \in \mathrm{RAND}^{\nu}$ if the singleton set $\{R\}$ does not have constructive ν-measure 0 (i.e., there is no constructive ν-martingale that succeeds on R).*

It is well known (and easy to see) that $\nu_{\text{constr}}(\mathrm{RAND}^{\nu}) = 1$.

We write $\mathrm{RAND}^{\vec{\beta}}$ for $\mathrm{RAND}^{\mu^{\vec{\beta}}}$ and RAND for RAND^{μ}.

We also use resource-bounded notions of randomness that have been investigated by Schnorr [34], Lutz [20], Ambos-Spies, Terwijn, and Zheng [2], and others.

DEFINITION. *Let ν be a probability measure on $\mathbf{C}$, and let $t : \mathbb{N} \to \mathbb{N}$.*

1. *A sequence* $R \in \mathbf{C}$ *is* Δ-ν-random, *and we write* $R \in \mathrm{RAND}^{\nu}(\Delta)$, *if there is no* Δ*-computable* ν*-martingale that succeeds on* R*.*
2. *A sequence* $R \in \mathbf{C}$ *is* $t(n)$-ν-random, *and we write* $R \in \mathrm{RAND}^{\nu}(t(n))$, *if there is no* $O(t(n))$*-time-computable* ν*-martingale that succeeds on* R*.*

We write $\mathrm{RAND}^{\vec{\beta}}(t(n))$ for $\mathrm{RAND}^{\mu^{\vec{\beta}}}(t(n))$.

6. Algorithmic information. In this section we present a variety of results and observations in which constructive and computable strong dimensions illuminate or clarify various aspects of algorithmic information theory. Included is our second main theorem, which says that every sequence that is random with respect to a computable sequence of biases $\beta_i \in [\delta, 1/2]$ has the lower and upper average entropies of $(\beta_0, \beta_1, \dots)$ as its dimension and strong dimension, respectively. We also present a result in which finite-state strong dimension clarifies an issue in data compression.

Mayordomo [27] proved that for all $S \in \mathbf{C}$,

$$\dim(S) = \liminf_{n\to\infty} \frac{K(S[0..n-1])}{n}, \tag{6.1}$$

where $K(w)$ is the Kolmogorov complexity of w. (Note: Here and below, $K(w)$ is the "self-delimiting" or "prefix" version of Kolmogorov complexity, as opposed to the "plain" complexity $C(w)$ [19].) Subsequently, Lutz [24] used termgales to define the dimension $\dim(w)$ of each (finite!) string $w \in \{0,1\}^*$ and proved that

$$\dim(S) = \liminf_{n\to\infty} \dim(S[0..n-1]) \tag{6.2}$$

for all $S \in \mathbf{C}$ and

$$K(w) = |w|\dim(w) \pm O(1) \tag{6.3}$$

for all $w \in \{0,1\}^*$, thereby giving a second proof of (6.1). The following theorem is a dual of (6.2) that yields a dual of (6.1) as a corollary.

THEOREM 6.1. *For all* $S \in \mathbf{C}$,

$$\mathrm{Dim}(S) = \limsup_{n\to\infty} \dim(S[0..n-1]).$$

Proof. This proof is analogous to the one for the dual statement (6.2) given in [24]. □

COROLLARY 6.2. *For all* $S \in \mathbf{C}$,

$$\mathrm{Dim}(S) = \limsup_{n\to\infty} \frac{K(S[0..n-1])}{n}.$$

By Corollary 6.2, the "upper algorithmic dimension" defined by Tadaki [42] is precisely the constructive strong dimension.

The rate at which a gambler can increase its capital when betting in a given situation is a fundamental concern of classical and algorithmic information and computational learning theories. In the setting of constructive gamblers, the following quantities are of particular relevance.

DEFINITION. *Let* d *be a supermartingale, let* $S \in \mathbf{C}$*, and let* $X \subseteq \mathbf{C}$*.*

1. *The* lower d-Lyapunov exponent *of* S *is* $\lambda_d(S) = \liminf_{n\to\infty} \frac{\log d(S[0..n-1])}{n}$.
2. *The* upper d-Lyapunov exponent *of* S *is* $\Lambda_d(S) = \limsup_{n\to\infty} \frac{\log d(S[0..n-1])}{n}$.

3. *The* lower Lyapunov exponent *of S is* $\lambda(S) = \sup\{\lambda_d(S) | d$ *is a constructive supermartingale*$\}$.
4. *The* upper Lyapunov exponent *of S is* $\Lambda(S) = \sup\{\Lambda_d(S) | d$ *is a constructive supermartingale*$\}$.
5. *The* lower Lyapunov exponent *of X is* $\lambda(X) = \inf_{S \in X} \lambda(S)$.
6. *The* upper Lyapunov exponent *of X is* $\Lambda(X) = \inf_{S \in X} \Lambda(S)$.

Lyapunov exponents such as these were investigated by Schnorr [34, 36], Ryabko [32], and Staiger [39, 40] (using slightly different notation) prior to the effectivization of Hausdorff dimension. The quantities $\lambda_d(S)$ and $\Lambda_d(S)$ are also called "exponents of increase" of d on S. It is implicit in Staiger's paper [39] that

$$\Lambda_{\text{comp}}(S) = 1 - \dim_{\text{comp}}(S)$$

for all $S \in \mathbf{C}$, where $\Lambda_{\text{comp}}(S)$ is defined like $\Lambda(S)$ above, but with d required to be a computable martingale. Similar reasoning leads to the following characterizations of the Lyapunov exponents.

Theorem 6.3. *Let $S \in \mathbf{C}$ and $X \subseteq \mathbf{C}$. Then $\Lambda(S) = 1 - \dim(S)$, $\lambda(S) = 1 - \mathrm{Dim}(S)$, $\Lambda(X) = 1 - \mathrm{cdim}(X)$, and $\lambda(X) = 1 - \mathrm{cDim}(X)$.*

Proof. We show that $\Lambda(S) = 1 - \dim(S)$. A similar argument shows that $\lambda(S) = 1 - \mathrm{Dim}(S)$. By Lemma 5.4, $\Lambda(X) = 1 - \mathrm{cdim}(X)$ and $\lambda(X) = 1 - \mathrm{cDim}(X)$ follow from the statements about sequences.

Let $t < s < \Lambda(S)$ with t computable, and let d be a constructive supermartingale for which $\Lambda_d(S) > s$. Then for infinitely many n, $d(S[0..n-1]) > 2^{sn}$. Define a constructive $(1-t)$-supergale d' by $d'(w) = 2^{-t|w|}d(w)$ for all $w \in \{0,1\}^*$. Then for infinitely many n, we have $d'(S[0..n-1]) = 2^{-tn}d(S[0..n-1]) > 2^{(s-t)n}$, so $S \in S^\infty[d]$. Therefore $\dim(S) \leq 1 - t$. This holds for all computable $t < \Lambda(S)$, so $\dim(S) \leq 1 - \Lambda(S)$.

Let $s > \dim(S)$ be computable, and let d be a constructive s-gale with $S \in S^\infty[d]$. Define a constructive martingale d' by $d'(w) = 2^{(1-s)|w|}d(w)$ for all $w \in \{0,1\}^*$. For infinitely many n, we have $d(S[0..n-1]) > 1$, and for each of these n, $d'(S[0..n-1]) > 2^{(1-s)n}$. Therefore $\Lambda_{d'}(S) \geq 1 - s$, so $\Lambda(S) \geq 1 - s$. This holds for all $s > \dim(S)$, so $\Lambda(S) \geq 1 - \dim(S)$. □

Constructive strong dimension can also be used to characterize entropy rates of the type investigated by Staiger [38, 39] and Hitchcock [16].

Definition. *Let $A \subseteq \{0,1\}^*$.*

1. *The* entropy rate *of $A \subseteq \{0,1\}^*$ is* $H_A = \limsup_{n \to \infty} \frac{\log |A_{=n}|}{n}$.
2. *We define the sets of sequences*

$$A^{\text{i.o.}} = \{S \in \mathbf{C} | (\exists^\infty n) S[0..n-1] \in A\}$$

and

$$A^{\text{a.e.}} = \{S \in \mathbf{C} | (\forall^\infty n) S[0..n-1] \in A\}.$$

Definition. *Let $X \subseteq \mathbf{C}$. The* constructive entropy rate *of X is*

$$\mathcal{H}_{\text{CE}}(X) = \inf\{H_A | X \subseteq A^{\text{i.o.}} \textit{ and } A \in \text{CE}\},$$

and the constructive strong entropy rate *of X is*

$$\mathcal{H}^{\text{str}}_{\text{CE}}(X) = \inf\{H_A | X \subseteq A^{\text{a.e.}} \textit{ and } A \in \text{CE}\}.$$

Hitchcock [16] proved that

$$\mathcal{H}_{\mathrm{CE}}(X) = \mathrm{cdim}(X) \tag{6.4}$$

for all $X \subseteq \mathbf{C}$. We have the following dual of (6.4).

THEOREM 6.4. *For any $X \subseteq \mathbf{C}$, $\mathcal{H}^{\mathrm{str}}_{\mathrm{CE}}(X) = \mathrm{cDim}(X)$.*

Proof. This proof is analogous to the proof of (6.4) given in [16]. □

In the classical case, Tricot [43] has defined a set to be *regular* if its Hausdorff and packing dimensions coincide, and defined its *irregularity* to be the difference between these two fractal dimensions. Analogously, we define the c-*irregularity* (i.e., constructive irregularity) of a sequence $S \in \mathbf{C}$ to be $\mathrm{Dim}(S) - \dim(S)$, and we define the c-*irregularity* of a set $X \subseteq \mathbf{C}$ to be $\mathrm{cDim}(X) - \mathrm{cdim}(X)$. We define a sequence or set to be c-*regular* (i.e., *constructively regular*) if its c-irregularity is 0.

As the following result shows, the c-irregularity of a sequence may be any real number in $[0, 1]$.

THEOREM 6.5. *For any two real numbers $0 \leq \alpha \leq \beta \leq 1$, there is a sequence $S \in \mathbf{C}$ such that $\dim(S) = \alpha$ and $\mathrm{Dim}(S) = \beta$.*

Proof. Let $R \in \mathrm{RAND}$ be a random sequence. It is well known that

$$K(R[0..n-1]) \geq n - O(1). \tag{6.5}$$

Write $R = r_1 r_2 r_3 \ldots$, where $|r_n| = 2n - 1$ for all n. Note that $|r_1 \cdots r_n| = n^2$.

For each n, define

$$\gamma_n = \begin{cases} \frac{1-\alpha}{\alpha} & \text{if } \log^* n \text{ is odd,} \\ \frac{1-\beta}{\beta} & \text{if } \log^* n \text{ is even,} \end{cases}$$

and let

$$k_n = \lceil |r_n| \gamma_n \rceil .$$

We now define $S \in \mathbf{C}$ as

$$S = r_1 0^{k_1} r_2 0^{k_2} \cdots r_n 0^{k_n} \cdots \ .$$

Note that for all n,

$$\begin{aligned} |r_n 0^{k_n}| &= \lceil |r_n|(1 + \gamma_n) \rceil \\ &= \begin{cases} \left\lceil \frac{1}{\alpha} |r_n| \right\rceil & \text{if } \log^* n \text{ is odd,} \\ \left\lceil \frac{1}{\beta} |r_n| \right\rceil & \text{if } \log^* n \text{ is even.} \end{cases} \end{aligned}$$

Let $w \sqsubseteq S$. Then for some n,

$$w = r_1 0^{k_1} \cdots r_{n-1} 0^{k_{n-1}} r'_n 0^j,$$

where $r'_n \sqsubseteq r_n$ and $0 \leq j \leq k_n$. We have

$$\begin{aligned} K(w) &\leq K(r_1 \cdots r_{n-1} r'_n) + K(k_1) + \cdots + K(k_{n-1}) + K(j) + O(1) \\ &\leq |r_1 \cdots r_{n-1} r'_n| + O(n \log n) \\ &\leq (n-1)^2 + O(n \log n). \end{aligned} \tag{6.6}$$

Also,

$$\begin{aligned} K(r_1 \cdots r_{n-1} r'_n) &\le K(w) + K(k_1) + \cdots + K(k_{n-1}) + K(j) + O(1) \\ &\le K(w) + O(n \log n), \end{aligned}$$

and so by (6.5),

$$\begin{aligned} K(w) &\ge K(r_1 \cdots r_{n-1} r'_n) - O(n \log n) \\ &\ge |r_1 \cdots r_{n-1} r'_n| - O(n \log n) \\ &\ge (n-1)^2 - O(n \log n). \end{aligned} \tag{6.7}$$

We bound the length of w in terms of n as

$$\begin{aligned} |w| &\ge |r_1|(1+\gamma_1) + \cdots + |r_{n-1}|(1+\gamma_{n-1}) + |r'_n| \\ &\ge \frac{|r_1 \cdots r_{n-1}|}{\beta} \\ &= \frac{1}{\beta}(n-1)^2 \end{aligned} \tag{6.8}$$

and

$$\begin{aligned} |w| &\le |r_1|(1+\gamma_1) + \cdots + |r_{n-1}|(1+\gamma_{n-1}) + |r_n|(1+\gamma_n) + n \\ &\le \frac{|r_1 \cdots r_{n-1} r_n|}{\alpha} + n \\ &\le \frac{1}{\alpha}(n+1)^2. \end{aligned} \tag{6.9}$$

From (6.6) and (6.8), we have

$$\limsup_{m\to\infty} \frac{K(S[0..m-1])}{m} \le \limsup_{n\to\infty} \frac{(n-1)^2 + O(n \log n)}{\frac{1}{\beta}(n-1)^2} = \beta, \tag{6.10}$$

and (6.7) and (6.9) yield

$$\liminf_{m\to\infty} \frac{K(S[0..m-1])}{m} \ge \liminf_{n\to\infty} \frac{(n-1)^2 - O(n \log n)}{\frac{1}{\alpha}(n+1)^2} = \alpha. \tag{6.11}$$

For each n, let

$$w_n = r_1 0^{k_1} \cdots r_n 0^{k_n}.$$

Recall the sequence of *towers* defined by t_j by $t_0 = 1$ and $t_{j+1} = 2^{t_j}$. If j is even, then for all $t_{j-1} < i \le t_j$, $\gamma_i = \frac{1-\beta}{\beta}$. Then

$$\begin{aligned} |w_{t_j}| &\le t_j + \sum_{i=1}^{t_j} |r_i|(1+\gamma_i) \\ &= t_j + \sum_{i=1}^{t_{j-1}} |r_i|(1+\gamma_i) + \frac{1}{\beta} \sum_{i=t_{j-1}+1}^{t_j} |r_i| \\ &\le t_j + \frac{1}{\alpha} t_{j-1}^2 + \frac{1}{\beta}(t_j^2 - t_{j-1}^2) \\ &\le \frac{1}{\beta} t_j^2 + t_j + O((\log t_j)^2). \end{aligned} \tag{6.12}$$

Similarly, if j is odd, we have

$$
\begin{aligned}
|w_{t_j}| &\geq \sum_{i=1}^{t_j} |r_i|(1+\gamma_i) \\
&= \sum_{i=1}^{t_{j-1}} |r_i|(1+\gamma_i) + \frac{1}{\alpha} \sum_{i=t_{j-1}+1}^{t_j} |r_i| \\
&\geq \frac{1}{\beta} {t_{j-1}}^2 + \frac{1}{\alpha}(t_j^2 - t_{j-1}^2) \\
&\geq \frac{1}{\alpha} t_j^2 - O((\log t_j)^2).
\end{aligned} \tag{6.13}
$$

Combining (6.7) and (6.12), we have

$$
\limsup_{m\to\infty} \frac{K(S[0..m-1])}{m} \geq \limsup_{n\to\infty} \frac{K(w_{t_{2n}})}{|w_{t_{2n}}|} \geq \beta. \tag{6.14}
$$

Putting (6.6) together with (6.13) yields

$$
\liminf_{m\to\infty} \frac{K(S[0..m-1])}{m} \leq \liminf_{n\to\infty} \frac{K(w_{t_{2n+1}})}{|w_{t_{2n+1}}|} \leq \alpha. \tag{6.15}
$$

By (6.1), (6.11), and (6.15), we have $\dim(S) = \alpha$. By Corollary 6.2, (6.10), and (6.14), we have $\mathrm{Dim}(S) = \beta$. □

We now come to the main theorem of this section. The following notation simplifies its statement and proof.

Notation. Given a bias sequence $\vec{\beta} = (\beta_0, \beta_1, \dots)$, $n \in \mathbb{N}$, and $S \in \mathbf{C}$, let

$$
\begin{aligned}
H_n(\vec{\beta}) &= \frac{1}{n}\sum_{i=0}^{n-1} \mathcal{H}(\beta_i), \\
H^-(\vec{\beta}) &= \liminf_{n\to\infty} H_n(\vec{\beta}), \\
H^+(\vec{\beta}) &= \limsup_{n\to\infty} H_n(\vec{\beta}).
\end{aligned}
$$

We call $H^-(\vec{\beta})$ and $H^+(\vec{\beta})$ the *lower* and *upper average entropies*, respectively, of $\vec{\beta}$.

THEOREM 6.6. *If $\delta \in (0, \frac{1}{2}]$ and $\vec{\beta}$ is a computable bias sequence with each $\beta_i \in [\delta, \frac{1}{2}]$, then for every sequence $R \in \mathrm{RAND}^{\vec{\beta}}$,*

$$
\dim(R) = H^-(\vec{\beta}) \quad \textit{and} \quad \mathrm{Dim}(R) = H^+(\vec{\beta}).
$$

Theorem 6.6 says that every sequence that is random with respect to a suitable bias sequence $\vec{\beta}$ has the lower and upper average entropies of $\vec{\beta}$ as its dimension and strong dimension, respectively. Since there exist $\vec{\beta}$-random sequences in Δ_2^0 when $\vec{\beta}$ is computable, this gives a powerful and flexible method for constructing Δ_2^0 sequences with given (Δ_2^0-computable) dimensions and strong dimensions.

Note that Theorem 6.6 also gives an alternative, though less constructive, proof of Theorem 6.5.

We now develop a sequence of results that are used in our proof of Theorem 6.6.

LEMMA 6.7. *Assume that $\delta > 0$, $\epsilon > 0$, and that, for each $\beta \in [\delta, 1-\delta]$, η_β is a bounded random variable with expectation $\mathrm{E}\eta_\beta \leq -\epsilon$ and $\mathrm{E}e^{t\eta_\beta}$ is, uniformly in t, a continuous function of β. Then there exists $\theta > 0$ such that, for all $\beta \in [\delta, 1-\delta]$ and $t \in (0, \theta]$,*

$$\mathrm{E}e^{t\eta_\beta} < 1 - \frac{t\epsilon}{2}.$$

Proof. Assume the hypothesis. Then the dominated convergence theorem [3] tells us that, for all $\beta \in [\delta, 1-\delta]$,

$$\begin{aligned}\lim_{t\to 0^+} \frac{\mathrm{E}e^{t\eta_\beta} - 1}{t} &= \lim_{t\to 0^+} \mathrm{E}\frac{e^{t\eta_\beta} - 1}{t} \\ &= \mathrm{E}\left(\lim_{t\to 0^+} \frac{e^{t\eta_\beta} - 1}{t}\right) \\ &= \mathrm{E}\left(\eta_\beta \lim_{t\to 0^+} \frac{e^{t\eta_\beta} - 1}{t\eta_\beta}\right) \\ &= \mathrm{E}\eta_\beta \\ &\leq -\epsilon.\end{aligned}$$

Hence, for each $\beta \in [\delta, 1-\delta]$, there exists $t_\beta > 0$ such that, for all $t \in (0, t_\beta]$,

$$\frac{\mathrm{E}e^{t\eta_\beta} - 1}{t} < -\frac{3\epsilon}{4}.$$

It follows by our continuity hypothesis that, for each $\beta \in [\delta, 1-\delta]$, there is an open neighborhood N_β of β such that, for all $t \in (0, t_\beta]$ and $\gamma \in N_\beta \cap [\delta, 1-\delta]$,

$$\frac{\mathrm{E}e^{t\eta_\gamma} - 1}{t} < -\frac{\epsilon}{2}.$$

The family $\mathcal{G} = \{N_\beta \mid \beta \in [\delta, 1-\delta]\}$ is an open cover of the compact set $[\delta, 1-\delta]$, so there is a finite set $B \subseteq [\delta, 1-\delta]$ such that the subcollection $\mathcal{G}' = \{N_\beta \mid \beta \in B\}$ is also a cover of $[\delta, 1-\delta]$. Let

$$\theta = \min\{t_\beta \mid \beta \in B\}.$$

Then $\theta > 0$ and, for all $\beta \in [\delta, 1-\delta]$ and $t \in (0, \theta]$,

$$\frac{\mathrm{E}e^{t\eta_\beta} - 1}{t} < -\frac{\epsilon}{2},$$

whence

$$\mathrm{E}e^{t\eta_\beta} < 1 - \frac{t\epsilon}{2}. \qquad \square$$

COROLLARY 6.8. *For each $\delta > 0$ and $\epsilon > 0$, there exists $\theta > 0$ such that, for all $\beta \in [\delta, 1-\delta]$, if we choose $a \in \{0, 1\}$ with $\mathrm{Prob}[a = 1] = \beta$, and if*

$$\eta = \xi - \mathcal{H}(\beta) - \epsilon$$

or

$$\eta = \mathcal{H}(\beta) - \xi - \epsilon,$$

where

$$\xi = (1-a)\log\frac{1}{1-\beta} + a\log\frac{1}{\beta},$$

then

$$\mathrm{E}e^{\theta\eta} < 1 - \frac{\theta\epsilon}{2}.$$

Proof. The random variables

$$\eta_{1,\beta} = \xi - \mathcal{H}(\beta) - \epsilon,$$
$$\eta_{2,\beta} = \mathcal{H}(\beta) - \xi - \epsilon$$

satisfy the hypothesis of Lemma 6.7 with $\mathrm{E}\eta_{1,\beta} = \mathrm{E}\eta_{2,\beta} = -\epsilon$, so we can choose $\theta_1 > 0$ for $\eta_{1,\beta}$ and $\theta_2 > 0$ for $\eta_{2,\beta}$ as in that lemma. Letting $\theta = \min\{\theta_1, \theta_2\}$ establishes the corollary. □

Notation. Given a bias sequence $\vec{\beta} = (\beta_0, \beta_1, \dots)$, $n \in \mathbb{N}$, and $S \in \mathbf{C}$, let

$$L_n(\vec{\beta})(S) = \log\frac{1}{\mu^{\vec{\beta}}(S[0..n-1])} = \sum_{i=0}^{n-1}\xi_i(S),$$

where

$$\xi_i(S) = (1 - S[i])\log\frac{1}{1-\beta_i} + S[i]\log\frac{1}{\beta_i}$$

for $0 \le i < n$.

Note that $L_n(\vec{\beta}), \xi_0, \dots, \xi_{n-1}$ are random variables with

$$\mathrm{E}L_n(\vec{\beta}) = \sum_{i=0}^{n-1}\mathrm{E}\xi_i = \sum_{i=0}^{n-1}\mathcal{H}(\beta_i) = nH_n(\vec{\beta}).$$

The following large deviation theorem tells us that $L_n(\vec{\beta})$ is very unlikely to deviate significantly from this expected value.

THEOREM 6.9. *For each $\delta > 0$ and $\epsilon > 0$, there exists $\alpha \in (0,1)$ such that, for all bias sequences $\vec{\beta} = (\beta_0, \beta_1, \dots)$ with each $\beta_i \in [\delta, 1-\delta]$ and all $n \in \mathbb{Z}^+$, if $L_n(\vec{\beta})$ and $H_n(\vec{\beta})$ are defined as above, then*

$$\mathrm{P}\left[|L_n(\vec{\beta}) - nH_n(\vec{\beta})| \ge \epsilon n\right] < 2\alpha^n,$$

where the probability is computed according to $\mu^{\vec{\beta}}$.

Proof. Let $\delta > 0$ and $\epsilon > 0$, and choose $\theta > 0$ as in Corollary 6.8. Let $\alpha = 1 - \frac{\theta\epsilon}{2}$, noting that $\alpha \in (0,1)$. Let $\vec{\beta}$ be as given, and let $n \in \mathbb{Z}^+$. Let $L = L_n(\vec{\beta})$, $H = H_n(\vec{\beta})$, and $\xi_0, \xi_1, \dots$ be as above. The proof is in two parts.

1. For each $i \in \mathbb{N}$, let $\eta_i = \xi_i - \mathcal{H}(\beta_i) - \epsilon$. Then Markov's inequality, indepen-

dence, and Corollary 6.8 tell us that

$$\begin{aligned}\mathrm{P}[L - nH \geq \epsilon n] &= P[e^{\theta(L-nH)} \geq e^{\theta \epsilon n}] \\ &\leq e^{-\theta\epsilon n}\mathrm{E}e^{\theta(L-nH)} \\ &= \mathrm{E}e^{\theta(L-nH)-\epsilon\theta n} \\ &= \mathrm{E}e^{\theta\sum_{i=0}^{n-1}\eta_i} \\ &= \mathrm{E}\prod_{i=0}^{n-1} e^{\theta\eta_i} \\ &= \prod_{i=0}^{n-1}\mathrm{E}e^{\theta\eta_i} \\ &< \alpha^n.\end{aligned}$$

2. Arguing as in part 1 with $\eta_i = \mathcal{H}(\beta_i) - \xi_i - \epsilon$ shows that $\mathrm{P}[nH - L \geq \epsilon n] < \alpha^n$. By parts 1 and 2 of this proof, we now have

$$\mathrm{P}[|L - nH| \geq \epsilon n] < 2\alpha^n. \qquad \square$$

Some of our arguments are simplified by the following constructive version of a classical theorem of Kakutani [17]. Say that two bias sequences $\vec{\beta}$ and $\vec{\beta}'$ are *square-summably equivalent*, and write $\vec{\beta} \approx^2 \vec{\beta}'$ if $\sum_{i=0}^{\infty}(\beta_i - \beta_i')^2 < \infty$.

THEOREM 6.10 (van Lambalgen [44, 45], Vovk [47]). *Let $\delta > 0$, and let $\vec{\beta}$ and $\vec{\beta}'$ be computable bias sequences with $\beta_i, \beta_i' \in [\delta, 1-\delta]$ for all $i \in \mathbb{N}$.*

1. *If $\vec{\beta} \approx^2 \vec{\beta}'$, then $\mathrm{RAND}^{\vec{\beta}} = \mathrm{RAND}^{\vec{\beta}'}$.*
2. *If $\vec{\beta} \not\approx^2 \vec{\beta}'$, then $\mathrm{RAND}^{\vec{\beta}} \bigcap \mathrm{RAND}^{\vec{\beta}'} = \emptyset$.*

COROLLARY 6.11. *If $\delta > 0$ and $\vec{\beta}$ is a computable bias sequence with each $\beta_i \in [\delta, 1-\delta]$, then there is an exactly computable bias sequence $\vec{\beta}'$ with each $\beta_i' \in [\frac{\delta}{2}, \beta_i]$ satisfying $\mathrm{RAND}^{\vec{\beta}'} = \mathrm{RAND}^{\vec{\beta}}$.*

Proof. Assume the hypothesis. Then there is a computable function $g : \mathbb{N}\times\mathbb{N} \to \mathbb{Q}$ such that $|g(i,r) - \beta_i| \leq 2^{-r}$ for all $i, r \in \mathbb{N}$. Let $m = 2 + \lceil \log \frac{1}{\delta} \rceil$, and let

$$\beta_i' = g(i, m+i) - 2^{-(m+i)}$$

for all $i \in \mathbb{N}$. It is easily verified that $\vec{\beta}'$ is exactly computable, each $\beta_i' \in [\frac{\delta}{2}, \beta_i]$, and $\vec{\beta}' \approx^2 \vec{\beta}$, whence Theorem 6.10 tells us that $\mathrm{RAND}^{\vec{\beta}'} = \mathrm{RAND}^{\vec{\beta}}$. $\square$

LEMMA 6.12. *If $\delta > 0$ and $\vec{\beta}$ is a computable bias sequence with each $\beta_i \in [\delta, 1-\delta]$, then every sequence $R \in \mathrm{RAND}^{\vec{\beta}}$ satisfies*

$$L_n(\vec{\beta})(R) = nH_n(\vec{\beta}) + o(n)$$

as $n \to \infty$.

Proof. Assume the hypothesis. By Corollary 6.11, we can assume that $\vec{\beta}$ is exactly computable. Let $\epsilon > 0$. For each $n \in \mathbb{N}$, define the set

$$Y_n = \left\{ S \in \mathbf{C} \;\middle|\; |L_n(\vec{\beta})(S) - nH_n(\vec{\beta})| \geq \epsilon n \right\},$$

and let

$$X_\epsilon = \{S \in \mathbf{C} \mid (\exists^\infty n) S \in Y_n\}.$$

It suffices to show that $\mu^{\vec{\beta}}_{\text{comp}}(X_\epsilon) = 0$.

For each $n \in \mathbb{N}$ and $w \in \{0,1\}^*$, let

$$d_n(w) = \begin{cases} \mu^{\vec{\beta}}(Y_n | \mathbf{C}_w) & \text{if } |w| \leq n, \\ d_n(w[0..n-1]) & \text{if } |w| > n. \end{cases}$$

It is easily verified that each d_n is a $\vec{\beta}$-martingale and that the function $(n, w) \mapsto d_n(w)$ is computable. It is clear that $Y_n \subseteq S^1[d_n]$ for all $n \in \mathbb{N}$, where $S^1[d_n] = \bigcup_{d_n(w) \geq 1} \mathbf{C}_w$. Finally, by Theorem 6.9, the series $\sum_{n=0}^{\infty} d_n(\lambda)$ is computably convergent, so the computable first Borel–Cantelli lemma [20] (extended to $\vec{\beta}$ as indicated in [4]) tells us that $\mu^{\vec{\beta}}_{\text{comp}}(X_\epsilon) = 0$. □

LEMMA 6.13. *If $\delta > 0$ and $\vec{\beta}$ is a computable bias sequence with each $\beta_i \in [\delta, \frac{1}{2}]$, then* $\text{cdim}(\text{RAND}^{\vec{\beta}}) \leq H^-(\vec{\beta})$ *and* $\text{cDim}(\text{RAND}^{\vec{\beta}}) \leq H^+(\vec{\beta})$.

Proof. Assume the hypothesis. By Corollary 6.11, we can assume that $\vec{\beta}$ is exactly computable. Let $s \in [0, \infty)$ be computable.

Define $d : \{0,1\}^* \to [0, \infty)$ by

$$d(w) = 2^{s|w|} \mu^{\vec{\beta}}(w)$$

for all $w \in \{0,1\}^*$. Then d is a constructive (in fact, computable) s-gale. For each $R \in \mathbf{C}$ and $n \in \mathbb{N}$, if we write $z_n = R[0..n-1]$, then

$$\log d(z_n) = sn + \log \mu^{\vec{\beta}}(z_n)$$

for all n. In particular, if $R \in \text{RAND}^{\vec{\beta}}$, if follows by Lemma 6.12 that

$$\log d(z_n) = n[s - H_n(\vec{\beta})] + o(n) \tag{6.16}$$

as $n \to \infty$. We now verify the two parts of the lemma. For both parts, we let

$$I_\epsilon = \{n \in \mathbb{N} \mid H_n(\vec{\beta}) < s - \epsilon\}.$$

To see that $\text{cdim}(\text{RAND}^{\vec{\beta}}) \leq H^-(\vec{\beta})$, let $s > H^-(\vec{\beta})$, and let $\epsilon = \frac{s - H^-(\vec{\beta})}{2}$. Then the set I_ϵ is infinite, so (6.16) tells us that $\text{RAND}^{\vec{\beta}} \subseteq S^\infty[d]$, whence $\text{cdim}(\text{RAND}^{\vec{\beta}}) \leq s$.

To see that $\text{cDim}(\text{RAND}^{\vec{\beta}}) \leq H^+(\vec{\beta})$, let $s > H^+(\vec{\beta})$, and let $\epsilon = \frac{s - H^+(\vec{\beta})}{2}$. Then the set I_ϵ is cofinite, so (6.16) tells us that $\text{RAND}^{\vec{\beta}} \subseteq S^\infty_{\text{str}}[d]$, whence $\text{cDim}(\text{RAND}^{\vec{\beta}}) \leq s$. □

LEMMA 6.14. *Assume that $\delta > 0$, $\vec{\beta}$ is a computable bias sequence with each $\beta_i \in [\delta, 1 - \delta]$, $s \in [0, \infty)$ is computable, and d is a constructive s-gale.*

1. *If $s < H^-(\vec{\beta})$, then $S^\infty[d] \bigcap \text{RAND}^{\vec{\beta}} = \emptyset$.*
2. *If $s < H^+(\vec{\beta})$, then $S^\infty_{\text{str}}[d] \bigcap \text{RAND}^{\vec{\beta}} = \emptyset$.*

Proof. Assume the hypothesis. Define $d' : \{0,1\}^* \to [0, \infty)$ by

$$d'(w) = \frac{d(w)}{2^{s|w|} \mu^{\vec{\beta}}(w)}$$

for all $w \in \{0,1\}^*$. Then d' is a $\vec{\beta}$-martingale, and d' is clearly constructive.

Let $R \in \mathrm{RAND}^{\vec{\beta}}$. Then d' does not succeed on R, so there is a constant $c > 0$ such that, for all $n \in \mathbb{N}$, if we write $z_n = R[0..n-1]$, then $d'(z_n) \le 2^c$, whence

$$\log d(z_n) \le c + sn + \log \mu^{\vec{\beta}}(z_n).$$

It follows by Lemma 6.12 that

$$\log d(z_n) \le c + n[s - H_n(\vec{\beta})] + o(n)$$

as $n \to \infty$. Hence, for any $\epsilon > 0$, if we let

$$I_\epsilon = \{n \in \mathbb{Z}^+ \mid s < H_n(\vec{\beta}) - \epsilon\},$$

then $\log d(z_n) < c$ for all sufficiently large $n \in I_\epsilon$. We now verify the two parts of the lemma.

1. If $s < H^-(\vec{\beta})$, let $\epsilon = \frac{H^-(\vec{\beta})-s}{2}$. Then I_ϵ is cofinite, so $\log d(z_n) < c$ for all sufficiently large $n \in \mathbb{Z}^+$, so $R \notin S^\infty[d]$.
2. If $s < H^+(\vec{\beta})$, let $\epsilon = \frac{H^+(\vec{\beta})-s}{2}$. Then I_ϵ is infinite, so $\log d(z_n) < c$ for infinitely many $n \in \mathbb{Z}^+$, so $R \notin S^\infty_{\mathrm{str}}[d]$. □

We now have all we need to prove the main theorem of this section.

Proof of Theorem 6.6. Assume the hypothesis, and let $R \in \mathrm{RAND}^{\vec{\beta}}$. By Lemma 6.13, $\dim(R) \le H^-(\vec{\beta})$ and $\mathrm{Dim}(R) \le H^+(\vec{\beta})$. To see that $\dim(R) \ge H^-(\vec{\beta})$ and $\mathrm{Dim}(R) \ge H^+(\vec{\beta})$, let $s, t \in [0, \infty)$ be computable with $s < H^-(\vec{\beta})$ and $t < H^+(\vec{\beta})$, let d^- be a constructive s-gale, and let d^+ be a constructive t-gale. It suffices to show that $R \notin S^\infty[d^-]$ and $R \notin S^\infty_{\mathrm{str}}[d^+]$. But these follow immediately from Lemma 6.14 and the $\vec{\beta}$-randomness of R. □

Corollary 6.15. *If $\vec{\beta}$ is a computable sequence of coin-toss biases such that $\overline{H}(\vec{\beta}) = \lim_{n\to\infty} H_n(\vec{\beta}) \in (0,1)$, then every sequence $R \in \mathbf{C}$ that is random with respect to $\vec{\beta}$ is c-regular, with* $\dim(R) = \mathrm{Dim}(R) = \overline{H}(\vec{\beta})$.

Note that Corollary 6.15 strengthens Theorem 7.6 of [24] because the convergence of $H_n(\vec{\beta})$ is a weaker hypothesis than the convergence of $\vec{\beta}$.

Generalizing the construction of Chaitin's random real number Ω [6], Mayordomo [27] and, independently, Tadaki [42] defined for each $s \in (0,1]$ and each infinite computably enumerable set $A \subseteq \{0,1\}^*$ the real number

$$\theta^s_A = \sum \left\{ 2^{-\frac{|\pi|}{s}} \,\middle|\, \pi \in \{0,1\}^* \text{ and } U(\pi) \in A \right\},$$

where U is a universal self-delimiting Turing machine. Given (6.1) and Corollary 6.2 above, the following fact is implicit in Tadaki's paper.

Theorem 6.16 (Tadaki [42]). *For each $s \in (0,1]$ and each infinite computably enumerable set $A \subseteq \{0,1\}^*$, the (binary expansion of the) real number θ^s_A is c-regular with* $\dim(\theta^s_A) = \mathrm{Dim}(\theta^s_A) = s$.

We define a set $X \subseteq \mathbf{C}$ to be *self-similar* if it has the form

$$X = A^\infty = \{S \in \mathbf{C} \mid S = w_0 w_1 w_2 \dots \text{ for some } w_0, w_1, w_2, \dots \in A\},$$

where $A \subseteq \{0,1\}^*$ is a finite prefix set. Self-similar sets are examples of c-regular sets.

Theorem 6.17. *Let $X = A^\infty$ be self-similar, where A is a finite prefix set. Then X is c-regular, with* $\mathrm{cdim}(X) = \mathrm{cDim}(X) = \inf\{s \mid \sum_{w\in A} 2^{-s|w|} \le 1\}$.

Proof. We say that a string w is *composite* if there are strings $w_1, \dots, w_k \in A$ such that $w = w_1 \cdots w_k$. Let s be computable such that $\sum_{w\in A} 2^{-s|w|} \le 1$. For any computable $\epsilon > 0$ we define a constructive $(s+\epsilon)$-supergale d as follows. Let $w \in \{0,1\}^*$, and let v be the maximal composite proper prefix of w. Then

$$d(w) = \sum_{u \in A : w \sqsubseteq vu} 2^{\epsilon|w|} 2^{-s(|vu|-|w|)}.$$

For all composite strings w, we have $d(w) = 2^{\epsilon|w|}$. It follows that $A^\infty \subseteq S^\infty_{\mathrm{str}}[d]$, and therefore $\mathrm{cDim}(A^\infty) \leq s + \epsilon$.

Let s be such that $\sum_{w \in A} 2^{-s|w|} > 1$, and let d be an s-gale. To show that $\mathrm{cdim}(A^\infty) > s$, it suffices to construct a sequence $S \in A^\infty - S^\infty[d]$. Initially, we let $w_0 = \lambda$. Assume that w_n has been defined, and let $u \in A$ such that $d(w_n u) \leq d(w_n)$. We know that such a u exists because of our choice of s. Then we let $w_{n+1} = w_n u$. Our sequence S is the unique one that has $w_n \sqsubseteq S$ for all n. □

Dai et al. [7] investigated the *finite-state compression ratio* $\rho_{\mathrm{FS}}(S)$, defined for each sequence $S \in \mathbf{C}$ to be the infimum, taken over all information-lossless finite-state compressors C (a model defined in Shannon's 1948 paper [37]) of the *(lower) compression ratio*

$$\rho_C(S) = \liminf_{n\to\infty} \frac{|C(S[0..n-1])|}{n}.$$

They proved that

$$\rho_{\mathrm{FS}}(S) = \dim_{\mathrm{FS}}(S) \tag{6.17}$$

for all $S \in \mathbf{C}$. However, it has been pointed out that the compression ratio $\rho_{\mathrm{FS}}(S)$ differs from the one investigated by Ziv [48]. Ziv was instead concerned with the ratio $R_{\mathrm{FS}}(S)$ defined by

$$R_{\mathrm{FS}}(S) = \inf_{k\in\mathbb{N}} \limsup_{n\to\infty} \inf_{C\in\mathcal{C}_k} \frac{|C(S[0..n-1])|}{n},$$

where $\mathcal{C}_k$ is the set of all k-state information-lossless finite-state compressors. The following result, together with (6.17), clarifies the relationship between $\rho_{\mathrm{FS}}(S)$ and $R_{\mathrm{FS}}(S)$.

THEOREM 6.18. *For all $S \in \mathbf{C}$, $R_{\mathrm{FS}}(S) = \mathrm{Dim}_{\mathrm{FS}}(S)$.*

The proof of Theorem 6.18 is based on the following lemma.

LEMMA 6.19. *Let $\mathcal{C}$ be the set of all finite-state compressors. For all $S \in \mathbf{C}$,*

$$R_{\mathrm{FS}}(S) = \inf_{C\in\mathcal{C}} \limsup_{n\to\infty} \frac{|C(S[0..n-1])|}{n}.$$

Proof. For each $k \in \mathbb{N}$ let $\mathcal{C}'_k$ be the set of all k-state information-lossless finite-state compressors whose output length per input bit is bounded by k. Notice that $\mathcal{C}'_k$ is a subset of $\mathcal{C}_k$ and that $\mathcal{C}'_k$ is finite. Ziv and Lempel implicitly prove in [49] that the following equality holds:

$$R_{\mathrm{FS}}(S) = \inf_{k\in\mathbb{N}} \limsup_{n\to\infty} \inf_{C\in\mathcal{C}'_k} \frac{|C(S[0..n-1])|}{n}.$$

Let

$$R'_{\mathrm{FS}}(S) = \inf_{C\in\mathcal{C}} \limsup_{n\to\infty} \frac{|C(S[0..n-1])|}{n}.$$

The inequality $R_{\mathrm{FS}}(S) \leq R'_{\mathrm{FS}}(S)$ is trivial. We use several results from [7] to obtain for each $k \in \mathbb{N}$ and $\epsilon > 0$ a finite-state compressor $C_{k,\epsilon}$ that is nearly optimal for all compressors in $\mathcal{C}'_k$. From Lemma 7.7 in [7] we obtain a finite-state gambler for each $C \in \mathcal{C}'_k$. By Lemma 3.7 in [7], we can combine these gamblers into a single finite-state gambler. Theorem 4.5 and Lemma 3.11 in [7] convert this single gambler into a 1-account nonvanishing finite-state gambler, and finally Lemma 7.10 converts this to the finite-state compressor $C_{k,\epsilon}$. Combining the five cited constructions in [7], we obtain that there is a constant $c_{k,\epsilon}$ such that, for all $w \in \{0,1\}^*$ and $C \in \mathcal{C}'_k$,

$$|C_{k,\epsilon}(w)| \leq |C(w)| + \epsilon|w| + c_{k,\epsilon}.$$

Then for all $k \in \mathbb{N}$ and $\epsilon > 0$,

$$\begin{aligned} R'_{\mathrm{FS}}(S) &\leq \limsup_{n\to\infty} \frac{|C_{k,\epsilon}(S[0..n-1])|}{n} \\ &\leq \limsup_{n\to\infty} \inf_{C\in\mathcal{C}'_k} \frac{|C(S[0..n-1])|}{n} + \epsilon, \end{aligned}$$

so $R'_{\mathrm{FS}}(S) \leq R_{\mathrm{FS}}(S)$. □

Proof of Theorem 6.18. The equality

$$\mathrm{Dim}_{\mathrm{FS}}(S) = \inf_{C\in\mathcal{C}} \limsup_{n\to\infty} \frac{|C(S[0..n-1])|}{n}$$

has a proof analogous to that of (6.17) given in [7]. Together with Lemma 6.19, this implies that $R_{\mathrm{FS}}(S) = \mathrm{Dim}_{\mathrm{FS}}(S)$. □

Thus, mathematically, the compression ratios $\rho_{\mathrm{FS}}(S)$ and $R_{\mathrm{FS}}(S)$ are both natural: they are the finite-state effectivizations of the Hausdorff and packing dimensions, respectively.

7. Computational complexity. In this section we prove our third main theorem, which says that the dimensions and strong dimensions of polynomial-time many-one degrees in exponential time are essentially unrestricted. Our proof of this result uses Theorem 6.9 and convenient characterizations of p-dimension and strong p-dimension in terms of feasible unpredictability.

DEFINITION. *A* predictor *is a function* $\pi : \{0,1\}^* \times \{0,1\} \to [0,1]$ *such that, for all* $w \in \{0,1\}^*$, $\pi(w,0) + \pi(w,1) = 1$.

We interpret $\pi(w,b)$ as the predictor's estimate of the probability that the bit b will occur next, given that w has occurred. We write $\Pi(\mathrm{p})$ for the class of all feasible predictors.

DEFINITION. *Let* $w \in \{0,1\}^*$, $S \in \mathbf{C}$, *and* $X \subseteq \mathbf{C}$.

1. *The* cumulative log-loss *of* π *on* w *is*

$$\mathcal{L}^{\log}(\pi, w) = \sum_{i=0}^{|w|-1} \log \frac{1}{\pi(w[0..i-1], w[i])}.$$

2. *The* log-loss rate *of* π *on* S *is*

$$\mathcal{L}^{\log}(\pi, S) = \liminf_{n\to\infty} \frac{\mathcal{L}^{\log}(\pi, S[0..n-1])}{n}.$$

3. *The* strong log-loss rate *of* π *on* S *is*

$$\mathcal{L}^{\log}_{\mathrm{str}}(\pi, S) = \limsup_{n\to\infty} \frac{\mathcal{L}^{\log}(\pi, S[0..n-1])}{n}.$$

4. *The* (worst-case) log-loss *of* π *on* X *is*

$$\mathcal{L}^{\log}(\pi, X) = \sup_{S\in X} \mathcal{L}^{\log}(\pi, S).$$

5. *The* (worst-case) strong log-loss *of* π *on* X *is*

$$\mathcal{L}^{\log}_{\mathrm{str}}(\pi, X) = \sup_{S\in X} \mathcal{L}^{\log}_{\mathrm{str}}(\pi, S).$$

6. *The* feasible log-loss unpredictability *of* X *is*

$$\mathrm{unpred}^{\log}_{\mathrm{p}}(X) = \inf_{\pi\in\Pi(\mathrm{p})} \mathcal{L}^{\log}(\pi, X).$$

7. *The* feasible strong log-loss unpredictability *of* X *is*

$$\mathrm{Unpred}^{\log}_{\mathrm{p}}(X) = \inf_{\pi\in\Pi(\mathrm{p})} \mathcal{L}^{\log}_{\mathrm{str}}(\pi, X).$$

Hitchcock [14] showed that feasible dimension exactly characterizes feasible log-loss unpredictability; that is,

$$\mathrm{unpred}^{\log}_{\mathrm{p}}(X) = \dim_{\mathrm{p}}(X) \tag{7.1}$$

for all $X \subseteq \mathbf{C}$. The same argument proves the following dual result for strong dimension.

THEOREM 7.1. *For all* $X \subseteq \mathbf{C}$*,* $\mathrm{Unpred}^{\log}_{\mathrm{p}}(X) = \mathrm{Dim}_{\mathrm{p}}(X)$*.*

The following theorem is the main result of this section. Recall that the *polynomial-time many-one degree* of a language $A \subseteq \{0,1\}^*$ is

$$\deg^{\mathrm{P}}_{\mathrm{m}}(A) = \mathrm{P}_{\mathrm{m}}(A) \cap \mathrm{P}^{-1}_{\mathrm{m}}(A),$$

where the *lower span* $\mathrm{P}_{\mathrm{m}}(A)$ and the *upper span* $\mathrm{P}^{-1}_{\mathrm{m}}(A)$ are defined by

$$\mathrm{P}_{\mathrm{m}}(A) = \{B \mid B \leq^{\mathrm{P}}_{\mathrm{m}} A\}, \quad \mathrm{P}^{-1}_{\mathrm{m}}(A) = \{B \mid A \leq^{\mathrm{P}}_{\mathrm{m}} B\}.$$

THEOREM 7.2. *For every pair of* Δ^0_2*-computable real numbers* x, y *with* $0 < x \leq y \leq 1$*, there exists* $A \in \mathrm{E}$ *such that*

$$\dim_{\mathrm{p}}(\deg^{\mathrm{P}}_{\mathrm{m}}(A)) = \dim(\deg^{\mathrm{P}}_{\mathrm{m}}(A)|\mathrm{E}) = x$$

and

$$\mathrm{Dim}_{\mathrm{p}}(\deg^{\mathrm{P}}_{\mathrm{m}}(A)) = \mathrm{Dim}(\deg^{\mathrm{P}}_{\mathrm{m}}(A)|\mathrm{E}) = y.$$

Most of this section is devoted to proving Theorem 7.2. Our proof is motivated by analogous, but simpler, arguments by Ambos-Spies et al. [1]. Like most dimension calculations, our proof consists of separate lower and upper bound arguments. The results from here through Lemma 7.7 are used for the lower bound. Lemma 7.8 uses Theorem 7.1 to establish the upper bound. The proof of Theorem 7.2 follows Lemma 7.8.

The first part of the following theorem is due to Ambos-Spies et al. [1]. The second part is an exact dual of the first part.

THEOREM 7.3. *Let* $A \in \mathrm{E}$*.*

1. $\dim_{\mathrm{p}}(\deg_{\mathrm{m}}^{\mathrm{P}}(A)) = \dim_{\mathrm{p}}(\mathrm{P}_{\mathrm{m}}(A))$ *and* $\dim(\deg_{\mathrm{m}}^{\mathrm{P}}(A)|\mathrm{E}) = \dim(\mathrm{P}_{\mathrm{m}}(A)|\mathrm{E})$.
2. $\mathrm{Dim}_{\mathrm{p}}(\deg_{\mathrm{m}}^{\mathrm{P}}(A)) = \mathrm{Dim}_{\mathrm{p}}(\mathrm{P}_{\mathrm{m}}(A))$ *and* $\mathrm{Dim}(\deg_{\mathrm{m}}^{\mathrm{P}}(A)|\mathrm{E}) = \mathrm{Dim}(\mathrm{P}_{\mathrm{m}}(A)|\mathrm{E})$.

The following lemma is a time-bounded version of Lemma 6.14.

Lemma 7.4. *Assume that* $k, l \in \mathbb{Z}^+$, $\delta > 0$, $\vec{\beta}$ *is an exactly* n^l*-time-computable bias sequence with each* $\beta_i \in \mathbb{Q} \cap [\delta, 1-\delta]$, $s \in \mathbb{Q} \cap [0, \infty)$, *and* d *is an* n^k*-time-computable* s*-gale.*

1. *If* $s < H^-(\vec{\beta})$, *then* $S^\infty[d] \bigcap \mathrm{RAND}^{\vec{\beta}}(n^{k+2l+1}) = \emptyset$.
2. *If* $s < H^+(\vec{\beta})$, *then* $S^\infty_{\mathrm{str}}[d] \bigcap \mathrm{RAND}^{\vec{\beta}}(n^{k+2l+1}) = \emptyset$.

Proof. We proceed exactly as in the proof of Lemma 6.14, noting that our present hypothesis implies that the $\vec{\beta}$-martingale d' is $O(n^{k+2l+1})$-time-computable. □

Our proof of Theorem 7.2 also uses the *martingale dilation technique*, which was introduced by Ambos-Spies, Terwijn, and Zheng [2] and extended by Breutzmann and Lutz [4]. Recall the standard enumeration $s_0, s_1, s_2, \ldots$ of $\{0,1\}^*$, defined in section 2.

Definition. *The* restriction *of a string* $w \in \{0,1\}^*$ *to a language* $A \subseteq \{0,1\}^*$ *is the string* $w \restriction A$ *defined by the following recursion:*

1. $\lambda \restriction A = \lambda$.
2. *For* $w \in \{0,1\}^*$ *and* $b \in \{0,1\}$,

$$(wb) \restriction A = \begin{cases} (w \restriction A)b & \text{if } s_{|w|} \in A, \\ w \restriction A & \text{if } s_{|w|} \notin A. \end{cases}$$

(That is, $w \restriction A$ *is the concatenation of the successive bits* $w[i]$ *for which* $s_i \in A$*.)*

Definition. *A function* $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ *is* strictly increasing *if, for all* $x, y \in \{0,1\}^*$,

$$x < y \Longrightarrow f(x) < f(y),$$

where $<$ *is the standard ordering of* $\{0,1\}^*$.

Notation. If $f : \{0,1\}^* \longrightarrow \{0,1\}^*$, then for each $n \in \mathbb{N}$, let n_f be the unique integer such that $f(s_n) = s_{n_f}$.

Definition. *If* $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ *is strictly increasing and* $\vec{\beta}$ *is a bias sequence, then the* f-dilation *of* $\vec{\beta}$ *is the bias sequence* $\vec{\beta}^f$ *given by* $\beta^f_n = \beta_{n_f}$ *for all* $n \in \mathbb{N}$.

Observation 7.5. *If* $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ *is strictly increasing and* $A \subseteq \{0,1\}^*$, *then for all* $n \in \mathbb{N}$,

$$\chi_{f^{-1}(A)}[0..n-1] = \chi_A[0..n_f - 1] \restriction \mathrm{range}(f).$$

Definition. *If* $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ *is strictly increasing and* d *is a martingale, then the* f-dilation *of* d *is the function* $f\hat{}d : \{0,1\}^* \longrightarrow [0,\infty)$,

$$f\hat{}d(w) = d(w \restriction \mathrm{range}(f)).$$

Intuitively, the f-dilation of d is a strategy for betting on a language A, assuming that d itself is a good betting strategy for betting on the language $f^{-1}(A)$. Given an opportunity to bet on the membership of a string $y = f(x)$ in A, $f\hat{}d$ bets exactly as d would bet on the membership or nonmembership of x in $f^{-1}(A)$.

The following result is a special case of Theorem 6.3 in [4].

Theorem 7.6 (martingale dilation theorem; see Breutzmann and Lutz [4]). *Assume that* $\vec{\beta}$ *is a bias sequence with each* $\beta_i \in (0,1)$, $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ *is strictly* *increasing, and d is a $\vec{\beta}^f$-martingale. Then $f\hat{\ }d$ is a $\vec{\beta}$-martingale and, for every language $A \subseteq \{0,1\}^*$, if d succeeds on $f^{-1}(A)$, then $f\hat{\ }d$ succeeds on A.*

Notation. For each $k \in \mathbb{Z}^+$, define $g_k : \{0,1\}^* \longrightarrow \{0,1\}^*$ by $g_k(x) = 0^{k|x|}1x$. Note that each g_k is strictly increasing and computable in polynomial time.

LEMMA 7.7. *Assume that $\vec{\beta}$ is a bias sequence with each $\beta_i \in (0,1)$, and $R \in \mathrm{RAND}^{\vec{\beta}}(n^2)$. Then, for each $k \geq 2$, $g_k^{-1}(R) \in \mathrm{RAND}^{\vec{\alpha}}(n^k)$, where $\vec{\alpha} = \vec{\beta}^{g_k}$.*

Proof. Let $\vec{\beta}$, k, and $\vec{\alpha}$ be as given, and assume that $g_k^{-1}(R) \notin \mathrm{RAND}^{\vec{\alpha}}(n^k)$. Then there is an n^k-time-computable $\vec{\alpha}$-martingale d that succeeds on $g_k^{-1}(R)$. It follows by Theorem 7.6 that $g_k\hat{\ }d$ is a $\vec{\beta}$-martingale that succeeds on R. The time required to compute $g_k\hat{\ }d(w)$ is $O(|w|^2 + |w'|^k)$ steps, where $w' = w \upharpoonright \mathrm{range}(g_k)$. (This allows $O(|w|^2)$ steps to compute w' and then $O(|w'|^k)$ steps to compute $d(w')$.) Now $|w'|$ is bounded above by the number of strings x such that $k|x| + |x| + 1 \leq |s_{|w|}| = \lfloor \log(1 + |w|) \rfloor$. Therefore $|w'| \leq 2^{1+\log(1+|w|)/k}$, which implies $|w'|^k = O(|w|)$. Thus $g_k\hat{\ }d(w)$ is an $O(n^2)$-time computable $\vec{\beta}$-martingale, so $R \notin \mathrm{RAND}^{\vec{\beta}}(n^2)$. □

Notation. From here through the proof of Theorem 7.2, we assume that α and β are Δ_2^0-computable real numbers with $0 < \alpha \leq \beta \leq 1/2$. It is well known that a real number is Δ_2^0-computable if and only if there is a computable sequence of rationals that converge to it. Slowing down this construction gives polynomial-time functions $\hat{\alpha}, \hat{\beta} : \mathbb{N} \to \mathbb{Q}$ such that

$$\lim_{n\to\infty} \hat{\alpha}(n) = \alpha, \qquad \lim_{n\to\infty} \hat{\beta}(n) = \beta.$$

We assume that for some $\delta > 0$, $\hat{\alpha}(n) > \delta$ for all n. We place the analogous assumption on $\hat{\beta}(n)$. For each n, we let

$$\kappa(n) = \begin{cases} \hat{\alpha}(n) & \text{if } n \text{ is even,} \\ \hat{\beta}(n) & \text{if } n \text{ is odd,} \end{cases}$$

and define a special-purpose bias sequence $\vec{\gamma}$ by

$$\gamma_n = \kappa(\log^* n).$$

Note that $\vec{\gamma}$ is $O(n)$-time-computable, $H^-(\vec{\gamma}) = \mathcal{H}(\alpha)$, and $H^+(\vec{\gamma}) = \mathcal{H}(\beta)$.

We now use the unpredictability characterizations from the beginning of this section to establish upper bounds on the dimensions and strong dimensions of lower spans of sequences random relative to $\vec{\gamma}$.

LEMMA 7.8. *For each $R \in \mathrm{RAND}^{\vec{\gamma}}(n^5)$,*

$$\dim_{\mathrm{p}}(\mathrm{P_m}(R)) \leq \mathcal{H}(\alpha)$$

and

$$\mathrm{Dim}_{\mathrm{p}}(\mathrm{P_m}(R)) \leq \mathcal{H}(\beta).$$

Proof. Fix a polynomial-time function $f : \{0,1\}^* \to \{0,1\}^*$. The *collision set* of f is

$$C_f = \{j \mid (\exists i < j) f(s_i) = f(s_j)\}.$$

For each $n \in \mathbb{N}$, let

$$\#C_f(n) = |C_f \cap \{0, \ldots, n-1\}|.$$

We use f to define the predictors

$$\pi_0^f(w,b) = \begin{cases} \frac{1}{2} & \text{if } |w| \notin C_f, \\ w[i] & \text{if } |w| \in C_f,\ i = \min\{j \mid f(s_j) = f(s_{|w|})\}, \text{ and } b = 1\ , \\ 1 - w[i] & \text{if } |w| \in C_f,\ i = \min\{j \mid f(s_j) = f(s_{|w|})\}, \text{ and } b = 0\ , \end{cases}$$

and

$$\pi_1^f(w,b) = \begin{cases} \gamma_{|w|}^f & \text{if } |w| \notin C_f \text{ and } b = 1, \\ 1 - \gamma_{|w|}^f & \text{if } |w| \notin C_f \text{ and } b = 0, \\ w[i] & \text{if } |w| \in C_f,\ i = \min\{j \mid f(s_j) = f(s_{|w|})\}, \text{ and } b = 1\ , \\ 1 - w[i] & \text{if } |w| \in C_f,\ i = \min\{j \mid f(s_j) = f(s_{|w|})\}, \text{ and } b = 0\ , \end{cases}$$

for all $w \in \{0,1\}^*$ and $b \in \{0,1\}$.

For each $S \in \mathbf{C}$, we now define several objects to facilitate the proof. First, we let

$$A^f(S) = f^{-1}(S);$$

that is, $A^f(S)$ is the language $\leq_{\mathrm{m}}^{\mathrm{P}}$-reduced to S by f. Observe that for all $w \sqsubseteq A^f(S)$,

$$\mathcal{L}^{\log}(\pi_0^f, w) = |w| - \#C_f(|w|). \tag{7.2}$$

Recall the sequence of towers defined by t_j by $t_0 = 1$ and $t_{j+1} = 2^{t_j}$. For any $j \in \mathbb{N}$ and $t_j < n \leq t_{j+1}$, define the entropy quantity

$$H_n^f = \sum_{\substack{i<n \\ i \notin C_f \text{ and } i_f > t_{j-1}}} \mathcal{H}(\gamma_n^f)$$

and the random variable

$$L_n^f(S) = \sum_{\substack{i<n \\ i \notin C_f \text{ and } i_f > t_{j-1}}} \log \frac{1}{\pi_1^f(A^f(S)[0..i-1], A^f(S)[i])}.$$

(Recall that i_f is the unique number such that $f(s_i) = s_{i_f}$.) We have

$$\begin{aligned} \mathcal{L}^{\log}(\pi_1^f, A^f(S)[0..n-1]) &= \sum_{i<n} \log \frac{1}{\pi_1^f(A^f(S)[0..i-1], A^f(S)[i])} \\ &= \sum_{\substack{i<n \\ i \notin C_f}} \log \frac{1}{\pi_1^f(A^f(S)[0..i-1], A^f(S)[i])} \\ &= L_n^f(S) + \sum_{\substack{i<n \\ i \notin C_f \text{ and } i_f \leq t_{j-1}}} \log \frac{1}{\pi_1^f(A^f(S)[0..i-1], A^f(S)[i])} \\ &\leq L_n^f(S) + (1 + t_{j-1}) \log \tfrac{1}{\delta}, \end{aligned}$$

so the bound

$$\mathcal{L}^{\log}(\pi_1^f, A^f(S)[0..n-1]) \leq L_n^f(S) + (1 + \log n) \log \tfrac{1}{\delta} \tag{7.3}$$

holds for all n. Finally, for any $\epsilon > 0$ and $\theta \in (0, 1)$, define the set

$$J^f_{\theta,\epsilon}(S) = \{n \mid \#C_f(n) < (1-\theta)n \text{ and } L^f_n(S) \geq H^f_n + \epsilon n\}$$

of natural numbers.

Claim. *For any rational $\theta \in (0,1)$ and $\epsilon > 0$,*

$$\mu^{\vec{\gamma}}_{n^5}\left(\{S \mid J^f_{\theta,\epsilon}(S) \textit{ is finite}\}\right) = 1.$$

Proof of claim. The argument is similar to the proof of Lemma 6.12. For each $n \in \mathbb{N}$, define the set

$$Y_n = \begin{cases} \{S \mid L^f_n(S) \geq H^f_n + \epsilon n\} & \text{if } \#C_f(n) < (1-\theta)n, \\ \emptyset & \text{otherwise,} \end{cases}$$

and let

$$X_\epsilon = \{S \in \mathbf{C} | (\exists^\infty n) S \in Y_n\}.$$

To prove the claim, we will show that $\mu^{\vec{\gamma}}_{n^5}(X_\epsilon) = 0$.

For each $n \in \mathbb{N}$ and $w \in \{0,1\}^*$, let

$$d_n(w) = \begin{cases} \mu^{\vec{\gamma}}(Y_n | \mathbf{C}_w) & \text{if } |w| \leq n, \\ d_n(w[0..n-1]) & \text{if } |w| > n. \end{cases}$$

It is clear that each d_n is a $\vec{\gamma}$-martingale and that $Y_n \subseteq S^1[d_n]$ for all $n \in \mathbb{N}$.

Let $S \in \mathbf{C}$. For each $n, j \in \mathbb{N}$, let

$$I^n_j = \{i_f \mid i < n, i \notin C_f, \text{ and } \log^* i_f = j\}.$$

Also, define $S^+ = \{i \mid S[i] = 1\}$ and $S^- = \{i \mid S[i] = 0\}$. Then, if n is large enough to ensure that $\log^* i_f \leq 1 + \log^* n$ for all $i < n$, we have

$$L^f_n(S) = \sum_{k=(\log^* n)-1}^{(\log^* n)+1} \left|I^n_k \cap S^+\right| \log \frac{1}{\kappa(k)} + \left|I^n_k \cap S^-\right| \log \frac{1}{1-\kappa(k)}.$$

For any n and k, write $i(n,k) = |I^n_k|$. Let T_n be the set of all tuples (l_{-1}, l_0, l_1) satisfying $0 \leq l_r \leq i(n, j+r)$ for $-1 \leq r \leq 1$ and

$$\sum_{r=-1}^{1} l_r \log \frac{1}{\kappa(j+r)} + (i(n,j+r) - l_r) \log \frac{1}{1-\kappa(j+r)} \geq H^f_n + \epsilon n,$$

where $j = \log^* n$. Then we have

$$\mu^{\vec{\gamma}}(Y_n) = \sum_{(l_{-1},l_0,l_1) \in T_n} \prod_{r=-1}^{1} \binom{i(n,j+r)}{l_r} \kappa(j+r)^{l_r} (1-\kappa(j+r))^{i(n,j+r)-l_r}.$$

We can write a similar formula for $\mu^{\vec{\gamma}}(Y_n|\mathbf{C}_w)$ when $w \neq \lambda$. From this it follows that the mapping $(n,w) \mapsto d_n(w)$ is exactly computable in $O(n^3)$ time.

Since $\hat{\gamma}$ is bounded away from 0 and 1, we can apply Theorem 6.9 to conclude that there exists $\rho \in (0,1)$ such that for all $n \in \mathbb{N}$ with $Y_n \neq \emptyset$ we have

$$\mu^{\vec{\gamma}}(Y_n) < 2\rho^{n-\#C_f(n)} < 2\rho^{\theta n}.$$

It follows that the series $\sum_{n=0}^{\infty} d_n(\lambda)$ is p-convergent, so the polynomial-time first Borel–Cantelli lemma [20] (extended to $\vec{\gamma}$ as indicated in [4]) tells us that $\mu^{\vec{\gamma}}_{n^5}(X_\epsilon) = 0$. □

Let $R \in \mathrm{RAND}^{\vec{\gamma}}(n^5)$. Let $\epsilon > 0$ and $\theta < \mathcal{H}(\alpha)+\epsilon$ be rational. Then by the above claim, $J^f_{\theta,\epsilon}(R)$ is finite. That is, for all but finitely many n,

$$\#C_f(n) \geq (1-\theta)n \quad \text{or} \quad L^f_n(R) < H^f_n + \epsilon n. \tag{7.4}$$

Writing $w_n = A^f(R)[0..n-1]$, (7.4) combined with (7.2) and (7.3) implies that for all but finitely many n,

$$\mathcal{L}^{\log}(\pi^f_0, w_n) \leq \theta n < (\mathcal{H}(\alpha)+\epsilon)n \tag{7.5}$$

or

$$\mathcal{L}^{\log}(\pi^f_1, w_n) < H^f_n + \epsilon n + (1+\log n)\log\tfrac{1}{\delta}. \tag{7.6}$$

As

$$\limsup_{n\to\infty} \frac{H^f_n}{n} \leq \mathcal{H}(\beta),$$

it follows that

$$\limsup_{n\to\infty} \frac{\min\{\mathcal{L}^{\log}(\pi^f_0, w_n), \mathcal{L}^{\log}(\pi^f_1, w_n)\}}{n} \leq \mathcal{H}(\beta)+\epsilon. \tag{7.7}$$

If (7.5) holds for infinitely many n, then

$$\mathcal{L}^{\log}(\pi^f_0, A^f(R)) \leq \mathcal{H}(\alpha)+\epsilon. \tag{7.8}$$

Otherwise, (7.6) holds for almost all n. Assuming

$$\liminf_{n\to\infty} \frac{H^f_n}{n} \leq \mathcal{H}(\alpha), \tag{7.9}$$

in this case we have

$$\mathcal{L}^{\log}(\pi^f_1, A^f(R)) \leq \mathcal{H}(\alpha)+\epsilon. \tag{7.10}$$

We now verify (7.9). For each n, let $m(n) = t_n^2$. Then for sufficiently large n, we have $i_f < t_{n+1}$ for all $i < m(n)$. Using the sets I^k_n from the proof of the claim, we then have

$$\begin{aligned} H^f_{m(n)} &= \left|I^{m(n)}_n\right| \mathcal{H}(\kappa(n)) + \left|I^{m(n)}_{n+1}\right| \mathcal{H}(\kappa(n+1)) \\ &\leq (t_n+1)\mathcal{H}(\kappa(n)) + m(n)\mathcal{H}(\kappa(n+1)). \end{aligned}$$

As $t_n = o(m(n))$ and $\kappa(2n) \to \alpha$ as $n \to \infty$, we have

$$\liminf_{n\to\infty} \frac{H^f_n}{n} \leq \liminf_{n\to\infty} \frac{H^f_{m(2n+1)}}{m(2n+1)} \leq \mathcal{H}(\alpha).$$

For each polynomial-time reduction f, we have defined and analyzed two predictors π_0^f and π_1^f. We now show how to combine all these predictors into a single predictor that will establish the lemma.

Let $\{f_j \mid j \in \mathbb{N}\}$ be a uniform enumeration of all polynomial-time functions $f_j : \{0,1\}^* \to \{0,1\}^*$ such that $f_j(x)$ is computable in $O(2^{|x|} + j)$ steps. For any predictor ρ, define a probability measure $\mu[\rho]$ by

$$\mu[\rho](w) = \prod_{i=0}^{|w|-1} \rho(w[0..i-1], w[i])$$

for all $w \in \{0,1\}^*$. For each $m \in \mathbb{N}$ and $w \in \{0,1\}^m$, let

$$\mu_m(w) = 2^{-(2m+1)} + \sum_{j=0}^{m-1} 2^{-(2j+3)} \left(\mu[\pi_0^{f_j}](w) + \frac{1}{2}\mu[\pi_1^{f_j}](w) \right).$$

Then

$$\begin{aligned}
\mu_{m+1}(w0) + \mu_{m+1}(w1) &= 2^{-(2m+3)} + \sum_{j=0}^{m} 2^{-(2j+3)} \left(\mu[\pi_0^{f_j}](w0) + \frac{1}{2}\mu[\pi_1^{f_j}](w0) \right) \\
&\quad + 2^{-(2m+3)} + \sum_{j=0}^{m} 2^{-(2j+3)} \left(\mu[\pi_0^{f_j}](w1) + \frac{1}{2}\mu[\pi_1^{f_j}](w1) \right) \\
&= 2^{-(2m+2)} + \sum_{j=0}^{m} 2^{-(2j+3)} \left(\mu[\pi_0^{f_j}](w) + \frac{1}{2}\mu[\pi_1^{f_j}](w) \right) \\
&= 2^{-(2m+3)} \left(2 + \mu[\pi_0^{f_m}](w) + \frac{1}{2}\mu[\pi_1^{f_m}](w) \right) \\
&\quad + \mu_m(w) - 2^{-(2m+1)} \\
&\le \mu_m(w) + 2^{-(2m+3)} \left(3 + \frac{1}{2} \right) - 2^{-(2m+1)} \\
&< \mu_m(w).
\end{aligned}$$

Now define a predictor π by

$$\begin{aligned}
\pi(w,1) &= \frac{\mu_{|w|+1}(w1)}{\mu_{|w|}(w)}, \\
\pi(w,0) &= 1 - \pi(w,1).
\end{aligned}$$

Then for all $w \in \{0,1\}^*$ and $b \in \{0,1\}$,

$$\pi(w,b) \ge \frac{\mu_{|w|+1}(wb)}{\mu_{|w|}(w)}.$$

For all $w \in \{0,1\}^*$, $k \in \{0,1\}$, and $j < |w|$, we have

$$\begin{aligned}\mathcal{L}^{\log}(\pi, w) &= \sum_{i=0}^{|w|-1} \log \frac{1}{\pi(w[0..i-1], w[i])} \\ &\le \sum_{i=0}^{|w|-1} \log \frac{\mu_i(w[0..i-1])}{\mu_{i+1}(w[0..i])} \\ &= \log \frac{\mu_0(\lambda)}{\mu_{|w|}(w)} \\ &\le \log \frac{2^{2j+3+k}}{\mu[\pi_k^{f_j}](w)} \\ &= 2j + 3 + k + \mathcal{L}^{\log}(\pi_k^{f_j}, w).\end{aligned}$$

For any $j \in \mathbb{N}$, it follows that

$$\mathcal{L}^{\log}_{\mathrm{str}}(\pi, A^{f_j}(R)) \le \mathcal{H}(\beta) + \epsilon$$

by using $f = f_j$ in (7.7). Also, since either (7.8) or (7.10) holds for $f = f_j$, we have

$$\mathcal{L}^{\log}(\pi, A^{f_j}(R)) \le \mathcal{H}(\alpha) + \epsilon.$$

As π is (exactly) polynomial-time computable, this establishes that

$$\mathrm{P_m}(R) = \{A^{f_j}(R) \mid j \in \mathbb{N}\}$$

has p-dimension at most $\mathcal{H}(\alpha) + \epsilon$ by (7.1) and strong p-dimension at most $\mathcal{H}(\beta) + \epsilon$ by Theorem 7.1. As $\epsilon > 0$ was arbitrary, the statement of the lemma follows. □

We now have the machinery we need to prove the main result of this section.

Proof of Theorem 7.2. Let x and y be Δ^0_2-computable real numbers with $0 < x \le y \le 1$. Then there exist Δ^0_2-computable real numbers α and β with $0 < \alpha \le \beta \le \frac{1}{2}$, $\mathcal{H}(\alpha) = x$, and $\mathcal{H}(\beta) = y$. Let $\vec{\gamma}$ be the bias sequence defined from α and β above (just prior to Lemma 7.8). It is well known [20, 2] that almost every language in E is n^5-$\vec{\gamma}$-random. In particular, there exists a language $A \in \mathrm{RAND}^{\vec{\gamma}}(n^5) \cap \mathrm{E}$. By Theorem 7.3, it suffices to prove that

$$\dim_{\mathrm{p}}(\mathrm{P_m}(A)) = \dim(\mathrm{P_m}(A)|\mathrm{E}) = \mathcal{H}(\alpha)$$

and

$$\mathrm{Dim}_{\mathrm{p}}(\mathrm{P_m}(A)) = \mathrm{Dim}(\mathrm{P_m}(A)|\mathrm{E}) = \mathcal{H}(\beta).$$

By Lemma 7.8, then, it suffices to prove that

$$\dim(\mathrm{P_m}(A)|\mathrm{E}) \ge \mathcal{H}(\alpha) \tag{7.11}$$

and

$$\mathrm{Dim}(\mathrm{P_m}(A)|\mathrm{E}) \ge \mathcal{H}(\beta). \tag{7.12}$$

Let $s \in [0, \mathcal{H}(\alpha)) \cap \mathbb{Q}$, and let d^- be an n^k-time computable s-gale. Similarly, let $t \in [0, \mathcal{H}(\beta)) \cap \mathbb{Q}$, and let d^+ be an n^k-time computable t-gale. It suffices to show that

$$\mathrm{P_m}(A) \cap \mathrm{E} \not\subseteq S^\infty[d^-] \tag{7.13}$$

and

$$\mathrm{P_m}(A) \cap \mathrm{E} \not\subseteq S^{\infty}_{\mathrm{str}}[d^+]. \tag{7.14}$$

Let $B = g^{-1}_{k+3}(A)$. It is clear that $B \in \mathrm{P_m}(A) \cap \mathrm{E}$. Also, by Lemma 7.7, we have $B \in \mathrm{RAND}^{\vec{\gamma}'}(n^k)$, where $\vec{\gamma}' = \vec{\gamma}^{g_{k+3}}$. Since

$$s < \mathcal{H}(\alpha) = H^-(\vec{\gamma}) = H^-(\vec{\gamma}'),$$

$$t < \mathcal{H}(\beta) = H^+(\vec{\gamma}) = H^+(\vec{\gamma}'),$$

and $\vec{\gamma}'$ is $O(n)$-time-computable, Lemma 7.4 tells us that $B \notin S^{\infty}[d^-]$ and $B \notin S^{\infty}_{\mathrm{str}}[d^+]$. Thus (7.13) and (7.14) hold. □

In light of Theorem 7.2, the following question concerning the relativized feasible dimension of NP is natural.

Open Question 7.9. *For which pairs of real numbers* $\alpha, \beta \in [0,1]$ *does there exist an oracle* A *such that* $\dim_{\mathrm{p}^A}(\mathrm{NP}^A) = \alpha$ *and* $\mathrm{Dim}_{\mathrm{p}^A}(\mathrm{NP}^A) = \beta$*?*

We conclude this section with two brief observations.

Fortnow and Lutz [11] have recently established a tight quantitative relationship between p-dimension and feasible predictability. Specifically, for each $X \subseteq \mathbf{C}$, they investigated the quantity $\mathrm{Pred_p}(X)$, which is the supremum, for all feasible predictors π, of the *(worst-case, upper) success rate*

$$\pi^+(S) = \inf_{S \in X} \limsup_{n \to \infty} \pi^+(S[0..n-1]), \tag{7.15}$$

where

$$\pi^+(w) = \frac{1}{|w|} \sum_{i=0}^{|w|-1} \pi(w[0..i-1], w[i])$$

is the expected fraction of correct predictions that π will make on w. They proved that $\mathrm{Pred_p}(X)$ is related to the p-dimension of X by

$$2(1 - \mathrm{Pred_p}(X)) \le \dim_{\mathrm{p}}(X) \le \mathcal{H}(\mathrm{Pred_p}(X)) \tag{7.16}$$

(where $\mathcal{H}(\alpha)$ is the Shannon entropy of α) and that these bounds are tight. If we call $\mathrm{Pred_p}(X)$ the *upper feasible predictability* of X and define the *lower feasible predictability* of X, $\mathrm{pred_p}(X)$, in the same fashion, but with the limit superior in (7.15) replaced by a limit inferior, then we have the following dual of (7.16).

Theorem 7.10. *For all* $X \subseteq \mathbf{C}$*,*

$$2(1 - \mathrm{pred_p}(X)) \le \mathrm{Dim_p}(X) \le \mathcal{H}(\mathrm{pred_p}(X)).$$

For each $s : \mathbb{N} \to \mathbb{N}$, let $\mathrm{SIZE}(s(n))$ be the class of all (characteristic sequences of) languages $A \subseteq \{0,1\}^*$ such that, for each $n \in \mathbb{N}$, $A_{=n}$ is decided by a Boolean circuit consisting of at most $s(n)$ gates.

Theorem 7.11. *For each* $\alpha \in [0,1]$*, the class* $X_\alpha = \mathrm{SIZE}(\alpha \cdot \frac{2^n}{n})$ *is pspace-regular, with* $\dim_{\mathrm{pspace}}(X_\alpha) = \mathrm{Dim}_{\mathrm{pspace}}(X_\alpha) = \dim(X_\alpha|\mathrm{ESPACE}) = \mathrm{Dim}(X_\alpha|\mathrm{ESPACE}) = \alpha$*.*

Proof. It was shown in [23] that $\dim_{\mathrm{pspace}}(X_\alpha) = \dim(X_\alpha|\mathrm{ESPACE}) = \alpha$. This proof also shows that the strong dimensions are α. □

Acknowledgments. The third author thanks Dan Mauldin for extremely useful discussions. We all thank two very thorough referees for many helpful suggestions. We also thank Dave Doty for pointing out a problem with an earlier version of our proof of Theorem 6.18.

REFERENCES

[1] K. Ambos-Spies, W. Merkle, J. Reimann, and F. Stephan, *Hausdorff dimension in exponential time*, in Proceedings of the 16th IEEE Conference on Computational Complexity, Chicago, IL, IEEE Computer Society Press, Piscataway, NJ, 2001, pp. 210–217.

[2] K. Ambos-Spies, S. A. Terwijn, and X. Zheng, *Resource bounded randomness and weakly complete problems*, Theoret. Comput. Sci., 172 (1997), pp. 195–207.

[3] P. Billingsley, *Probability and Measure*, 3rd ed., John Wiley & Sons, New York, 1995.

[4] J. M. Breutzmann and J. H. Lutz, *Equivalence of measures of complexity classes*, SIAM J. Comput., 29 (1999), pp. 302–326.

[5] J. Cai and J. Hartmanis, *On Hausdorff and topological dimensions of the Kolmogorov complexity of the real line*, J. Comput. Systems Sci., 49 (1994), pp. 605–619.

[6] G. J. Chaitin, *A theory of program size formally identical to information theory*, J. ACM, 22 (1975), pp. 329–340.

[7] J. J. Dai, J. I. Lathrop, J. H. Lutz, and E. Mayordomo, *Finite-state dimension*, Theoret. Comput. Sci., 310 (2004), pp. 1–33.

[8] K. Falconer, *Techniques in Fractal Geometry*, John Wiley & Sons, New York, 1997.

[9] K. Falconer, *Fractal Geometry: Mathematical Foundations and Applications*, 2nd ed., John Wiley & Sons, New York, 2003.

[10] S. A. Fenner, *Gales and Supergales are Equivalent for Defining Constructive Hausdorff Dimension*, Technical Report cs.CC/0208044, Computing Research Repository, Cornell University, Ithaca, NY, 2002.

[11] L. Fortnow and J. H. Lutz, *Prediction and dimension*, J. Comput. System Sci., 70 (2005), pp. 570–589.

[12] F. Hausdorff, *Dimension und äußeres Maß*, Math. Ann., 79 (1919), pp. 157–179.

[13] J. M. Hitchcock, *MAX3SAT is exponentially hard to approximate if NP has positive dimension*, Theoret. Comput. Sci., 289 (2002), pp. 861–869.

[14] J. M. Hitchcock, *Fractal dimension and logarithmic loss unpredictability*, Theoret. Comput. Sci., 304 (2003), pp. 431–441.

[15] J. M. Hitchcock, *Gales suffice for constructive dimension*, Inform. Process. Lett., 86 (2003), pp. 9–12.

[16] J. M. Hitchcock, *Correspondence principles for effective dimensions*, Theory Comput. Systems, 38 (2005), pp. 559–571.

[17] S. Kakutani, *On the equivalence of infinite product measures*, Ann. Math., 49 (1948), pp. 214–224.

[18] P. Lévy, *Théorie de l'Addition des Variables Aleatoires*, 2nd ed., Gauthier-Villars, Paris, 1954.

[19] M. Li and P. M. B. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications*, 2nd ed., Springer-Verlag, Berlin, 1997.

[20] J. H. Lutz, *Almost everywhere high nonuniform complexity*, J. Comput. System Sci., 44 (1992), pp. 220–258.

[21] J. H. Lutz, *Resource-bounded measure*, in Proceedings of the 13th IEEE Conference on Computational Complexity, Buffalo, NY, 1998, IEEE Computer Society Press, Piscataway, NJ, 1998, pp. 236–248.

[22] J. Lutz, *Dimension in complexity classes*, in Proceedings of the 15th IEEE Conference on Computational Complexity, Florence, Italy, 2000, IEEE Computer Society Press, Piscataway, NJ, 2000, pp. 158–169.

[23] J. H. Lutz, *Dimension in complexity classes*, SIAM J. Comput., 32 (2003), pp. 1236–1259.

[24] J. H. Lutz, *The dimensions of individual strings and sequences*, Inform. Comput., 187 (2003), pp. 49–79.

[25] P. Martin-Löf, *The definition of random sequences*, Information and Control, 9 (1966), pp. 602–619.

[26] P. Mattila and R. Mauldin, *Measure and dimension functions: Measurability and densities*, Math. Proc. Cambridge Philos. Soc., 121 (1997), pp. 81–100.

[27] E. Mayordomo, *A Kolmogorov complexity characterization of constructive Hausdorff dimension*, Inform. Process. Lett., 84 (2002), pp. 1–3.

[28] E. Mayordomo, *Effective Hausdorff dimension*, in Classical and New Paradigms of Computation and Their Complexity Hierarchies, Trends in Logic 23, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2004, pp. 171–186.

[29] B. Ya. Ryabko, *Coding of combinatorial sources and Hausdorff dimension*, Soviet Math. Dokl., 30 (1984), pp. 219–222.

[30] B. Ya. Ryabko, *Noiseless coding of combinatorial sources*, Probl. Inform. Transm., 22 (1986), pp. 170–179.

[31] B. Ya. Ryabko, *Algorithmic approach to the prediction problem*, Probl. Inform. Transm., 29 (1993), pp. 186–193.

[32] B. Ya. Ryabko, *The complexity and effectiveness of prediction problems*, J. Complexity, 10 (1994), pp. 281–295.

[33] C. P. Schnorr, *A unified approach to the definition of random sequences*, Math. Systems Theory, 5 (1971), pp. 246–258.

[34] C. P. Schnorr, *Zufälligkeit und Wahrscheinlichkeit*, Lecture Notes in Math. 218, Springer-Verlag, New York, 1971.

[35] C. P. Schnorr, *Process complexity and effective random tests*, J. Comput. System Sci., 7 (1973), pp. 376–388.

[36] C. P. Schnorr, *A survey of the theory of random sequences*, in Basic Problems in Methodology and Linguistics, R. E. Butts and J. Hintikka, eds., D. Reidel, Boston, 1977, pp. 193–211.

[37] C. E. Shannon, *A mathematical theory of communication*, Bell System Tech. J., 27 (1948), pp. 379–423, 623–656.

[38] L. Staiger, *Kolmogorov complexity and Hausdorff dimension*, Inform. Comput., 103 (1993), pp. 159–194.

[39] L. Staiger, *A tight upper bound on Kolmogorov complexity and uniformly optimal prediction*, Theory Comput. Syst., 31 (1998), pp. 215–229.

[40] L. Staiger, *How much can you win when your adversary is handicapped?*, in Numbers, Information and Complexity, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000, pp. 403–412.

[41] D. Sullivan, *Entropy, Hausdorff measures old and new, and limit sets of geometrically finite Kleinian groups*, Acta Math., 153 (1984), pp. 259–277.

[42] K. Tadaki, *A generalization of Chaitin's halting probability* Ω *and halting self-similar sets*, Hokkaido Math. J., 31 (2002), pp. 219–253.

[43] C. Tricot, *Two definitions of fractional dimension*, Math. Proc. Cambridge Philos. Soc., 91 (1982), pp. 57–74.

[44] M. van Lambalgen, *Random Sequences*, Ph.D. thesis, Department of Mathematics, University of Amsterdam, Amsterdam, 1987.

[45] M. van Lambalgen, *Von Mises' definition of random sequences reconsidered*, J. Symbolic Logic, 52 (1987), pp. 725–755.

[46] J. Ville, *Étude Critique de la Notion de Collectif*, Gauthier-Villars, Paris, 1939.

[47] V. G. Vovk, *On a randomness criterion*, Soviet Math. Dokl., 35 (1987), pp. 656–660.

[48] J. Ziv, *Coding theorems for individual sequences*, IEEE Trans. Inform. Theory, 24 (1978), pp. 405–412.

[49] J. Ziv and A. Lempel, *Compression of individual sequences via variable rate coding*, IEEE Trans. Inform. Theory, 24 (1978), pp. 530–536.

SIAM J. COMPUT.
Vol. 37, No. 3, pp. 706–721

NEW APPROACHES FOR VIRTUAL PRIVATE NETWORK DESIGN*

FRIEDRICH EISENBRAND†, FABRIZIO GRANDONI‡, GIANPAOLO ORIOLO§, AND MARTIN SKUTELLA¶

Abstract. *Virtual private network design* is the following NP-hard problem. We are given a communication network represented as a weighted graph with thresholds on the nodes which represent the amount of flow that a node can send to and receive from the network. The task is to reserve capacities at minimum cost and to specify paths between every ordered pair of nodes such that all valid traffic-matrices can be routed along the corresponding paths. Recently, this network design problem has received considerable attention in the literature. It is motivated by the fact that the exact amount of flow which is exchanged between terminals is not known in advance and prediction is often elusive. The main contributions of this paper are as follows: (1) Using Hu's 2-commodity flow theorem, we provide a new and considerably stronger lower bound on the cost of an optimum solution. With this lower bound we reanalyze a simple routing scheme which has been described in the literature many times, and provide an improved upper bound on its approximation ratio. (2) We present a new randomized approximation algorithm. In contrast to earlier approaches from the literature, the resulting solution does not have tree structure. A combination of our new algorithm with the simple routing scheme yields an expected performance ratio of 3.79 for virtual private network design. This is a considerable improvement of the previously best known 5.55-approximation result [A. Gupta, A. Kumar, and T. Roughgarden, *Simpler and better approximation algorithms for network design*, in Proceedings of the ACM Symposium on Theory of Computing, ACM, New York, 2003, pp. 365–372]. (3) Our VPND algorithm uses a Steiner tree approximation algorithm as a subroutine. It is known that an optimum Steiner tree can be computed in polynomial time if the number of terminals is logarithmic. Replacing the approximate Steiner tree computation with an exact one whenever the number of terminals is sufficiently small, we finally reduce the approximation ratio to 3.55. To the best of our knowledge, this is the first time that a nontrivial result from exact (exponential) algorithms leads to an improved polynomial-time approximation algorithm.

Key words. approximation algorithms, randomized algorithms, network design

AMS subject classifications. 68W20, 68W25

DOI. 10.1137/060654827

1. Introduction.

Consider a communication network which is represented by an undirected graph $G = (V, E)$ with edge-weights $c : E \to \mathbb{R}_+$. Within this network there is a set of terminals $T \subseteq V$ which want to communicate with each other. However, the exact amount of traffic between pairs of terminals is not known in advance. Instead, each terminal $v \in T$ has associated input and output thresholds $b_{in}(v) \in \mathbb{Z}_{\geq 0}$ and $b_{out}(v) \in \mathbb{Z}_{\geq 0}$. A *traffic-matrix* $D \in \mathbb{Q}_{\geq 0}^{TT}$ is *valid* if it respects the lower and upper bounds on the incoming and outgoing traffic of the terminals, i.e., if

*Received by the editors March 22, 2006; accepted for publication (in revised form) February 2, 2007; published electronically June 29, 2007. Preliminary versions of these results appeared in the Proceedings of SODA'05 [7] and of ICALP'05 [8].

http://www.siam.org/journals/sicomp/37-3/65482.html

†Fachbereich Informatik, Universität Dortmund, 44221 Dortmund, Germany (friedrich.eisenbrand@cs.uni-dortmund.de).

‡Dipartimento di Informatica, Università di Roma "La Sapienza," Via Salaria 113, 00198 Roma, Italy (grandoni@di.uniroma1.it).

§Dipartimento di Ingegneria dell'Impresa, Università di Roma "Tor Vergata," Via del Politecnico 1, 00165, Roma, Italy (oriolo@disp.uniroma2.it).

¶Fachbereich Mathematik, Universität Dortmund, 44221 Dortmund, Germany (martin.skutella@uni-dortmund.de).

the following holds for each terminal $i \in T$:

$$\sum_{j \in T, j \neq i} D(i,j) \leq b_{out}(i) \quad \text{and} \quad \sum_{j \in T, j \neq i} D(j,i) \leq b_{in}(i).$$

The *(asymmetric) virtual private network design (VPND)* problem defined by G, c, and T consists of finding capacities $u(e)$, $e \in E$ and paths P_{ij} for each ordered pair $(i,j) \in TT$ such that the following conditions hold:

(i) All valid traffic-matrices can be routed without exceeding the installed capacities where all traffic from terminal i to terminal j is routed along path P_{ij}.

(ii) The total cost of the reservation $\sum_{e \in E} u(e)\, c(e)$ is minimal.

A reservation of capacities $u : E \to \mathbb{R}_+$ is a *tree reservation* if the subgraph of G induced by the edges $e \in E$ with $u(e) > 0$ is a tree. A general reservation is sometimes referred to as a *graph reservation*.

The virtual private network design problem is NP-hard by the following reduction from the Steiner tree problem [11]. Given an instance of the Steiner tree problem, pick a terminal which has to be connected with the other terminals in a Steiner tree. This terminal has thresholds $b_{in}(v) = 0$ and $b_{out}(v) = 1$. All other terminals u of the Steiner tree instance have $b_{in}(u) = 1$ and $b_{out}(u) = 0$. A minimum cost Steiner tree also yields an optimum reservation for this VPND instance.

The virtual private network design problem was independently defined by Fingerhut, Suri, and Turner [10] and by Duffield et al. [6] and has since then been studied by various authors in several variations which we next discuss. In the following list, the last one ($AsymG$) is the one which we refer to as VPND.

($SymT$) Symmetric thresholds, tree reservation: In this variant, each terminal $i \in T$ has only one threshold $b(i)$, which is an upper bound on the cumulative amount of traffic that terminal i can send or receive. The task is to find an optimal tree reservation which supports all valid traffic-matrices. Gupta et al. [11] show that ($SymT$) is polynomially solvable.

($SymG$) Symmetric thresholds, graph reservation: This variant is defined in the same way as ($SymT$), except that the capacity reservation can be arbitrary and not necessarily a tree. Gupta et al. [11] present a 2-approximation for ($SymG$). It is not known whether $SymG$ is NP-hard.

($BalT$) Balanced thresholds, tree reservation: The thresholds are *balanced*, which means that $\sum_{v \in T} b_{in}(v) = \sum_{v \in T} b_{out}(v)$. The reservation has to be a tree. Italiano, Leonardi, and Oriolo [15] show that this variant can be solved in polynomial time.

($BalG$) Balanced thresholds, graph reservation: The same as ($BalT$), except that an arbitrary graph reservation is allowed.

($AsymT$) Asymmetric thresholds, tree reservation: This problem is NP-hard [11]. Constant approximation algorithms are presented in [11, 12]. Interestingly, while the algorithm in [11] is deterministic, the algorithm in [12] is randomized and seems difficult to derandomize.

($AsymG$) Asymmetric thresholds, graph reservation: This is the VPND problem defined above. We have seen that this problem is NP-hard. The randomized 5.55-approximation result presented in [12] in fact compares the computed tree solution to an optimal graph reservation.

Simplifying assumptions and a lower bound. Following [12], we make some simplifying assumptions without loss of generality. By duplicating nodes, we can assume that each terminal is either a *sender* s, with $b_{out}(s) = 1$ and $b_{in}(s) = 0$, or a *receiver* r, with $b_{out}(r) = 0$ and $b_{in}(r) = 1$. This simplifying assumption is feasible as long as we make sure that the selected paths in our solution between copies of a terminal v and copies of a terminal u are all equal. The algorithms presented in this paper can easily be adapted to run in polynomial time even when the thresholds are not polynomially bounded and to satisfy the consistence property described above. Let $\mathcal{S}$ and $\mathcal{R}$ be the set of senders and the set of receivers, respectively. By $S = |\mathcal{S}|$ and $R = |\mathcal{R}|$ we denote the corresponding cardinalities. For symmetry reasons, we always assume $R \geq S$.

We can now interpret VPND as follows. Let $B = (\mathcal{S} \cup \mathcal{R}, E^B)$ be the complete bipartite graph with nodes partitioned into senders and receivers. We have to reserve capacities on the edges of G and we have to specify a set of paths $\mathcal{P}$ in graph G containing one path P_{sr} for each edge $sr \in E^B$ such that each bipartite matching of B can be routed along these paths. In other words, for each edge $e \in E$, the reservation $u(e)$ has to satisfy the following condition:

$$(1) \qquad \left|\{P_{rs} \in \mathcal{P} \mid e \in P_{rs} \text{ and } rs \in M\}\right| \leq u(e) \quad \text{for each matching } M \text{ in } B.$$

Notice that for a fixed set of paths $\mathcal{P}$, an optimal reservation of capacity is the componentwise minimal u satisfying (1). (In particular, given $\mathcal{P}$, the integral capacity $u(e)$ of edge e can be obtained by a maximum bipartite matching computation.) Thus, a solution to VPND can be encoded by specifying only a set of paths $\mathcal{P}$ in G.

The cost of a bipartite matching between senders and receivers in the metric closure of G is obviously a lower bound on OPT, the value of an optimum solution to the VPND instance. We denote the shortest path distance between nodes u and v of G by $\ell(u, v)$. Thus, if edges (r, s) in B are assigned weights $\ell(r, s)$, then the cost of any matching in B is a lower bound on OPT. This lower bound is used in the analysis of all previous constant-factor approximation algorithms for VPND.

Lemma 1 (see [12]). *Let $B = (\mathcal{S} + \mathcal{R}, E^B)$ be the complete bipartite graph on the senders and receivers with edge-weights $\ell : E^B \to \mathbb{R}_+$ given by the shortest path distances in the graph G. Then, the weight of any matching in B is a lower bound on OPT.*

Contribution of this paper. The design of good approximation results usually requires two main ingredients—cleverly constructed algorithms and thoroughly chosen lower bounds on the optimum such that the quality of the computed solutions can be assessed in terms of the lower bounds. We considerably advance the state of the art of approximating VPND by making contributions to both ingredients.

In section 2 we present a new lower bound which strengthens the one stated in Lemma 1. We prove that the weight of *any matching* (not necessarily bipartite) on the union of the senders and at most S receivers is at most OPT. The edge-weights in the matching are again the shortest path distances in G. This new lower bound relies on an interesting interrelation between a special case of VPND and 2-commodity flows. Its proof is based on an application of Hu's 2-commodity flow theorem [13].

In section 3 we employ the new lower bound in order to show that the following simple algorithm achieves performance ratio $1 + R/S$: Find a vertex $v \in V$ which minimizes the sum of the shortest path distances between v and the union of senders and receivers; cumulatively install a capacity of one on each such shortest path. One interesting consequence of this result is that ($BalG$), VPND with balanced thresholds and graph reservation, has a 2-approximation. Thus our result improves upon the 3-approximation of Italiano, Leonardi, and Oriolo [15] for this problem and generalizes the 2-approximation for ($SymG$) by Gupta et al. [11].

In section 4 we present a new randomized algorithm for VPND. The algorithm chooses a random subset of receivers and connects each sender via its own Steiner tree to this subset. The remaining receivers are then connected to the randomly chosen subset of receivers by shortest paths. Due to the Steiner trees for each individual sender, the resulting solution has in general no tree structure. In contrast to our new approach, the previous algorithm by Gupta, Kumar, and Roughgarden [12] constructs only one random "high-bandwidth core" which is a Steiner tree with high capacity. All previous approximation algorithms for VPND produce tree solutions.

In section 4 we show also that our new algorithm in combination with the simple algorithm from above yields a 3.79 randomized approximation algorithm. The previously best known algorithm [12] achieves performance ratio 5.55.

Our VPND algorithm uses a Steiner tree approximation algorithm as a subroutine. Dreyfus and Wagner [5] showed how to compute optimum Steiner trees in polynomial time when the number of terminals is logarithmic. In section 5, by replacing the approximate Steiner tree computation with an exact one whenever the number of terminals is sufficiently small, we eventually obtain a 3.55-approximation algorithm. To the best of our knowledge, this is the first time a result from exact (exponential) algorithms leads to an improved polynomial-time approximation algorithm, which might be of independent interest.

Related work. We have seen that the Steiner tree problem is a special case of VPND. The current best approximation ratio for the Steiner tree problem is $\rho_{st} < 1.55$ [18]. A related problem is *buy-at-bulk* network design (see, e.g., [1, 2]). Here, there is a fixed demand $d_{i,j}$ between each pair of nodes in the graph, specifying the amount of flow which has to be sent from i to j. The costs of the capacities however is a concave function on the amount purchased, which reflects "economies of scale." Gupta, Kumar, and Roughgarden [12] consider the single source buy-at-bulk network design problem and present a simple constant-factor approximation algorithm.

Another important issue in this context is to cope with arc failures in the network [3, 4]. Italiano, Rastogi, and Yener [16] consider the problem of restoring the network, when at most one arc in a tree solution to VPND might fail and provide a constant-factor approximation algorithm.

It is conjectured that $(SymG)$ can be solved in polynomial time. It is in fact conjectured that an optimal solution to $(SymT)$ is also an optimal solution to $(SymG)$. Hurkens, Keijsper, and Stougie [14] show that this conjecture holds for rings. The authors also describe an integer programming formulation for VPND, which proves that a fractional version can be solved in polynomial time. This fractional version allows us to specify several paths for each sender-receiver pair and requires the fraction for each of these paths, which describes how the commodity has to be split.

2. A new lower bound via Hu's 2-commodity flow theorem.

This section is devoted to proving a new lower bound on the cost of an optimal solution to VPND. Generalizing Lemma 1, we prove that the cost of an arbitrary (not necessarily bipartite) matching between terminals in $\mathcal{S} \cup \mathcal{R}'$ is at most OPT for any subset of receivers $\mathcal{R}' \subseteq \mathcal{R}$ of cardinality $|\mathcal{R}'| = S$. The proof of this result is based on Hu's classical 2-commodity flow theorem [13].

THEOREM 1 (Hu's 2-commodity flow theorem). *Let $G = (V, E)$ be a graph and let s_1, r_1, s_2, r_2 be pairs of vertices of G; let $u : E \to \mathbb{R}_+$ be a capacity function on the edges and let $d_1, d_2 \in \mathbb{R}_+$. There exists a (fractional) 2-commodity flow of value d_1, d_2 if and only if the* cut condition *is satisfied, i.e., if and only if $u(\delta(U)) \geq$*

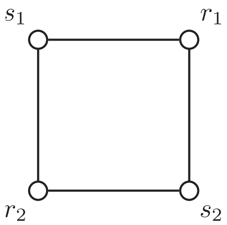


FIG. 1. *An undirected graph with unit capacities on the edges. While there exist integral* 2*-commodity flows satisfying unit demands for terminal pairs* $\{s_1, r_1\}, \{s_2, r_2\}$ *and for terminal pairs* $\{s_1, r_2\}, \{s_2, r_1\}$*, there is only a half-integral* 2*-commodity flow for terminal pairs* $\{s_1, s_2\}, \{r_1, r_2\}$*.*

$d_1\,\chi_1(U)+d_2\,\chi_2(U)$ *for each* $U \subseteq V$*. Here* $\delta(U)$ *denotes the cut induced by* U *and, for* $i \in \{1, 2\}$*,* $\chi_i(U) = 1$ *if the cut* $\delta(U)$ *separates* s_i *from* r_i*, and* $\chi_i(U) = 0$ *otherwise.*

Hu's theorem immediately implies the following corollary.

COROLLARY 1. *Let* $G = (V, E)$ *be an undirected graph with edge capacity function* $u : E \to \mathbb{R}_+$ *and* $s_1, s_2, r_1, r_2 \in V$*. In the following, all demand values are equal to* 1*. If there exists a feasible* 2*-commodity flow for terminal pairs* $\{s_1, r_1\}, \{s_2, r_2\}$ *and for terminal pairs* $\{s_1, r_2\}, \{s_2, r_1\}$*, then there also exists a feasible* 2*-commodity flow for terminal pairs* $\{s_1, s_2\}, \{r_1, r_2\}$*.*

Proof. Hu's 2-commodity flow theorem states that there exists a feasible 2-commodity flow if and only if the "cut condition" is satisfied. In the case of unit demands, the cut condition says that, for all $U \subseteq V$, the capacity $u(\delta(U))$ of the cut induced by U must be at least the number of terminal pairs which are separated by the cut.

It thus remains to show that the cut condition holds for terminal pairs $\{s_1, s_2\}$, $\{r_1, r_2\}$ if it holds for $\{s_1, r_1\}$, $\{s_2, r_2\}$ and for $\{s_1, r_2\}$, $\{s_2, r_1\}$. Consider an arbitrary $U \subseteq V$. If the corresponding cut separates neither $\{s_1, s_2\}$ nor $\{r_1, r_2\}$, nothing needs to be shown. If $\delta(U)$ separates one terminal pair, say, $\{s_1, s_2\}$, then it separates either $\{s_1, r_1\}$ or $\{s_2, r_1\}$ since s_1 and s_2 lie on different sides of the cut. In particular, the capacity of the cut is at least 1. Finally, if $\delta(U)$ separates both terminal pairs $\{s_1, s_2\}, \{r_1, r_2\}$, then it must separate either $\{s_1, r_1\}$ and $\{s_2, r_2\}$ or $\{s_1, r_2\}$ and $\{s_2, r_1\}$. In both cases the capacity of the cut is at least 2. □

We remark that Corollary 1 is no longer true if we replace "2-commodity flow" by "integral 2-commodity flow." A counterexample is given in Figure 1. Even, Itai, and Shamir show that finding an integer 2-commodity flow is NP-hard [9]. On the other hand, Hu's result states that there always exists a half-integral flow in this case. For a more detailed account of results we refer the reader to Schrijver's book [19, Chapter 71].

The next lemma shows that a partition of the senders and receivers into k-subsets each and the "addition" of the optimal solutions of the k subproblems induced by the pairs of subsets provide a lower bound on the optimal solution.

LEMMA 2. *Suppose that* $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k$ *is a partition of* $\mathcal{S}$ *and that* $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_k$ *is a partition of* $\mathcal{R}$*. Let* I_i *be the* VPND *instance on graph* G *with senders* $\mathcal{S}_i$ *and receivers* $\mathcal{R}_i$*, and let* OPT_i *be the value of an optimal solution to instance* I_i*. Then the following holds:*

$$\sum_{i=1}^{k} OPT_i \le OPT.$$

Proof. Let $\mathcal{P}$ be an optimal set of paths for the original VPND instance with resulting capacity reservation $u : E \to \mathbb{R}_+$. The subset $\mathcal{P}_i \subseteq \mathcal{P}$ of paths with endpoints in $\mathcal{S}_i \cup \mathcal{R}_i$ defines a solution to subinstances I_i with capacity reservation $u_i : E \to \mathbb{R}_+$. It suffices to show that $u(e) \geq \sum_{i=1}^{k} u_i(e)$ for each edge $e \in E$.

It follows from (1) that for each $i = 1, 2, \ldots, k$

$$u_i(e) = \max_{M_i} |\{P_{rs} \in \mathcal{P}_i \mid e \in P_{rs} \text{ and } rs \in M_i\}|,$$

where M_i ranges over all bipartite matchings between senders $\mathcal{S}_i$ and receivers $\mathcal{R}_i$. We denote the matching for which the maximum is attained by $\tilde{M}_i$. Then, the disjoint union $\tilde{M} := \bigcup_{i=1}^{k} \tilde{M}_i$ is a bipartite matching between senders $\mathcal{S}$ and receivers $\mathcal{R}$. It thus follows from (1) that

$$\begin{aligned} u(e) &\geq |\{P_{rs} \in \mathcal{P} \mid e \in P_{rs} \text{ and } rs \in \tilde{M}\}| \\ &= \sum_{I=1}^{k} |\{P_{rs} \in \mathcal{P}_i \mid e \in P_{rs} \text{ and } rs \in \tilde{M}_i\}| = \sum_{i=1}^{k} u_i(e) \end{aligned}$$

for each edge $e \in E$. This concludes the proof. □

We are now ready to prove the following theorem which gives a new lower bound on the cost of an optimal VPND solution.

THEOREM 2. *Consider an arbitrary subset $\mathcal{R}' \subseteq \mathcal{R}$ of S receivers. Let M be a matching in the complete graph on $\mathcal{S} \cup \mathcal{R}'$. Then*

$$\sum_{vw \in M} \ell(v, w) \leq OPT.$$

Proof. Let $\mathcal{S} = \{s_1, s_2, \ldots, s_S\}$ and $\mathcal{R}' = \{r_1, r_2, \ldots, r_S\}$. It suffices to prove the claim for perfect matchings. Suppose that the matching consists of the edges

$$s_1s_2,\ s_3s_4, \ldots, s_{2k-1}s_{2k} \quad \text{and} \quad r_1r_2,\ r_3r_4, \ldots, r_{2k-1}r_{2k},$$
$$\text{and} \quad s_{2k+1}r_{2k+1},\ s_{2k+2}r_{2k+2}, \ldots, s_Sr_S.$$

Consider the following partition of $\mathcal{S}$ and $\mathcal{R}'$ into $S - k$ subsets $\mathcal{S}_i$ and $\mathcal{R}'_i$:

$$\begin{aligned} &\mathcal{S}_i = \{s_{2\,i-1}, s_{2\,i}\},\ \mathcal{R}'_i = \{r_{2\,i-1}, r_{2\,i}\}, && 1 \leq i \leq k, \\ &\mathcal{S}_i = \{s_i\},\ \mathcal{R}'_i = \{r_i\}, && 2\,k+1 \leq i \leq S. \end{aligned}$$

By Lemma 2, the sum of the optimum solutions OPT_i of the VPND subinstances I_i with senders $\mathcal{S}_i$ and receivers $\mathcal{R}'_i$ is a lower bound on OPT. Thus we need only to prove that $\ell(s_{2\,i-1}, s_{2\,i}) + \ell(r_{2\,i-1}, r_{2\,i}) \leq OPT_i$ for each $1 \leq i \leq k$.

An optimum solution to the subproblem I_i, $1 \leq i \leq k$, is a reservation of capacities that supports 2-commodity flows with unit demands for the terminal pairs $\{s_{2\,i-1}, r_{2\,i-1}\}$, $\{s_{2\,i}, r_{2\,i}\}$ and for the terminal pairs $\{s_{2\,i-1}, r_{2\,i}\}$, $\{s_{2\,i}, r_{2\,i-1}\}$. By Corollary 1, it must also support a 2-commodity flow for the terminal pairs $\{s_{2\,i-1}, s_{2\,i}\}$, $\{r_{2\,i-1}, r_{2\,i}\}$. Therefore, the cost of this solution is at least $\ell(s_{2\,i-1}, s_{2\,i}) + \ell(r_{2\,i-1}, r_{2\,i})$. This concludes the proof. □

3. The quality of a simple routing scheme. Consider the following simple VPND algorithm.

Algorithm 1 (simple routing scheme).

(1) *Compute a vertex $v \in V$ such that $\sum_{s \in \mathcal{S}} \ell(s, v) + \sum_{r \in \mathcal{R}} \ell(r, v)$ is minimal.*

(2) *Add one unit of capacity along the shortest path between each $u \in \mathcal{R} \cup \mathcal{S}$ and v.*

Algorithm 1 selects the vertex $v \in V$ which minimizes the sum of the distances from v to the union of the senders and receivers, and reserves one unit of capacity along each of the shortest paths. Note that the effects of installing capacities along the shortest paths is cumulative. In other words, if k shortest paths share the same edge, the algorithm adds k units of capacity to that edge. Moreover, we assume the shortest paths are computed with a consistent tie-breaking rule. This way, the edges with nonzero capacity form a tree.

Algorithm 1 produces an optimal tree reservation in the symmetric case (*SymT*) [11] and in the balanced case (*BalT*) [15]. In the symmetric case, Gupta et al. [11] showed that the tree produced by the algorithm is a 2-approximate solution to the optimum graph reservation. Italiano, Leonardi, and Oriolo [15] show that, in the balanced case, the produced tree is a 3-approximate solution to the optimum graph reservation.

In this section, we apply our new lower-bound result to show that the algorithm produces a tree solution which is within $1 + R/S$ from the optimum graph reservation. Thus (*BalG*) can also be approximated within a factor of two. We first need the following simple lower bound on OPT. Let $\ell(E') := \sum_{uv \in E'} \ell(u, v)$ for any subset of edges E'.

Lemma 3. *The sum of the distances between each sender-receiver pair is at most R times the optimum:*

$$\sum_{s \in \mathcal{S}} \sum_{r \in \mathcal{R}} \ell(s, r) \leq R \cdot OPT. \tag{2}$$

Proof. Let $B = (\mathcal{S} + \mathcal{R}, E^B)$ be the complete bipartite graph on the senders and receivers with edge-weights $\ell : E^B \to \mathbb{R}_+$ given by the shortest path distances in the graph G. The edges of B can be partitioned into a set $\mathcal{M}$ of R matchings. By Lemma 1, the cost $\ell(M)$ of each $M \in \mathcal{M}$ is at most OPT. Hence

$$\sum_{s \in \mathcal{S}} \sum_{r \in \mathcal{R}} \ell(s, r) = \sum_{M \in \mathcal{M}} \ell(M) \leq R \cdot OPT.$$

We are now ready to bound the approximation ratio provided by Algorithm 1.

Theorem 3. *Algorithm 1 is a $(1 + R/S)$-approximation algorithm for* VPND.

Proof. Let $G^m = (\mathcal{R} \cup \mathcal{S}, E^m)$ be the metric closure of $\mathcal{R} \cup \mathcal{S}$, i.e., the complete graph on $\mathcal{R} \cup \mathcal{S}$ with edge-weight $\ell(u, v)$ given by the shortest path distance between u and v in G. We show that there exists a node $u \in \mathcal{R} \cup \mathcal{S}$ such that the cost of the complete star centered at u satisfies

$$\sum_{v \in \mathcal{R} \cup \mathcal{S}} \ell(u, v) \leq (1 + R/S)\, OPT.$$

If $R = S$, the edges of E^m can be partitioned into $2S - 1$ perfect matchings. By Theorem 2, the weight of each matching is a lower bound on OPT. Since each edge is contained in exactly two stars of G^m,

$$\sum_{v\in\mathcal{R}\cup\mathcal{S}} \sum_{u\in\mathcal{R}\cup\mathcal{S}} \ell(v,\,u) = 2\ell(E^m) \leq 2(2S-1)\,OPT.$$

Thus there must exist one star, whose weight is at most $2(2S-1)\,OPT/(2S) < 2\,OPT$.

Suppose for the remainder of the proof that $R > S$. In the following we denote by $\mathcal{M}_\mathcal{S}$ and $\mathcal{M}_\mathcal{R}$ the set of matchings of cardinality at most $\lfloor S/2 \rfloor$ involving only senders and only receivers, respectively. Theorem 2 implies the inequality

$$\ell(M_\mathcal{S}) + \ell(M_\mathcal{R}) \;\leq\; OPT \qquad \text{for each } M_\mathcal{S} \in \mathcal{M}_\mathcal{S},\, M_\mathcal{R} \in \mathcal{M}_\mathcal{R}. \tag{3}$$

In consideration of (3), we distinguish two cases.

(A) $\ell(M_\mathcal{S}) \leq OPT/2$ *for each* $M_\mathcal{S} \in \mathcal{M}_\mathcal{S}$. Consider the subgraph $G^m_\mathcal{S}$ of G^m which is induced by the senders. The edges $E^m_\mathcal{S}$ of $G^m_\mathcal{S}$ can be partitioned into S matchings. Therefore

$$\sum_{s'\in\mathcal{S}} \sum_{s\in\mathcal{S}} \ell(s',s) \;=\; 2\,\ell(E^m_\mathcal{S}) \leq 2\,S\frac{OPT}{2} = S\,OPT. \tag{4}$$

Combining (4) and (2), one obtains

$$\sum_{s'\in\,\mathcal{S}} \left(\sum_{s\in\,\mathcal{S}} \ell(s',s) + \sum_{r\in\,\mathcal{R}} \ell(s',r) \right) \;\leq\; (S+R)\,OPT.$$

Therefore there is a sender s^* such that

$$\sum_{v\in\,\mathcal{S}\,\cup\mathcal{R}} \ell(s^*,\,v) \leq (1+R/S)OPT.$$

(B) $\ell(M_\mathcal{S}) > OPT/2$ *for some maximum weight matching* $M_\mathcal{S} \in \mathcal{M}_\mathcal{S}$. Let $G^m_\mathcal{R}$ denote the subgraph of G^m which is induced by the receivers. We will show below that, for any maximal matching $\tilde{M}$ in $G^m_\mathcal{R}$,

$$\ell(\tilde{M}) \leq (R/S)\,OPT/2. \tag{5}$$

Since the edges of $G^m_\mathcal{R}$ can be partitioned into at most R maximal matchings, we can then argue in a similar manner as in case (A) that

$$\sum_{r'\in\,\mathcal{R}} \sum_{r\in\,\mathcal{R}} \ell(r,\,r') \;\leq\; 2\,R\,\frac{R}{S}\,\frac{OPT}{2} \;=\; (R^2/S)\,OPT.$$

Together with (2) this implies

$$\sum_{r'\in\,\mathcal{R}} \left(\sum_{s\in\,\mathcal{S}} \ell(s,r') + \sum_{r\in\,\mathcal{R}} \ell(r,r') \right) \;\leq\; (R + R^2/S)\,OPT,$$

from which we conclude that there is a receiver r^* satisfying

$$\sum_{v\in\,\mathcal{S}\,\cup\mathcal{R}} \ell(v,\,r^*) \leq (1+R/S)OPT.$$

It remains to prove (5). We distinguish between the two subcases in which S is even and S is odd.

(B.1) *S is even.* Theorem 2, together with the assumption $\ell(M_{\mathcal{S}}) > OPT/2$, implies $\ell(M_{\mathcal{R}}) \leq OPT/2$ for each matching $M_{\mathcal{R}} \in \mathcal{M}_{\mathcal{R}}$. Note that $|\tilde{M}| = \lfloor R/2 \rfloor \geq S/2$. Consider the $S/2$ most expensive edges M' of $\tilde{M}$. Since $M' \in \mathcal{M}_{\mathcal{R}}$, $\ell(M') \leq OPT/2$. Hence the average cost of one edge of $\tilde{M}$ is upper bounded by $(OPT/2)/(S/2) = OPT/S$. Since $\tilde{M}$ has at most $R/2$ edges, we get

$$\ell(\tilde{M}) \leq |\tilde{M}| \frac{OPT}{S} \leq \frac{R}{2}\,\frac{OPT}{S} = (R/S)OPT/2.$$

(B.2) *S is odd.* There is a sender s^* which is missed by the maximum cost matching $M_{\mathcal{S}}$ of $G^m_{\mathcal{S}}$. For each matching $M_{\mathcal{R}} \in \mathcal{M}_{\mathcal{R}}$ and $r_1^* \in \mathcal{R}$ which is not matched by $M_{\mathcal{R}}$, Theorem 2 yields

$$\ell(M_{\mathcal{S}}) + \ell(M_{\mathcal{R}}) + \ell(s^*, r_1^*) \;\leq\; OPT,$$

and hence

$$\ell(M_{\mathcal{R}}) + \ell(s^*, r_1^*) \;\leq\; OPT/2.$$

Consider any other receiver r_2^* which is not matched by $M_{\mathcal{R}}$. This receiver must exist since $R > S$. By the triangle inequality one has $\ell(r_1^*, r_2^*) \leq \ell(s^*, r_1^*) + \ell(s^*, r_2^*)$. As a result we get

$$\ell(M_{\mathcal{R}}) + 1/2\,\ell(r_1^*, r_2^*) \;\leq\; OPT/2 \tag{6}$$

for each matching $M_{\mathcal{R}} \in \mathcal{M}_{\mathcal{R}}$ and receivers r_1^*, r_2^* which are missed by $M_{\mathcal{R}}$.

Now consider the $(S-1)/2$ most expensive edges M' of $\tilde{M}$, and let e' be the next most expensive edge of $\tilde{M}$. Since $M' \in \mathcal{M}_{\mathcal{R}}$, by (6),

$$\ell(M') + 1/2\,\ell(e') \leq OPT/2.$$

It follows that half the average cost of one edge of $\tilde{M}$ is upper bounded by $(OPT/2)/(S-1+1) = OPT/(2\,S)$, from which we conclude

$$\ell(\tilde{M}) \leq 2\,|\tilde{M}| \frac{OPT}{2\,S} \leq 2\frac{R}{2}\frac{OPT}{2\,S} = (R/S)\,OPT/2.$$

4. A new algorithm for VPND.

In section 3 we described an algorithm which guarantees a good approximation ratio for R close to S. Here we present a better approximation algorithm in case R is sufficiently larger than S.

The algorithm by Gupta et al. [12] constructs one random "high-bandwidth core," i.e., a small Steiner tree with high capacity, where the terminals are a random sender and a random subset of receivers. Such a Steiner tree collects and distributes the demands from outside, and routes them along its high capacity paths. Our algorithm is also based on Steiner tree computations, but we proceed by connecting each sender to a previously sampled subset of receivers via S distinct Steiner trees of low capacity, and by connecting the other receivers along their shortest paths to the sampled subset (see Figure 2). More precisely, our algorithm works as follows.

Algorithm 2.

(1) *Partition $\mathcal{R}$ into S subsets uniformly at random. Select one subset $\mathcal{R}'$ uniformly at random, among the nonempty subsets in the partition.*

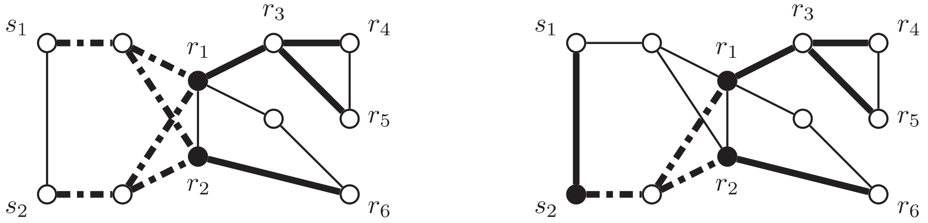


FIG. 2. *Intuitive comparison between Algorithm* 2 *(on the left) and the algorithm in* [12]. *Black nodes form the randomly sampled subsets. Positive capacity is reserved on thick edges only: Dashed thick edges correspond to Steiner trees, while full ones correspond to shortest paths.*

(2) *For each sender $s \in \mathcal{S}$, compute a ρ_{st}-approximate Steiner tree $T(s)$ on $\{s\} \cup \mathcal{R}'$, and add one unit of capacity to each edge of $T(s)$.*

(3) *Add one unit of capacity along the shortest path between each receiver $r \in \mathcal{R}$ and $\mathcal{R}'$.*

It remains to specify the path between each sender-receiver pair (s, r). Assume that the shortest paths are computed with a consistent tie-breaking rule. Let r^* be the receiver in $\mathcal{R}'$ which is closest to r. The path P_{sr} between s and r is obtained by concatenating the (simple) path between s and r^* in $T(s)$, with the shortest path between r^* and r.

The thereby produced solution is not a tree solution. Though an optimal tree solution is a constant-factor approximation to an optimal graph solution, it is known [11] that an optimal solution to $(AsymT)$ is not an optimal solution to VPND. All previous constant-factor approximation algorithms for VPND, however [12, 7], produce tree reservations.

Before we proceed with the analysis of Algorithm 2, we state a corollary of Lemma 2. Here, given a subset V' of nodes, we denote the cost of the optimum Steiner tree on V' by $st(V')$.

COROLLARY 2 (see [12]). *Let $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_s$ be a partition of $\mathcal{R}$ into S (disjoint) subsets. Consider an arbitrary perfect matching between $\mathcal{S}$ and this family of subsets. Let $\mathcal{R}(s)$ be the subset matched with sender s. The sum over $\mathcal{S}$ of the costs of the optimum Steiner trees on $\{s\} \cup \mathcal{R}(s)$ is a lower bound on OPT:*

$$\sum_{s \in \mathcal{S}} st(\{s\} \cup \mathcal{R}(s)) \leq OPT.$$

Proof. This follows from Lemma 2 since a solution of a subinstance $\{s\}, \mathcal{R}(s)$ contains a Steiner tree with terminals $\{s\} \cup \mathcal{R}(s)$. ▯

THEOREM 4. *Algorithm* 2 *is a $(2 + \rho_{st})/(1 - e^{-R/S})$-approximation algorithm for* VPND.

Theorem 4 is a straightforward consequence of the following lemmas.

LEMMA 4. *For a uniformly chosen random sender s',*

$$E[st(\{s'\} \cup \mathcal{R}')] \leq \frac{OPT}{S(1 - e^{-R/S})}.$$

Proof. Consider the following random process. For each receiver r, we *assign* r to a sender s chosen uniformly at random. Let $\mathcal{R}(s)$ be the subset of receivers assigned to s. Note that the subsets $\mathcal{R}(s)$ partition $\mathcal{R}$ into S (possibly empty) subsets. Thus, by Corollary 2,

$$\sum_{s\in\mathcal{S}} st(\{s\}\cup\mathcal{R}(s)) \le OPT.$$

This means that, for the random sender s',

$$E[st(\{s'\}\cup\mathcal{R}(s'))] \le OPT/S.$$

Let A denote the event that $\mathcal{R}(s')$ is empty. By elementary probability theory,

$$E[st(\{s'\}\cup\mathcal{R}(s'))] = P(A)\, E[st(\{s'\}\cup\mathcal{R}(s')) \mid A] + P(\overline{A})\, E[st(\{s'\}\cup\mathcal{R}(s')) \mid \overline{A}].$$

Now observe that

$$P(\overline{A}) = 1-(1-1/S)^R \ge 1-e^{-R/S}.$$

Moreover

$$E[st(\{s'\}\cup\mathcal{R}(s')) \mid A] = E[st(\{s'\})] = 0.$$

Thus

$$E[st(\{s'\}\cup\mathcal{R}(s')) \mid \overline{A}] = \frac{E[st(\{s'\}\cup\mathcal{R}(s'))]}{P(\overline{A})} \le \frac{OPT}{S(1-e^{-R/S})}.$$

The claim follows by observing that, given $\overline{A}$, $\mathcal{R}(s')$ and $\mathcal{R}'$ are identically distributed. Thus

$$E[st(\{s'\}\cup\mathcal{R}')] = E[st(\{s'\}\cup\mathcal{R}(s')) \mid \overline{A}] \le \frac{OPT}{S(1-e^{-R/S})}.$$

Lemma 5. *The expected cost of the capacity installed in the second step of Algorithm* 2 *is at most*

$$\rho_{st}\, OPT/(1-e^{-R/S}).$$

Proof. The expected cost considered is

$$E\left[\sum_{s\in\mathcal{S}} c(T(s))\right],$$

where $c(T(s))$ is the cost of the Steiner tree $T(s)$. Of course

$$c(T(s)) \le \rho_{st}\, st(\{s\}\cup\mathcal{R}').$$

Let s' be a sender chosen uniformly at random. By Lemma 4,

$$E\left[\sum_{s\in\mathcal{S}} c(T(s))\right] \le \rho_{st}\, E\left[\sum_{s\in\mathcal{S}} st(\{s\}\cup\mathcal{R}')\right] = \rho_{st}\, S\, E[st(\{s'\}\cup\mathcal{R}')]$$

$$\le \rho_{st}\, S\, \frac{OPT}{S(1-e^{-R/S})} = \frac{\rho_{st}\, OPT}{1-e^{-R/S}}.$$

LEMMA 6. *The expected cost of the capacity installed in the third step of Algorithm* 2 *is at most*

$$2\,OPT/(1-e^{-R/S}).$$

Proof. Let r' be an arbitrary receiver in $\mathcal{R}'$. Observe that the probability of any other receiver r'' being in $\mathcal{R}'$, given that r' is in $\mathcal{R}'$, is $1/S$. In fact, let $\mathcal{R}^*$ be a random (possibly empty) partition element. Then

$$P(r' \in \mathcal{R}') = P(r' \in \mathcal{R}^* \mid \mathcal{R}^* \neq \emptyset) = \frac{P(r' \in \mathcal{R}^* \cap \mathcal{R}^* \neq \emptyset)}{P(\mathcal{R}^* \neq \emptyset)}$$
$$= \frac{P(r' \in \mathcal{R}^*)}{P(\mathcal{R}^* \neq \emptyset)} = \frac{1/S}{1-(1-1/S)^R}.$$

By basically the same argument

$$P(r' \in \mathcal{R}' \cap r'' \in \mathcal{R}') = \frac{(1/S)^2}{1-(1-1/S)^R}.$$

Thus

$$P(r'' \in \mathcal{R}' \mid r' \in \mathcal{R}') = \frac{P(r'' \in \mathcal{R}' \cap r' \in \mathcal{R}')}{P(r' \in \mathcal{R}')} = \frac{(1/S)^2/(1-(1-1/S)^R)}{(1/S)/(1-(1-1/S)^R)} = \frac{1}{S}.$$

Now consider the following random process. In step t, let $\mathcal{A}_t$ be the subset of receivers considered so far, and let $\mathcal{B}_t$ be the bought receivers in $\mathcal{A}_t$. Initially $\mathcal{A}_1 = \mathcal{B}_1 = \{r'\}$, where r' is a random receiver. In step t, we consider the receiver $r_t \in \mathcal{R} \setminus \mathcal{A}_t$ which is closest to $\mathcal{B}_t$, and we set $\mathcal{A}_{t+1} = \mathcal{A}_t \cup \{r_t\}$. Then, with probability $1/S$, we *buy* r_t; that is,

- we add S units of capacity along the shortest path from r_t to $\mathcal{B}_t$, and
- we set $\mathcal{B}_{t+1} = \mathcal{B}_t \cup \{r_t\}$.

Otherwise, we *rent* r_t; that is,

- we add one unit of capacity along the shortest path from r_t to $\mathcal{B}_t$, and
- we set $\mathcal{B}_{t+1} = \mathcal{B}_t$.

Note that, at the end of the process, the set of bought receivers $\mathcal{B}'$ has the same distribution as the set $\mathcal{R}'$ of selected receivers. Let $\ell(v, V') = \min_{v' \in V'}\{\ell(v, v')\}$ denote the minimum distance between a node v and a subset of nodes V'. The expected cost of the third step of the algorithm is upper bounded by the expected cost c_{rent} of renting receivers

$$E\left[\sum_{r \in \mathcal{R}\setminus\mathcal{R}'} \ell(r, \mathcal{R}')\right] = E\left[\sum_{r_t \in \mathcal{R}\setminus\mathcal{B}'} \ell(r_t, \mathcal{B}')\right] \leq E\left[\sum_{r_t \in \mathcal{R}\setminus\mathcal{B}'} \ell(r_t, \mathcal{B}_t)\right]$$
$$= E\left[\sum_{r_t \in \mathcal{R}} \left(1-\frac{1}{S}\right) \ell(r_t, \mathcal{B}_t)\right] = c_{rent},$$

where the inequality comes from the fact that $\mathcal{B}_t \subseteq \mathcal{B}'$ for any t.

The expected cost c_{buy} of buying receivers is an upper bound on c_{rent}:

$$c_{rent} = E\left[\sum_{r_t\in\mathcal{R}}\left(1-\frac{1}{S}\right)\ell(r_t,\,\mathcal{B}_t)\right] \le E\left[\sum_{r_t\in\mathcal{R}}\left(\frac{1}{S}\right)S\,\ell(r_t,\,\mathcal{B}_t)\right]$$
$$= E\left[\sum_{r_t\in\mathcal{B}'} S\,\ell(r_t,\,\mathcal{B}_t)\right] = c_{buy}\,.$$

Note that c_{buy} is equal to S times the expected cost of the Steiner tree on $\mathcal{B}'$ which is obtained via the minimum spanning tree heuristic (starting Prim's algorithm from node r'). It is well known that this heuristic provides a 2-approximate solution [17]. Thus

$$c_{buy} \le 2\,S\,E[st(\mathcal{B}')] = 2\,S\,E[st(\mathcal{R}')].$$

For a random sender s', by Lemma 4,

$$E[st(\mathcal{R}')] \le E[st(\{s'\}\cup\mathcal{R}')] \le \frac{OPT}{S(1-e^{-R/S})}.$$

Altogether,

$$E\left[\sum_{r\in\mathcal{R}}\ell(r,\mathcal{R}')\right] \le c_{rent} \le c_{buy} \le 2\,S\,\frac{OPT}{S(1-e^{-R/S})} = \frac{2\,OPT}{1-e^{-R/S}}.$$

In section 3 we described a $(1+R/S)$-approximation algorithm. The factors $1+R/S$ and $(2+\rho_{st})/(1-e^{-R/S})$ are equal for $R/S = 2.78\ldots < 2.79$. Note that $1+R/S$ is increasing in R/S and $(2+\rho_{st})/(1-e^{-R/S})$ is decreasing in R/S. It follows that a combination (taking the minimum cost solution) of Algorithms 1 and 2 has an expected approximation guarantee of 3.79, which is a considerable improvement compared to the 5.55-approximation ratio achieved by Gupta, Kumar, and Roughgarden [12].

Theorem 5. *The combination (taking the cheaper solution) of Algorithms* 1 *and* 2 *is an expected* 3.79*-approximation algorithm for* VPND.

5. Computing optimum Steiner trees. The performance ratios of Algorithms 1 and 2 meet roughly at $R/S \simeq 2.78$. In this case, the sampled set $\mathcal{R}'$ of receivers has expected constant size. An optimum Steiner tree on a graph with n nodes and t terminals can be computed in $O(3^t\,n + 2^t\,n^2 + n^3)$ time with the Dreyfus–Wagner algorithm [5]. This suggests the following variant of Algorithm 2, which computes optimal Steiner trees in step (2), instead of ρ_{st}-approximate ones, whenever $|\mathcal{R}'| \le \log n$. Here n is the number of nodes in the original graph G. Without loss of generality, we assume n is sufficiently large.

Algorithm 3.

(1) *Partition $\mathcal{R}$ into S subsets uniformly at random. Select one subset $\mathcal{R}'$ uniformly at random, among the nonempty subsets in the partition.*

(2) *For each sender $s \in \mathcal{S}$, compute a ρ_{st}-approximate Steiner tree $T(s)$ on $\{s\}\cup\mathcal{R}'$ if $|\mathcal{R}'| > \log n$. Otherwise, compute an optimum Steiner tree $T(s)$ on $\{s\}\cup\mathcal{R}'$. In both cases, add one unit of capacity to each edge of $T(s)$.*

(3) *Add one unit of capacity along the shortest path between each receiver $r \in \mathcal{R}$ and $\mathcal{R}'$.*

Clearly, Algorithm 3 is a polynomial-time algorithm whose expected approximation guarantee is not worse than the one of Algorithm 2. In particular, if R/S is very large, say, $R/S \geq \log\log n$, the approximation achieved is

$$\frac{2+\rho_{st}}{1-e^{-R/S}} \leq \frac{2+\rho_{st}}{1-1/\log n} = 2+\rho_{st}+o(1) < 3.55.$$

What can be said about the approximation guarantee if $R/S \leq \log\log n$? In that case, the expected size of $\mathcal{R}'$ is $1+(R-1)/S < 1+\log\log n$. The probability that the size of $\mathcal{R}'$ exceeds $\log n$ is at most $(1+\log\log n)/\log n$ by Markov's inequality. Thus with high probability Algorithm 3 computes optimum Steiner trees. When this happens, the approximation ratio is bounded by $3.325\ldots < 3.326$. In the very unlikely event that Algorithm 3 computes ρ_{st}-approximate Steiner trees, the approximation guaranteed by Algorithm 1 alone is $O(\log\log n)$. Hence this event contributes to the expected approximation ratio with $o(1)$ only. This is the intuition behind the following theorem.

THEOREM 6. *For a sufficiently large n, the combination (taking the cheaper solution) of Algorithms* 1 *and* 3 *is an expected* 3.55*-approximation algorithm for* VPND.

Proof. Following the discussion above, if $R/S > \log\log n$, the expected approximation ratio achieved is $2+\rho_{st}+o(1) < 3.55$. Then we can restrict our analysis to the case $R/S \leq \log\log n$. Let APX denote the expected cost of the solution computed. One has

$$APX \leq E\left[\min\left\{(1+R/S)\,OPT, \sum_{s\in\mathcal{S}} c(T(s)) + \sum_{r\in\mathcal{R}} \ell(r,\mathcal{R}')\right\}\right].$$

Of course,

$$APX \leq (1+R/S)OPT, \tag{7}$$

and

$$APX \leq E\left[\sum_{r\in\mathcal{R}} \ell(r,\mathcal{R}')\right] + E\left[\min\left\{(1+R/S)\,OPT, \sum_{s\in\mathcal{S}} c(T(s))\right\}\right].$$

By Lemma 6,

$$E\left[\sum_{r\in\mathcal{R}} \ell(r,\mathcal{R}')\right] \leq \frac{2\,OPT}{1-e^{-R/S}}. \tag{8}$$

Let A denote the event that $R' = |\mathcal{R}'| \leq \log n$. By elementary probability theory one has

$$\begin{aligned} E\left[\min\left\{(1+R/S)OPT, \sum_{s\in\mathcal{S}} c(T(s))\right\}\right] &\leq P(\overline{A})\, E\left[(1+R/S)\,OPT \mid \overline{A}\right] \\ &\quad + P(A)\, E\left[\sum_{s\in\mathcal{S}} c(T(s)) \mid A\right]. \end{aligned} \tag{9}$$

We now consider both terms separately. By Markov's inequality one has $P(\overline{A}) \leq (1 + \log\log n)/\log n$. Thus

$$\begin{aligned} P(\overline{A})E\big[(1 + R/S)\,OPT \mid \overline{A}\big] &\leq P(\overline{A})E\big[(1 + \log\log n)\,OPT \mid \overline{A}\big] \\ &\leq \frac{(1 + \log\log n)^2}{\log n} OPT. \end{aligned}$$

Given A, Algorithm 3 computes optimal Steiner trees $T(s)$ of cost $st(\{s\} \cup \mathcal{R}')$. Also $E\big[st(\{s\} \cup \mathcal{R}') \mid R' \leq h\big]$ is a nondecreasing function of h. Thus, from the proof of Lemma 5, the second term on the right of (9) can be bounded by

$$P(A)\,E\left[\sum_{s \in \mathcal{S}} c(T(s)) \mid A\right] \leq E\left[\sum_{s \in \mathcal{S}} st(\{s\} \cup \mathcal{R}')\right] \leq \frac{OPT}{1 - e^{-R/S}}.$$

One therefore has

$$\begin{aligned} E\left[\min\left\{(1 + R/S)\,OPT, \sum_{s \in \mathcal{S}} c(T(s))\right\}\right] &\leq \frac{(1 + \log\log n)^2}{\log n} OPT + \frac{OPT}{1 - e^{-R/S}} \\ &\leq \left(1 + \frac{(1 + \log\log n)^2}{\log n}\right) \frac{OPT}{1 - e^{-R/S}}. \qquad (10) \end{aligned}$$

Combining (7), (8), and (10), we conclude that

$$APX \leq \min\left\{(1 + R/S)\,OPT, \left(3 + \frac{(1 + \log\log n)^2}{\log n}\right) \frac{OPT}{1 - e^{-R/S}}\right\}.$$

Thus, for a sufficiently large n, the expected approximation ratio for $R/S \leq \log\log n$ is upper bounded by $3.325\ldots < 3.326$. Altogether, the approximation ratio is

$$\min\{3.326,\ 2 + \rho_{st} + o(1)\} < 3.55.$$

COROLLARY 3. *There is an expected* 3.55*-approximation algorithm for* VPND.

Proof. When n is upper bounded by a constant, the optimum solution can be computed in polynomial time by trivial enumeration. The claim follows from Theorem 6. □

REFERENCES

[1] M. Andrews and L. Zhang, *The access network design problem*, in Proceedings of the IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 1998, pp. 40–49.

[2] B. Awerbuch and Y. Azar, *Buy-at-bulk network design*, in Proceedings of the IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 1997, pp. 542–547.

[3] G. Brightwell, G. Oriolo, and F. B. Shepherd, *Reserving resilient capacity in a network*, SIAM J. Discrete Math., 14 (2001), pp. 524–539.

[4] G. Brightwell, G. Oriolo, and F. B. Shepherd, *Reserving resilient capacity for a single commodity with upper-bound constraints*, Networks, 41 (2003), pp. 87–96.

[5] S. E. Dreyfus and R. A. Wagner, *The Steiner problem in graphs*, Networks, 1 (1971/1972), pp. 195–207.

[6] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan, and J. E. van der Merwe, *A flexible model for resource management in virtual private networks*, in Proceedings of the ACH Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, Cambridge, MA, 1999, pp. 95–108.

[7] F. Eisenbrand and F. Grandoni, *An improved approximation algorithm for virtual private network design*, in Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, SIAM, Philadelphia, 2005, pp. 928–932.

[8] F. Eisenbrand, F. Grandoni, G. Oriolo, and M. Skutella, *New approaches for virtual private network design*, in Proceedings of the International Colloquium on Automata, Languages and Programming, Lisbon, Portugal, 2005, pp. 1152–1162.

[9] S. Even, A. Itai, and A. Shamir, *On the complexity of timetable and multicommodity flow problems*, SIAM J. Comput., 5 (1976), pp. 691–703.

[10] J. A. Fingerhut, S. Suri, and J. S. Turner, *Designing least-cost nonblocking broadband networks*, J. Algorithms, 24 (1997), pp. 287–309.

[11] A. Gupta, J. Kleinberg, A. Kumar, R. Rastogi, and B. Yener, *Provisioning a virtual private network: A network design problem for multicommodity flow*, in Proceedings of the ACM Symposium on Theory of Computing, ACM, New York, 2001, pp. 389–398.

[12] A. Gupta, A. Kumar, and T. Roughgarden, *Simpler and better approximation algorithms for network design*, in Proceedings of the ACM Symposium on Theory of Computing, ACM, New York, 2003, pp. 365–372.

[13] T. C. Hu, *Multi-commodity network flows*, Oper. Res., 11 (1963), pp. 344–360.

[14] C. Hurkens, J. Keijsper, and L. Stougie, *Virtual private network design: A proof of the tree routing conjecture on ring networks*, in Proceedings of the International Conference on Integer Programming and Combinatorial Optimization, Berlin, Germany, 2005, pp. 407–421.

[15] G. Italiano, S. Leonardi, and G. Oriolo, *Design of networks in the hose model*, in Proceedings of ARACNE, Rome, Italy, 2002, pp. 65–76.

[16] G. F. Italiano, R. Rastogi, and B. Yener, *Restoration algorithms for virtual private networks in the hose model*, in Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Society, New York, NY, 2002, pp. 131–139.

[17] L. Kou, G. Markowsky, and L. Berman, *A fast algorithm for Steiner trees*, Acta Inform., 15 (1981), pp. 141–145.

[18] G. Robins and A. Zelikovsky, *Improved Steiner tree approximation in graphs*, in Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, SIAM, Philadelphia, 2000, pp. 770–779.

[19] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*, Algorithms and Combinatorics 24, Springer-Verlag, Berlin, 2003.

SIAM J. COMPUT. © 2007 Society for Industrial and Applied Mathematics
Vol. 37, No. 3, pp. 722–756

THE TIME-COMPLEXITY OF LOCAL DECISION IN DISTRIBUTED AGREEMENT*

PARTHA DUTTA[†], RACHID GUERRAOUI[‡], AND BASTIAN POCHON[‡]

Abstract. Agreement is at the heart of distributed computing. In its simple form, it requires a set of processes to decide on a common value out of the values they propose. The time-complexity of distributed agreement problems is generally measured in terms of the number of communication rounds needed to achieve a global decision, i.e., for *all* nonfaulty (correct) processes to reach a decision. This paper studies the time-complexity of *local* decisions in agreement problems, which we define as the number of communication rounds needed for *at least one* correct process to decide. We explore bounds for *early* local decision that depend on the number f of actual failures (that occur in a given run of an algorithm), out of the maximum number t of failures tolerated (by the algorithm). We first consider the synchronous message-passing model where we give tight local decision bounds for three variants of agreement: consensus, uniform consensus, and (nonblocking) atomic commit. We use these results to (1) show that, for consensus, local decision bounds are not compatible with global decision bounds (roughly speaking, they cannot be reached by the same algorithm), and (2) draw the first sharp line between the time-complexity of uniform consensus and atomic commit. Then we consider the eventually synchronous model, where we give tight local decision bounds for synchronous runs of uniform consensus. (In this model, consensus and uniform consensus are similar, atomic commit is impossible, and one cannot bound the number of rounds to reach a decision in nonsynchronous runs of consensus algorithms.) We prove a counterintuitive result that the early local decision bound is the same as the early global decision bound. We also give a matching early deciding consensus algorithm that is significantly better than previous eventually synchronous consensus algorithms.

Key words. distributed systems, agreement problems, lower bounds

AMS subject classifications. 68Q25, 68W15

DOI. 10.1137/S0097539704446220

1. Introduction.

Local versus global agreement decisions. Determining how long it takes to reach agreement among a set of processes is an important question in distributed computing. For instance, the performance of a replicated system is impacted by the performance of the underlying consensus service used to ensure that the replica processes agree on the same order to deliver client requests [20]. Similarly, the performance of a distributed transactional system is impacted by the performance of the underlying atomic commit service used to ensure that the database servers agree on a transaction outcome [15].

Traditionally, lower bounds on the time-complexity of distributed agreement have been stated in terms of the number of communication rounds (also called communication steps) needed for *all* correct processes to decide [21] (i.e., *global decision*), or

*Received by the editors October 7, 2004; accepted for publication (in revised form) March 18, 2007; published electronically June 29, 2007. This paper is an extended and revised version of a paper titled "Tight bounds on early local decisions in uniform consensus" by the same authors, that appeared in the 17*th International Symposium on Distributed Computing* (DISC 2003), Lecture Notes in Comput. Sci. 2848, Springer-Verlag.

http://www.siam.org/journals/sicomp/37-3/44622.html

[†] Bell Labs Research India, Lucent Technologies India Ltd., Bangalore-560095, India (parthad@alcatel-lucent.com). This work was done while this author was a graduate student at Distributed Programming Laboratory, EPFL.

[‡] Distributed Programming Laboratory, EPFL, CH-1015, Lausanne, Switzerland (rachid.guerraoui@epfl.ch, bastian.pochon@epfil.ch).

even *halt* [6], possibly as a function of the number of failures f that actually occur in a given run of an algorithm, out of the total number of failures t that are tolerated by the algorithm. (In this paper we consider only crash-stop failures.)

From a practical perspective, what we might sometimes want to measure and optimize is the number of rounds needed for *at least one* correct process to decide, i.e., for a *local decision.* Indeed, a replicated service can respond to its clients as soon as a single replica decides on a reply (and knows that other replicas will reach the same decision). Similarly, the client of an atomic commit service might be happy to know the outcome of a transaction once the outcome has been determined, even if some database servers have yet to be informed of the outcome.

Motivations. Surprisingly, despite the large body of work on the performance of agreement, so far, no study on local decision lower bounds has appeared in the literature. To get an intuition of some of the specific ramifications underlying such a study, consider the consensus problem [27, 22] in the synchronous model, where a set of processes, $\{p_1, p_2, \ldots, p_n\}$, proceeds by exchanging messages in a round by round manner, and t out of the n processes may fail by crashing [23].

In this problem, the processes must decide on a common final value, out of the values they initially proposed, such that all correct processes eventually decide and agree on a common decision. The following algorithm (from [16]) conveys the fact that there can indeed be a difference between local and global decision lower bounds. (Round numbers start from 1.) At the beginning of round 1, process p_1 decides on its proposal value and then sends its decision value to all processes. At the end of every round $i \geq 1$, process p_{i+1} decides on the value contained in the last received message, and if p_{i+1} has not received any message, p_{i+1} decides on its proposal value. Process p_{i+1} then sends its decision value to all processes in round $i+1$. (The correct process with the lowest id, say, p_j, succeeds in sending its decision value to all processes in round j. Subsequently, all processes with higher ids decide and propagate the decision value of p_j.) If there are no failures, i.e., $f = 0$, then p_1 decides before sending any message in round 1, and we say that p_1 decides in round 0. In runs of this algorithm with at most $1 \leq f \leq t$ failures, at least one correct process decides by round f. Hence, if we denote by l_f the tight local decision lower bound for consensus in runs of the synchronous model with f failures, the very existence of the algorithm means that $l_f \leq f$. In fact, a closer look reveals that l_f is exactly f. However, if we denote by g_f the tight global decision lower bound, we know from [21] that g_f is exactly $f + 1$. This observation opens several questions.

1. Can we match both lower bounds with the *same* algorithm? The synchronous consensus algorithm we just sketched matches the lower bound $l_f = f$ but clearly does not match the lower bound $g_f = f + 1$. Is there any other algorithm that does so? Otherwise, we would be highlighting a rather interesting trade-off in the design of consensus algorithms.

2. What is the impact of the very nature of the agreement?

(i) Consider, for instance, the *uniform* variant of consensus [18], where no process disagrees with any other process, even one that crashed. Clearly, the algorithm sketched above needs to be revisited. We can easily exhibit a uniform consensus algorithm in which at least one correct process decides by round $f + 1$, in runs with at most f failures; i.e., $l_f \leq f + 1$. Additionally, we know from [4, 19] that, for most values of f, $g_f = f + 2$. Is $g_f = l_f + 1$?

(ii) Similarly, consider the nonblocking atomic commit problem [29, 18], where the processes have to decide 0 if some process proposes 0, and have to decide 1 if no process proposes 0 or crashes. We know that the tight global decision lower bound for atomic commit is the same as for uniform consensus [3, 10]. But, do the two problems have the same tight local decision bounds as well?

3. What is the impact of the model? Consider consensus, for instance, in the eventually synchronous model [11]. If we compare (a) the number of rounds g_f^{es} needed for all correct processes to decide in *synchronous* runs with f process crashes, and (b) the number of rounds l_f^{es} needed for at least one correct process to decide in such runs, is $g_f^{es} = l_f^{es} + 1$?

Contributions. 1.We show in the synchronous model that, except for some specific values of f (which we make precise in the paper), $g_f = l_f + 1$ for consensus, uniform consensus, and nonblocking atomic commit. (In fact, to exhibit a matching algorithm for uniform consensus and nonblocking atomic commit, we give an algorithm for the "stronger" interactive consistency problem [27].) Furthermore, we highlight an interesting trade-off in the design of consensus algorithms by showing that no consensus algorithm can match both global and local decision bounds. More precisely, no consensus algorithm can match both l_{f+1} and g_f. In addition, we show that, for the failure-free case (i.e., $f = 0$) of nonblocking atomic commit, the local decision bound is higher than that of uniform consensus. Since both problems have identical global decision lower bounds [3, 10], we draw the first line between their time-complexity.

2. We also consider uniform consensus in the eventually synchronous model. (In this model, nonblocking atomic commit is not solvable when $t \geq 1$, and consensus is equivalent to uniform consensus [16].) We determine a local decision lower bound of $f + 2$ rounds for synchronous runs of the model, with f failures (for $f \leq t - 3$). Then we present an algorithm that, in synchronous runs with f failures, globally (and hence locally) decides in $f + 2$ rounds. In addition to matching the local decision bound, to our knowledge, our algorithm is the first to match the $f + 2$ rounds global decision lower bound presented in [4, 19, 9]. In other words, we show that, for synchronous runs of the eventually synchronous model, tight local decision bounds are the same as for global decision; i.e., $g_f^{es} = l_f^{es} = f + 2$.

Related work. The consensus problem was introduced in [27, 22] and (nonblocking) atomic commit was defined in [15, 29]. The distinction between consensus and uniform consensus, and the relationship with the atomic commit problem were discussed in [18, 16]. Initial lower bound results on the time-complexity of agreement problems were proved in [13] and studied further in [21, 7, 25, 12, 1]. The eventually synchronous model was introduced in [5, 11].

In the synchronous model, one of the initial early halting agreement algorithms was presented in [21]. The early halting lower bound for consensus was proved in [6]. The early decision lower bound for uniform consensus and its difference from the nonuniform case were studied in [4, 19].

In the eventually synchronous model, the first consensus algorithm was presented in [11]. The equivalence between consensus and uniform consensus in the eventually synchronous model was shown in [16]. Tight bounds for synchronous runs of the eventually synchronous model in the failure-free case ($f = 0$) were shown in [19, 28, 26] and in the worst-case ($f = t$) were shown in [9]. Techniques that use forward inductions to prove lower bounds on agreement problems were introduced in [24, 1].

Roadmap. Section 2 recalls the models we consider. Section 3 recalls the definitions of the agreement problems we study. In section 4, we introduce the definition of our local decision metric, and we recall other time-complexity metrics. We also devise a compact notation for presenting various lower bound results on agreement problems. Section 5 recalls the layering technique of [24], also used in [19], which we slightly extend to prove local decisions results. Sections 6 and 7 present our lower bound results and matching algorithms in the synchronous model, respectively. The lower bound results for the eventually synchronous model and the matching algorithm are presented in sections 8 and 9, respectively. Section 10 concludes the paper.

2. Models. The distributed system we consider consists of a set of $n \geq 3$ processes, denoted by $\Pi = \{p_1, p_2, \ldots, p_n\}$, that communicate by message-passing: every pair of processes is connected by a bidirectional communication channel that does not create, duplicate, or alter messages. However, messages may be lost or reordered. The processes may fail by crashing and do not recover from a crash. The computation proceeds in rounds of message-exchange with round numbers starting from 1 and increasing by 1 in every round. A distributed algorithm A is a collection of deterministic automata, where the automaton for each process executes the following two phases in every round: (a) in the *send phase*, the processes send messages to all processes; (b) in the *receive phase*, the processes receive some messages sent in the send phase (of the current round or of a lower round) and update local states (which might include a decision event). A *run* of algorithm A is an infinite sequence of rounds of A. A *partial run* is a finite prefix of some run. A (partial) run r *extends* some partial run pr if pr is a prefix of r. A process that does not crash in a run is said to be *correct* in that run; otherwise the process is *faulty.* We say that a message m sent in a run is *lost* (in that run) if m is never received in that run.

In the distributed system described above, a *model* is a set of runs selected by restricting when processes can crash and specifying which messages are received. A *submodel* of a model M is a model that is a subset of M. We consider the following models.

- For every t such that $0 \leq t \leq n-1$, we define the t-resilient synchronous crash-stop model [23], denoted SCS_t, as follows. In every given run of SCS_t, the following properties hold: (1) if a process starts some round k then it either completes that round or crashes; (2) at most t processes crash; and (3) in round k, if p_i completes the send phase of the round, then every process that completes the receive phase of the round receives in that phase the round k message sent by p_i. (If p_i crashes in the send phase of round k, then there are no delivery guarantees—an arbitrary subset of messages sent by p_i in round k may be lost.)
- For every t such that $0 \leq t \leq n-1$, we define $\mathrm{SCS1}_t$ as the submodel of SCS_t that contains those runs of SCS_t in which at most one process crashes in a round.
- For every t such that $0 \leq t \leq n-1$, we define the t-resilient eventually synchronous crash-stop model, denoted ES_t, as follows. In ES_t, the runs may be "asynchronous" for an arbitrary yet finite number of rounds but eventually become "synchronous." A message sent in the "asynchronous period" may be *delayed* for a finite number of rounds, i.e., received in a round higher than the round in which it was sent. More precisely, in every given run of ES_t, the following properties hold: (1) if a process starts some round k then it either completes that round or crashes; (2) (*t-resilience*) at most t processes crash, and every process that completes any round k receives in that round the round k messages from at least $n-t$ processes; (3) (*reliable channels*) every message sent by a correct process to a correct process in any round k

is received in round k or in a higher round; (4) (*eventual synchrony*) there is an unknown but finite round number GSR (global stabilization round) such that, in every round $k \geq GSR$, if p_i completes the send phase of the round k, then every process that completes the receive phase of the round receives in that phase the round k message sent by p_i. (If p_i crashes in the send phase of round k, then, similar to SCS_t, there are no delivery guarantees; an arbitrary subset of messages sent by p_i in round k may be lost.) Also, we say that the run is *synchronous from round GSR*.

Observe that, for every $0 \leq f \leq t \leq n-1$, SCS_f is a submodel of SCS_t, and ES_f is a submodel of ES_t. Furthermore, every run of SCS_t is a run of ES_t with $GSR = 1$. Thus, SCS_t is a submodel of ES_t.

Hereafter, we make a slight change in terminology: instead of saying that there is a unique synchronous model, we say that each of the $2n$ models SCS_t and $SCS1_t$ $(0 \leq t \leq n-1)$ is a different synchronous model (i.e., there is no unique synchronous model). Similarly, we say that each of the n models ES_t $(0 \leq t \leq n-1)$ is a different eventually synchronous model.

3. Agreement problems. We consider three agreement problems: consensus, uniform consensus, and nonblocking atomic commit.

- In the (nonuniform) *consensus* problem [22], denoted NC, the processes start with a proposal value and eventually decide on a final value such that the following properties are satisfied: (validity) if a process decides v, then some process has proposed v; (agreement) no two correct processes decide differently; and (termination) every correct process eventually decides.
- *Uniform consensus* [18], denoted UC, is a variant of consensus in which the agreement property is replaced by the following *uniform agreement* property: no two processes decide differently.
- In the *nonblocking atomic commit* problem [15, 29], denoted NBAC, each process casts a vote of whether to abort or commit a transaction, and eventually decides. The termination and the uniform agreement properties are the same as that for uniform consensus. Validity is defined in two parts: (abort validity) abort can be decided only if some process proposes to abort or fails, and (commit validity) commit can be decided only if all processes propose to commit. For presentation uniformity, we make the following changes in notation: (1) we say that a process proposes 0 (resp., 1) if the process votes abort (resp., commit), and (2) we say that a process decides 0 (resp., 1) if the process decides to abort (resp., to commit).

To prove our lower bounds, we consider variants of consensus and uniform consensus. We define the *weak binary agreement* problem, denoted WA, where the processes are allowed to propose either 0 or 1. WA satisfies the agreement and termination properties of consensus, and the following weak validity property (from [19]): for every value $v \in \{0, 1\}$, there is a failure-free run in which correct processes decide v. The *weak binary uniform agreement* problem, denoted UA, is identical to WA except that it also satisfies uniform agreement (no two processes decide differently).

Clearly, any NC, UC, or NBAC algorithm can solve WA without any additional communication. Thus, our time-complexity lower bounds on WA immediately apply to the three agreement problems. Similarly, any UC or NBAC algorithm can solve UA without any additional communication, and hence, our time-complexity lower bounds on UA immediately apply to UC and NBAC problems.

In the synchronous models, we present the matching algorithms for uniform consensus and nonblocking atomic commit by first devising an interactive consistency algorithm, which we then transform to consensus and nonblocking atomic commit algorithms. In the *interactive consistency* problem [27], denoted IC, each process proposes an initial value and eventually decides on a vector of values. Termination and agreement properties are the same as for uniform consensus. Validity is defined as follows: for every decision vector V, the jth component of V is either the value proposed by p_j or $\perp$, and may be $\perp$ only if p_j fails.

4. Time complexity metrics. Let r be any run of an algorithm that solves one of the agreement problems described in section 3. We say that a process p_i *decides* in round $k \geq 1$ in r if p_i decides in the receive phase of round k, and a process decides at round 0 if it decides before sending any message in round 1. We say that a process *halts* in round k in r if it does not crash by round k, and does not take any step after round k.

We distinguish four different time complexity metrics for runs of agreement algorithms: *global decision*, *global halting*, *local decision*, and *local halting.* Consider any run r of an algorithm that solves an agreement problem.

- We say that run r *globally decides* (resp., *globally halts*) in round k if all correct processes decide (resp., halt) in round k, or in a *lower* round, and some correct process decides (resp., halts) in round k [13, 6, 4, 19].
- We say that run r *locally decides* (resp., *locally halts*) in round k if all correct processes decide (resp., halt) in round k, or in a *higher* round, and some correct process decides (resp., halts) in round k.

We introduce the following notation. If a run r globally decides at round k, we write $(r, gd) = k$. Similarly, the round at which run r globally halts, locally decides, and locally halts are denoted by (r, gh), (r, ld), and (r, lh), respectively. Note that, since every correct process decides before it halts, $(r, ld) \leq (r, lh)$, and $(r, ld) \leq (r, gd) \leq (r, gh)$. Given a model M1, a submodel M2 of M1, an agreement problem P, and a time complexity metric T, we denote by the ordered tuple (M1, M2, P, T) the following tight bound. (M1, M2, P, T) is the round number k such that (1) (lower bound) every algorithm that solves P in M1 has a run r in M2 such that $(r, T) \geq k$, and (2) (matching algorithm) there is an algorithm *Alg* that solves P in M1 such that every run r of *Alg* in M2 has $(r, T) \leq k$.

In other words, for algorithms that solve problem P in model M1, (M1, M2, P, T) is the tight bound for achieving T in submodel M2. The notation captures the common time-complexity tight bounds for agreement problems, where submodel M2 denotes the set of runs (e.g., failure-free runs) for which we want to optimize the algorithms in M1. If we set M2 = M1, the tuple denotes the worst-case bound in M1.

Before delving into our lower bounds, we recall some known results on consensus (NC) and uniform consensus (UC) using our notation. (For every pair of reals $a \leq b$, $[a, b]$ denotes the set of integers x such that $a \leq x \leq b$; when $a > b$, $[a, b]$ denotes the emptyset.)

- $\forall t \in [0, n-2]$, $(SCS_t$, SCS_t, NC, $gd) = t + 1$. Every consensus algorithm in SCS_t has a run (in SCS_t) in which some correct process decides in round $t+1$ or in a higher round, and there is a consensus algorithm A in SCS_t such that, in every run of A (in SCS_t), every correct process decides by round $t+1$ [13, 23].
- $\forall t \in [2, n-2]$, $\forall f \in [0, t-1]$, $(SCS_t$, SCS_f, NC, $gh) = f+2$. Every consensus algorithm in SCS_t has a run in SCS_f in which some correct process halts in

round $f+2$ or in a higher round, and there is a consensus algorithm A in SCS_t such that, in every run of A in SCS_f, every correct process halts by round $f+2$ [6].

- $\forall t \in [2, n-1]$, $\forall f \in [0, t-2]$, (SCS_t, SCS_f, UC, gd) $= f+2$. Every uniform consensus algorithm in SCS_t has a run in SCS_f in which some correct process decides in round $f+2$ or in a higher round, and there is a uniform consensus algorithm A in SCS_t such that, in every run of A in SCS_f every correct process decides by round $f+2$ [4, 19].
- $\forall t \in [1, (n-1)/2]$, ($ES_t$, SCS_t, NC, gd) $= t+2$. Every consensus algorithm in ES_t has a run in SCS_t in which some correct process decides in round $t+2$ or in a higher round, and there is a consensus algorithm A in ES_t such that, in every run of A in SCS_t every correct process decides by round $t+2$ [9].

Roughly speaking, in this paper we investigate tight bounds when the time-complexity metric is local decision (ld). In particular, we determine (SCS_t, SCS_f, UC, ld), (SCS_t, SCS_f, NC, ld), and (ES_t, SCS_f, UC, ld).

5. Layering.

Our lower bound proofs are devised following the layering technique of [24], which is also used in [19]. We first introduce some definitions and then recall the notion of layering from [24, 19]. We then present two lemmas (that are slightly modified from [19]) from which we derive our lower bound results. (In the following, we point out when our notions differ from those in [19].)

5.1. Configurations and extensions.

Consider a model M and an agreement algorithm A devised in M. For each run r of algorithm A in model M, we denote by $val(r)$ the decision value of any correct process in r. (This definition is unambiguous because, in every agreement problem we consider, no two correct processes decide differently.) For a run r of A in M we define the *configuration* C at the end of round k (also called *round k configuration*), as an ordered tuple of size $n+n^2$, where the element i, for $1 \leq i \leq n$, is the state of process p_i at the end of round k in run r, and the rest of the elements contain the set of delayed messages in the n^2 communication channels at the end of round k in run r. (Since there are no delayed messages in synchronous models, the channels are empty at the end of every round. Hence, in a synchronous model, we ignore state of channels in configurations at the end of a round.) The state of a process that has crashed is denoted by the special symbol $\perp$. We say that a process p_i is *alive* in a configuration if p_i has not crashed in that configuration. In the *initial configuration* (which we also call round 0 configuration) of run r, the state of each process is its proposal value, and the state of every communication channel is the emptyset $\emptyset$.

Given a round k configuration C of algorithm A in model M, we define the following concepts. A run r of algorithm A in model M is an *extension* of the round k configuration C if the round k configuration of run r is C. A round k_1 configuration C′ of algorithm A in model M is an *extension* of the round k configuration C if $k \leq k_1$ and there is a run r of A in M such that the round k configuration of r is C and round k_1 configuration of r is C′. If M is a synchronous model, we denote by $r(C)$ the run which is an extension of C such that no process crashes after round k. We define $val(C)$ as $val(r(C))$. Observe that a process p_i is alive in C if and only if p_i is correct in $r(C)$.

5.2. Layering in synchronous models.

In this subsection, we consider any given weak binary agreement (WA) algorithm A in model $SCS1_t$. (See section 2 and section 3 for a definition of SCS1$_t$ and WA, respectively.)

Extensions in $\mathbf{SCS1_t}$. A run of algorithm A is completely defined by its initial configuration and its failure pattern. (The failure pattern for a run in $SCS1_t$ consists, for each round k, of the process p_i that crashes in round k and the set of processes that did not receive the round k message from p_i.) In model $SCS1_t$, we denote an *extension by one round*, of a round k configuration C, as follows: for $1 \leq i \leq n$ and $S \subseteq \Pi$, $C.(i, S)$ denotes the round $k+1$ configuration reached by crashing p_i in round $k+1$ such that a process p_j *does not* receive a round $k+1$ message from p_i if at least one of the following holds: (1) $p_j = p_i$, (2) p_j is crashed in C, or (3) $p_j \in S$. Configuration $C.(0, \emptyset)$ denotes the one round extension of C in which no process crashes. Clearly, $C.(i, S)$ for $i > 0$ and $S \subseteq \Pi$ is a possible extension of C if at most $t - 1$ processes have crashed in C and p_i is alive in C. We then say that (i, S) is *applicable* to C. Configuration $C.(0, \emptyset)$ is always applicable to C.

Layers. A layer $L(C)$ is the set of configurations defined as $\{C.(i, S) | i \in \Pi, S \subseteq \Pi, (i, S) \text{ is applicable to } C\}$. (In other words, if C is a round k configuration, then $L(C)$ is the set of all round $k+1$ configurations that extend C in $SCS1_t$.) For a set of round k configurations SC, $L(SC)$ is a set of round $k+1$ configurations defined as $\cup_{C \in SC} L(C)$. $L^k(SC)$ is recursively defined as follows: $L^0(SC) = SC$ and for $k > 0$, $L^k(SC) = L(L^{k-1}(SC))$. (In other words, if SC is a set of round l configurations then $L^k(SC)$ is the set of all round $(l + k)$ configurations that extend any configuration in SC.)

Similar configurations. Consider a set of round k configurations SC. Two configurations C and D in SC are *similar*, denoted $C \sim D$, if they are identical or they differ at exactly one process. A pair of configurations C and D in SC is *similarity connected* if there are configurations $C = C_0, \ldots, C_m = D$ in SC such that $C_i \sim C_{i+1}$ for every i such that $0 \leq i \leq m - 1$. The set SC is *similarity connected* if every pair of configurations in SC is similarity connected. (Our definition of *similarity* does not include the second requirement in the original definition of [19]: there exists a process that is alive in both C and D and has identical states in C and D. When this property is required in our lower bound proofs, we derive it directly from our assumption on t and n.)

We now revisit Lemma 2.3 of [19]. Roughly speaking, this lemma says that, in $SCS1_t$, if we start with a similarity connected set SC of configurations, we can keep the set of extensions from SC similarity connected, provided we can crash one process in every round.

LEMMA 5.1. *In $SCS1_t$, let $SC = L^0(SC)$ be a similarity connected set of configurations such that in every configuration of SC no process has crashed. Then for all $k \in [1, t]$, $L^k(SC)$ is a similarity connected set of configurations in which no more than k processes have crashed in any configuration.*

Proof. The proof is by induction on round number k. The base case $k = 0$ is immediate. For the inductive step, assume that $L^{k-1}(SC)$ is similarity connected and in every configuration of $L^{k-1}(SC)$ at most $k-1$ processes have crashed. Notice that, in every extension by one round that is applicable to a configuration in $L^{k-1}(SC)$, at most one more process can crash. Therefore, in every configuration in $L^k(SC)$ at most k processes have crashed. We now show that $L^k(SC)$ is similarity connected through the following three claims.

1. *For every configuration $C \in L^{k-1}(\text{SC})$, $L(C)$ is similarity connected.* Consider any configuration in $L(C)$ that is different from $C.(0, \emptyset)$, say, $C1 = C.(i, Q)$, where $Q \subseteq \Pi$, and p_i is alive in C. We claim that $C1$ and $C.(0, \emptyset)$ are similarity connected. Since $C1$ is arbitrarily selected from $L(C)$, our claim implies that every

configuration in $L(C)$ is similarity connected to $C.(0, \emptyset)$, and hence, $L(C)$ is similarity connected.

Now we prove our claim. $C.(i, \emptyset) \sim C.(0, \emptyset)$ since the configurations differ only at p_i. If $Q = \emptyset$ then we are done. Hence, let $Q = \{q_1, q_2, \ldots, q_m\}$. For every l in $[1, m]$, let $Q_l = \{q_1, \ldots, q_l\}$, and $Q_0 = \emptyset$. For every l in $[0, m-1]$, $C.(i, Q_l) \sim C.(i, Q_{l+1})$ because the two configurations differ only at q_{l+1}. Thus, $C(i, \emptyset) = C.(i, Q_0)$ and $C1 = C.(i, Q_m)$ are similarity connected.

2. *For every pair of configurations* $C, D \in L^{k-1}(\mathrm{SC})$*, if* $C \sim D$ *then* $L(C) \cup L(D)$ *is similarity connected.* If C and D are identical then the claim immediately follows from claim 1. So consider the case where C and D are distinct. As $C \sim D$, there is a process p_i such that C and D are different only at p_i. Then, configurations $C.(i, \Pi)$ and $D.(i, \Pi)$ are identical because no process receives message from p_i in round k, and p_i has crashed. Hence, $C.(i, \Pi) \sim D.(i, \Pi)$. We know from claim 1 that $L(C)$ and $L(D)$ are each similarity connected. Thus every configuration in $L(C)$ is similarity connected to $C.(i, \Pi)$ and every configuration in $L(D)$ is similarity connected to $D.(i, \Pi)$. As, $C.(i, \Pi) \sim D.(i, \Pi)$, so every configuration in $L(C)$ is similarity connected to every configuration in $L(D)$. Thus, $L(C) \cup L(D)$ is similarity connected.

3. $L^k(\mathrm{SC})$ *is similarity connected.* Consider any pair of configurations $C', D' \in L^k(SC)$. Thus, there are configurations $C, D \in L^{k-1}(SC)$ such that $C' \in L(C)$ and $D' \in L(D)$. As $L^{k-1}(SC)$ is similarity connected, there is a chain of configurations $C = C_0, \ldots, C_m = D$ such that, for every $l \in [0, m-1]$, $C_l \sim C_{l+1}$. Thus, from claim 2, $L(C_l) \cup L(C_{l+1})$ is similarity connected. A simple induction shows that $L(C_1) \cup \ldots \cup L(C_m)$ is similarity connected. Thus $C' \in L(C = C_0)$ is similarity connected to $D' \in L(D = C_m)$. As C' and D' are arbitrarily selected from $L^k(SC)$, $L^k(SC)$ is similarity connected. □

Remarks. The above lemma is a simple generalization of Lemma 2.3 of [19]. The statement of the lemma is similar; however, the proof is slightly different because our model $SCS1_t$ is slightly different from that of [19]. Their model is actually a submodel of $SCS1_t$. Consider any crashed process p_i, and the set of processes J to which messages from p_i were lost in the round in which p_i crashed. Then, in the model of [19], J is allowed only to be a prefix of processes $\{p_1, \ldots, p_k\}$, whereas in $\mathrm{SCS1}_t$, J is allowed to be any subset of Π.

Informally, the next lemma says that, for any WA algorithm in $SCS1_t$, there are two round f configurations that are almost identical (they differ at only one process) but have different decision values in failure-free extensions.

Recall that, for any configuration y in a synchronous model, $val(y)$ is the decision value of correct processes in a run which extends y and has no crashes after y.

LEMMA 5.2. *Consider any WA algorithm A in* SCS_t *such that* $t \in [1, n-1]$*. For every* $f \in [0, t]$*, there are two runs of A in* $SCS1_t$ *such that their round* f *configurations,* y *and* y'*, satisfy the following:* (1) *at most* f *processes have crashed in each configuration,* (2) *the configurations differ at exactly one process, and* (3) $val(y) = 0$*, whereas* $val(y') = 1$*.*

Proof. Consider any WA algorithm A in SCS_t. We claim that A solves WA in $SCS1_t$ as well. A maintains the agreement and termination properties in all runs of $SCS1_t$ because every run in $SCS1_t$ is also a run in SCS_t. The weak validity property is bit different—it is a condition on the set of failure-free runs. However, observe that the SCS_t and $SCS1_t$ have the same set of failure-free runs. It follows that if A satisfies the weak validity property in SCS_t, then A also satisfies the property in $SCS1_t$. Thus, A solves WA in $SCS1_t$.

Now consider WA algorithm A in $SCS1_t$. Let C$'$ be any initial configuration of algorithm A and C be the initial configuration in which all processes propose 0. Consider the following $n-1$ (not necessarily distinct) initial configurations: for every i in $[1, n-1]$, in configuration C_i, processes p_1 to p_i propose the same value as in C', and the remaining processes propose 0. Notice that, for every i in $[1, n-2]$, C_i and C_{i+1} may differ only at p_{i+1}. Furthermore, C_1 and C may differ only at p_1, and C' and C_{n-1} may differ only at p_n. Thus C and C' are connected through a chain of configurations, such that any two adjacent configurations in the chain are similar. Since C' was arbitrarily selected, the set of initial configurations of A in $SCS1_t$ is similarity connected. From Lemma 5.1 it follows that the set of round f configurations of A in $SCS1_t$ is similarity connected.

Consider any failure-free run $r0$ of algorithm A in which correct processes decide 0. (From the validity property of WA, such a run of A exists.) We denote by z the round f configuration of $r0$. Similarly, consider any failure-free run $r1$ of A in which correct processes decide 1. We denote by z' the round f configuration of $r1$. Obviously, $val(z) = 0$ and $val(z') = 1$.

As the set of round f configurations of A in $SCS1_t$ is similarity connected, there are some round f configurations of A in $SCS1_t$, $z = y_0, y_1, \ldots, y_m = z'$, such that $y_j \sim y_{j+1}$ for every j in $[0, m-1]$. Clearly, there is some $y_i \in \{y_0, \ldots, y_{m-1}\}$ such that $val(y_0) = \cdots = val(y_i) \neq val(y_{i+1})$. (Otherwise, $val(z) = val(y_0) = val(y_1) = \cdots = val(y_m) = val(z')$, which is a contradiction.)

As $val(y_i) = val(y_0)$ and $y = y_0$, $val(y_i) = 0$. Therefore, $val(y_{i+1}) = 1$. Since both y_i and y_{i+1} are round f configurations in $SCS1_t$, at most f processes have crashed in each configuration. As $y_i \sim y_{i+1}$, the two configurations either are identical or differ at exactly one process. Since $val(y_i) \neq val(y_{i+1})$, the configurations cannot be identical; i.e., they differ at exactly one process. □

6. Synchronous lower bounds.

6.1. Consensus. In the following we show a local decision lower bound for weak binary agreement (WA) in synchronous models (SCS_t with $1 \leq t \leq n-1$). We then show the impossibility of simultaneously matching both local decision and global decision lower bounds of WA. Since any consensus (NC) algorithm solves WA, the results immediately apply to consensus.

We observe that every run of an algorithm in $SCS1_t$ is also a run in SCS_t. Thus, Lemma 5.2 holds when $SCS1_t$ is replaced by SCS_t.

Local decision. The following proposition states that any WA algorithm in SCS_t has a run in SCS_f (i.e., a run with at most f crashes) in which every correct process decides in round f or in a higher round.

PROPOSITION 6.1. *For all* $t \in [1, n-1]$, *for all* $f \in [0, t]$, (SCS_t, SCS_f, WA, *ld*) $\geq f$.

Proof. Suppose by contradiction that there is a WA algorithm A in SCS_t and an integer f in $[0, t]$ such that, in every run of A with f failures, some correct process decides by round $f-1$. Notice that the contradiction is immediate for the case $f = 0$: no process can decide by round -1. So we consider the case $f \in [1, t]$. (Also recall that we define deciding at round 0 as deciding before sending any message in round 1.)

It follows from Lemma 5.2 that there are two runs of A in SCS_t such that their round $f-1$ configurations, y and y', satisfy the following: (1) at most $f-1$ processes have crashed in each configuration, (2) the configurations differ at exactly one process, say, p_i, and (3) $val(y) = 0$ and $val(y') = 1$.

As $r(y)$ is a run with at most $f-1$ crashes, it follows from our assumption on A that, in $r(y)$, there is a correct process q_1 that has decided $val(y) = 0$ by round $f-1$. As all correct processes in $r(y)$ are alive in y, it follows that, in y, q_1 is alive and has decided $val(y) = 0$.

We now show that no alive process distinct from p_i has decided in y (which implies $p_i = q_1$). Suppose by contradiction that some alive process distinct from p_i, say, q_2, has decided in y. Since q_2 is alive in y, it is correct in $r(y)$, and hence, q_2 has decided $val(y) = 0$ in y. As y and y' differ only at p_i, and p_i is distinct from q_2, q_2 is alive and has decided 0 in y'. Thus, in $r(y')$, q_2 is a correct process and decides 0. However, every correct process in $r(y')$ decides $val(y') = 1$, which is a contradiction.

Thus, p_i is the only alive process that has decided in y. Consider any run r' that extends y and in which only process p_i crashes after round $f-1$. At most f processes crash in r'. At the end of round $f-1$ in r', the only alive process that has decided is p_i, but p_i is a faulty process in r'. Thus, r' is a run with f failures in which no correct process decides by round $f-1$, which is a contradiction. □

Incompatibility. It is easy to design a consensus algorithm that matches either the early local decision or the early global decision lower bound. We now show that, maybe surprisingly, no consensus algorithm can match both the early local decision and the early global decision lower bounds, even for two consecutive values of f. This is in contrast to uniform consensus, where a single algorithm can match both local decision and global decision lower bounds (as we show in section 7).

PROPOSITION 6.2. *For all $t \in [1, n-2]$, for all $f \in [0, t-1]$, there is no WA algorithm in SCS_t that matches the following two conditions:* (a) *in every run with at most f crashes, every correct process decides by round $f+1$, and* (b) *in every run with at most $f+1$ crashes, some correct process decides by round $f+1$.*

Remarks. Condition (a) is for matching the global decision lower bound for f crashes, and condition (b) is for matching the local decision lower bound for $f+1$ crashes. Note that, we do not consider the case $f = t$, because when $f = t$, (a) implies (b), as there is no run in SCS_t with $t+1$ crashes.

Proof. Suppose by contradiction that there is a WA algorithm A in SCS_t and an integer f in $[0, t-1]$ such that (a) by round $f+1$ of every run with at most f failures, every correct process decides, and (b) by round $f+1$ of every run with at most $f+1$ failures, some correct process decides.

It follows from Lemma 5.2 that, at the end of round f there are two configurations y_0 and y_1 such that (a) at most f processes have crashed in each configuration, (b) the configurations differ at exactly one process, say, p_i, and (c) $val(y_0) = 0$ and $val(y_1) = 1$.

Consider run $r(y_0)$. Obviously, $r(y_0)$ is a run with at most f failures, and from our initial assumption, every correct process decides $val(y_0) = 0$ at the end of round $f+1$. Similarly, we construct run $r(y_1)$, which is a failure-free extension of y_1, and every correct process decides $val(y_1) = 1$ at the end of round $f+1$. There are two cases to consider.

Case 1. Process p_i is alive in y_0 and y_1. Consider the extension of y_0 to a run $r'(y_0)$ such that p_i crashes in round $f+1$ before sending any message, and no process crashes thereafter. (Recall that $f \leq t-1$.) Notice that $r'(y_0)$ is a run with at most $f+1$ failures and p_i is a faulty process in $r'(y_0)$. Thus, from our initial assumption about A, it follows that there is a correct process $p_j(\neq p_i)$ in $r'(y_0)$ which decides some value $v \in \{0, 1\}$ at round $f+1$. (Notice that, since $p_j \neq p_i$, p_j cannot decide before round $f+1$: as y_0 and y_1 differ only at p_i, if p_j decides by round f, then p_j

decides identical values in y_0 and y_1.) Also, as $f \leq n-3$, there is a process p_l distinct from p_i and p_j such that p_l decides 0 and 1 at the end of round $f+1$ in $r(y_0)$ and $r(y_1)$, respectively.

Now we construct a run r'' by extending configuration y_{1-v}: process p_i crashes in the send phase of round $f+1$ such that, in round $f+1$, p_l receives a message from p_i but p_j does not receive any message from p_i. No process distinct from p_i crashes in round $f+1$ or a higher round. Obviously, p_j and p_l are correct in r''. At the end of round $f+1$ in run r'', p_j cannot distinguish r'' from $r'(y_0)$ because the round f configurations of the two runs differ only at p_i, and p_j does not receive any round $f+1$ message from p_i in both runs. Therefore, p_j decides v at the end of round $f+1$ in r''. However, since p_l receives a message from p_i in round $f+1$, at the end of round $f+1$, p_l cannot distinguish r'' from $r(y_{1-v})$, and therefore, decides $1-v$ at the end of round $f+1$, which is a contradiction to the agreement property of WA.

Case 2. Process p_i has crashed in either y_0 or y_1. Without loss of generality, we can assume that p_i has crashed in y_0, and hence, p_i is alive in y_1. (Recall that p_i has different states in the two configurations.) As at most f processes, including p_i, have crashed in y_0, and p_i has not crashed in y_1, it follows that at most $f-1$ processes have crashed in y_1. Since $f \leq n-3$ and at most $f-1$ processes have crashed in y_1, there are at least two correct process p_j and p_l (both distinct from p_i) in $r(y_1)$. Consider the run r' which extends y_1 such that process p_i crashes in round $f+1$ and the only alive process that *does not* receive the round $f+1$ message from p_i is p_l, and no process crashes after round $f+1$. Obviously p_j and p_l are correct in r'. At the end of round $f+1$, p_l cannot distinguish $r(y_0)$ from r' because p_l does not receive the round $f+1$ message from p_i in both runs. Thus, p_l decides 0 at the end of round $f+1$ in r'. At the end of round $f+1$, p_j cannot distinguish $r(y_1)$ from r' because both runs extend y_1 and p_j receives round $f+1$ message from p_i in both runs. Thus, p_j decides 1 at the end of round $f+1$ in r', which is a contradiction to the agreement property of WA. □

6.2. Uniform consensus. In the following, we show a local decision lower bound for weak binary uniform agreement (UA) in the synchronous models (SCS_t with $1 \leq t \leq n-1$). Since any uniform consensus (UC) and nonblocking atomic commit (NBAC) algorithm solves UA, the lower bound immediately applies to UC and NBAC. In section 6.3, we show that the lower bound holds for IC as well.

The following proposition says that any UA algorithm in SCS_t has a run in SCS_f (i.e., a run with at most f crashes) in which every correct process decides in round $f+1$ or in a higher round.

We observe that any UA algorithm also solves WA, and every run of an algorithm in $SCS1_t$ is also a run in SCS_t. Thus, Lemma 5.2 holds when WA and $SCS1_t$ are replaced by UA and SCS_t, respectively.

Proposition 6.3. *For all* $t \in [1, n-1]$*, for all* $f \in [0, t-1]$*,* (SCS_t*,* SCS_f*, UA, ld*) $\geq f+1$.

Proof. Suppose by contradiction that there is a UA algorithm A in SCS_t and an integer f in $[0, t-1]$ such that, in every run of A with f failures, some correct process decides by round f.

As every UA algorithm solves WA, it follows from Lemma 5.2 that there are two runs of A in SCS_t such that their round f configurations, y and y', satisfy the following: (1) at most f processes have crashed in each configuration, (2) the configurations differ at exactly one process, say, p_i, and (3) $val(y) = 0$ and $val(y') = 1$.

From our initial assumption about algorithm A, it follows that there is an alive process q_1 in y that has already decided. (Otherwise, since every correct process in $r(y)$ is an alive process in y, $r(y)$ is a run with at most f crashes in which no correct process decides by round f.) Furthermore, q_1 has decided $val(y) = 0$ in $r(y)$ (and hence, in y) because q_1 is a correct process in $r(y)$. Similarly, in y', there is an alive process q_2 that has decided $val(y') = 1$. There are two cases to consider.

(1) $q_1 \neq p_i$: As y and y' are identical at all processes different from p_i, in y', q_1 is alive and has decided 0. Thus in $r(y')$, q_1 is a correct process and decides 0. However, in $r(y')$ every correct process decides $val(y') = 1$, which is a contradiction.

(2) $q_1 = p_i$: We distinguish two subcases:

- $q_2 = p_i$: Thus $p_i = q_1 = q_2$, and hence, p_i is alive in y and y'. Consider a run $r1$ that extends y and in which p_i crashes in round $f + 1$ before sending any message. (Recall that $f \leq t-1$.) As p_i has decided 0 in y, it follows from the uniform agreement property that every correct process decides 0 in $r1$. Since $t < n$, there is at least one correct process, say, p_l, in $r1$. Now consider a run $r2$ that extends y' and in which p_i crashes in round $f + 1$ before sending any message. Notice that no correct process can distinguish $r1$ from $r2$: at the end of round f no alive process that is distinct from p_i can distinguish y from y', and p_i crashes before sending any message in round $f + 1$. Thus every correct process decides the same value in $r1$ and $r2$; in particular, p_l decides 0 in $r2$. However, $p_i = q_2$ decides 1 in $r2$, which is a contradiction to uniform agreement.
- $q_2 \neq p_i$: Then, q_2 has the same state in y and y'. Thus in y, q_2 is alive and has decided 1. In any run that extends y, $p_i = q_1$ has decided 0 and q_2 has decided 1, which is a contradiction to uniform agreement. □

6.3. Nonblocking atomic commit and interactive consistency. Recall that the local decision lower bound presented in section 6.2 holds for UC and NBAC. In the following, we show that for NBAC and IC, the local decision lower bound for the failure-free case ($f = 0$) can be shifted to 2. However, this result does not hold for UC: in section 7.4 we exhibit a UC algorithm that locally decides in 1 round in failure-free runs.

PROPOSITION 6.4. *For all* $t \in [2, n-1]$, $(\text{SCS}_t, \text{SCS}_0, \textit{NBAC, ld}) \geq 2$.

Proof. Suppose by contradiction that there is an NBAC algorithm A such that, in every failure-free run, some process decides in round 1. Let $C1$ be the initial configuration in which all processes propose 1. Consider the failure-free run $R1$ starting from $C1$; i.e., $R1 = r(C1)$. Suppose that some process p_i decides at the end of round 1. From the abort validity property of NBAC, we know that p_i cannot decide 0 (and hence, p_i decides 1) in $R1$.

Consider another run $R2$ starting from $C1$, but some process p_j ($\neq p_i$) crashes in round 1 and only p_i receives the round 1 message from p_j. Also, process p_i crashes in round 2, before sending the round 2 message to any process, and no process crashes thereafter. At the end of round 1, p_i cannot distinguish $R1$ from $R2$. Thus, p_i decides 1 in $R2$. From uniform agreement, we know that every process distinct from p_i and p_j decides 1. There exists at least one such process, say, p_l, because $t \leq n - 1$.

Let $C0$ be the initial configuration in which p_j proposes 0 and all other processes propose 1. Consider a run $R3$ starting from $C0$ with the same failure pattern as $R2$; i.e., p_j crashes in round 1 and only p_i receives the round 1 message from p_j, p_i crashes in round 2 before sending the round 2 message to any process, and no process crashes thereafter. No process distinct from p_i and p_j can distinguish $R2$ from $R3$: at the end of round 1, only p_i receives the message from p_j, but p_i crashes before sending any message in round 2. Therefore, every process distinct from p_i and p_j decides 1 (as in $R2$), particularly p_l. But the commit validity property of NBAC requires that no process decides 1 in $R3$ because some process p_j has proposed 0, which is a contradiction. □

The above proposition highlights a fundamental difference between the time-complexity of NBAC and UC in synchronous models. However, the proposition extends to IC. In fact, any IC algorithm can be easily transformed to an NBAC algorithm (without any additional rounds) as follows. Let $V1$ denote an ordered n-tuple in which every component is 1. Suppose we have an IC algorithm with IC-propose() primitive. We implement the NBAC-propose() primitive of the NBAC specification in the following way. When a process NBAC-proposes $v \in \{0, 1\}$, then if IC-proposes v. If a process IC-decides $V1$, then it NBAC-decides 1; if the process IC-decides an n-tuple different from $V1$, then it NBAC-decides 0. Note that the transformation by itself does not require any additional communication and hence can be performed even in an asynchronous model. Thus, this transformation immediately implies that the bound in section 6.2 and Proposition 6.4 applies to IC.

In a related work [10], we show for NBAC algorithms, an incompatibility between globally deciding by round 2 in the failure-free run where all processes propose 1 and globally deciding by round 1 in every run where some process proposes 0. However, that paper does not consider local decisions.

7. A matching synchronous algorithm. In [21], an NC algorithm was proposed that matches the global decision and global halting lower bounds. The algorithm can be easily modified to derive another algorithm that matches corresponding bounds for UC. However, we knew of no UC algorithm that matches the local decision lower bounds.

In this section, we present an algorithm for IC that *simultaneously* matches the local decision, global decision, and global halting lower bounds for most values of f and t. (We do not match the bounds in some boundary cases when f, t, and n are close to each other.) From our IC algorithm, we then derive matching algorithms for UC and NBAC. (Algorithms that match either the local decision or the global decision of NC are straightforward, but, as we showed in Proposition 6.2, no single NC algorithm can match both local and global decision lower bounds.)

7.1. IC algorithm overview. Our IC algorithm (Figure 7.1) is inspired by the Byzantine Generals algorithm of [21]. The algorithm runs for at most $t + 1$ rounds. Process p_i maintains two primary variables: (1) an ordered n-tuple est_i, component j of which contains the proposal value of p_j, provided p_i has received that value (either directly from p_j or relayed by some other process), and $\perp$ otherwise, and (2) a set of processes $halt_i$ that p_i knows to have either crashed or halted. In each round, the processes exchange estimate (EST) messages containing their *est* values. If the *halt* set at a process does not change in round k, then (1) if the *est* does not change in round k as well, the process decides on its *est* in round k, and otherwise, (2) the process decides on its *est* in round $k + 1$. Before halting, a process sends a special decision (DEC) message to all processes, so that the processes can distinguish a halt from a crash.

Roughly speaking, if the *halt* set at a process p_i does not change in some round k, then at the end of round k, no *alive* process has seen more proposal values than p_i. Thus, p_i can decide on its current est_i value, provided p_i ensures that all other processes see its current est_i. So p_i sends its *est* to all processes in round $k + 1$ and

```
at process $p_i$:
 1: propose($v_i$)
 2: Ordered $n$-tuples $est_i$ and $newest_i$: element $i$ initialized to $v_i$ and all other elements initialized to $\bot$
 3: Set $halt_i \leftarrow newhalt_i \leftarrow \emptyset$
 4: Boolean $decided_i \leftarrow lastRound_i \leftarrow false$
 5: for $1 \le r \le t+1$ do Multiset $S_i^r \leftarrow \emptyset$

 6: for round $r$ from 1 to $t+1$ do
 7:     $halt_i \leftarrow newhalt_i$
 8:     $est_i \leftarrow newest_i$

 9:     Send phase
10:     if $lastRound_i$ then
11:         send($r$, DEC, $est_i$) to all
12:     else
13:         send($r$, EST, $est_i$) to all

14:     Receive phase
15:     $S_i^r \leftarrow \{est_j \mid (r, \text{EST}, est_j)$ was received$\}$
16:     if $lastRound_i$ then
17:         if not $decided_i$ then
18:             decide($est_i$)                                                    {decision}
19:         return                                                                 {halt}
20:     if received any $(r, \text{DEC}, est_j)$ then
21:         $newest_i \leftarrow est_j$
22:         $lastRound_i \leftarrow true$
23:     else
24:         $newhalt_i \leftarrow \Pi \backslash$sender($S_i^r$)      {processes from which $p_i$ did not receive any message}
25:         for $1 \le j \le n$ do
26:             if there is any $est' \in S_i^r$ s.t. $est'[j] \ne \bot$ then $newest_i[j] \leftarrow est'[j]$ else $newest_i[j] \leftarrow \bot$
27:         if $newhalt_i = halt_i$ then
28:             if $est_i = newest_i$ then
29:                 decide($est_i$); $decided_i \leftarrow true$                  {decision}
30:             $lastRound_i \leftarrow true$
31:     if $r = t+1$ then
32:         if not $decided_i$ then
33:             decide($newest_i$)                                                 {decision}
34:         return                                                                 {halt}
```

FIG. 7.1. *An early deciding (and halting) interactive consistency algorithm.*

then decides. However, if the *est* of p_i does not change in round k, then p_i has already sent that *est* to all processes in round k; so p_i can decide at the end of that round.

7.2. Correctness.

In the following, a variable *var* at a process p_i is denoted var_i, and if p_i reaches the end of any round r, the value of var_i at the end of round r is denoted var_i^r; var_i^0 denotes the value of the variable at the end of line 5. (We omit the subscript of the variable when we make a statement that applies to multiple processes.) For $1 \le r \le t+1$, $faulty^r$ denotes the set of processes that have crashed by round r, and $faulty^0$ equals $\emptyset$. For any pair of ordered n-tuples d and d', we say that (1) $d = d'$ if for all $j \in [1, n]$, $d[j] = d'[j]$, (2) $d \preceq d'$ if for all $j \in [1, n]$, either $d[j] = \bot$ or $d[j] = d'[j]$, and (3) $d \not\preceq d'$ if $d \preceq d'$ is false.

First, we make the following simple observations that we frequently use: (1) (*Observation* O1) For the *est* value at every process and every $j \in [1, n]$, $est[j]$ is either the proposal value of p_j or $\bot$. (2) (*Observation* O2) If, before deciding, p_j receives an EST message from some process p_l in round k, then $newest_l^{k-1} \preceq newest_j^k$. (It follows that $newest_j^{k-1} \preceq newest_j^k$.)

Every process decides on some *est* value; thus, validity immediately follows from Observation O1. Termination follows from the simple observations that no process halts without deciding and no process completes round $t+1$ without halting (lines 31 to 34). Thus we detail only the proof of uniform agreement. We start with some general lemmas about the algorithm.

LEMMA 7.1. *If for some* $r \in [1, t]$ *no process decides by round* r*, then the following holds for every process* p_i *that completes round* r*. If* $lastRound_i^r = true$*, then every process* p_j *that completes round* r *has* $newest_j^r \preceq newest_i^r$*.*

Proof. We prove the lemma by induction on round number r, such that $r \in [1, t]$.

Base case $r = 1$. Suppose $lastRound_i^r = true$ and no process decides in round 1. Then p_i has executed either line 22 or line 30 of round 1. Observe that p_i executes line 22 only if some process sends DEC message to p_i. Since *lastRound* is initialized to false and the processes send DEC messages only when $lastRound = true$, no process has sent a DEC message in round 1. Thus p_i has executed line 30. So $newhalt_i^1 = halt_i^1 = \emptyset$, and hence, $newest_i^1$ contains proposal values of all processes. Thus, every process p_j that completes round 1 has $newest_j^1 \preceq newest_i^1$.

Induction hypothesis $r = k$. If no process decides by round k, then the following holds for every process p_i that completes round k. If $lastRound_i^k = true$, then every process p_j that completes round k has $newest_j^k \preceq newest_i^k$.

Induction step $r = k + 1 \leq t$. Suppose by contradiction that (1) no process decides by round $r = k + 1$, (2) there is a process p_i that completes round $k + 1$ such that $lastRound_i^{k+1} = true$ and $newest_i^{k+1} = d'$, and (3) another process p_j completes round $k + 1$ with $newest_j^{k+1} = d$ such that $d \not\preceq d'$. Process p_i has executed either line 22 or line 30. If p_i executed line 22, then p_i has received a ($k + 1$, DEC, d') message from some process p_l. To send a DEC message in round $k + 1$, p_l must have set $lastRound_l$ to *true* in round k. Thus, from the induction hypothesis, every process that completes round k has $newest^k \preceq d'$. Since $d \not\preceq d'$, process p_j receives a round $k + 1$ message from some process with an n-tuple d'' such that $d'' \not\preceq d'$, which is a contradiction because, for all processes that complete round k, we have $newest^k \preceq d'$. Hence, p_i executed line 30, and $halt_i^{k+1} = newhalt_i^{k+1}$. Since p_j completes round $k + 1$, p_i received the round $k + 1$ message from p_j containing $newest_j^k$, and hence, $newest_j^k \preceq newest_i^{k+1} = d'$. As $newest_j^{k+1} = d \not\preceq d'$, it follows that p_j received ($k+1$, $*$, d'') from some process p_m such that $d'' \not\preceq d'$, and p_i did not receive ($k + 1$, $*$, d'') from p_m (otherwise, $d'' \preceq newest_i^{k+1} = d'$). Thus $p_m \in newhalt_i^{k+1}$. However, as p_m completed round k, $p_m \notin newhalt_i^k = halt_i^{k+1}$. Thus, $halt_i^{k+1} \neq newhalt_i^{k+1}$, which is a contradiction. □

LEMMA 7.2. *If a process* p_i *does not halt or crash by round* $r \in [0, t]$*, then* p_i *has* $halt_i^k \neq newhalt_i^k$ *for all* $k \in [1, r - 1]$*.*

Proof. The proof is obvious from the algorithm. □

LEMMA 7.3. *If no correct process halts by some round* $r - 1 \in [0, t - 1]$*, and if there is a process* p_i *such that, for every round number* $r' \in [1, r]$*,* $halt_i^{r'} \neq newhalt_i^{r'}$*, then* $|faulty^r| \geq r$*.*

Proof (for uniformity of presentation, we slightly abuse the terminology and say that *for all runs, no process halts or crashes by round* 0). Suppose there is a round r such that no correct process halts by round $r - 1$ and there exists a process p_i such that, for every round number $r' \in [1, r]$, $halt_i^{r'} \neq newhalt_i^{r'}$. Clearly, $halt_i^{r'} = newhalt_i^{r'-1} \subseteq newhalt_i^{r'}$. Thus $|newhalt_i^r| \geq r$. Every process in $newhalt_i^r$ has either halted by round $r - 1$ or crashed by round r. Since no correct process halts by round $r - 1$, $newhalt_i^r \subseteq faulty^r$, and hence, $|faulty^r| \geq r$. □

LEMMA 7.4. *If no correct process halts by round* $r+1 \in [1, t]$*, then* $|faulty^r| \geq r$*.*

Proof. The proof is trivial for $r + 1 = 1$. So we consider the case $r + 1 \in [2, t]$. Suppose that no correct process halts by round $r + 1$. Consider any correct process p_i. Since p_i does not halt by round $r + 1 \leq t$, it follows from Lemma 7.2 that for

$r' \in [1, r]$, $halt_i^{r'} \neq newhalt_i^{r'}$. Since no correct process halts by round $r - 1 \leq t - 1$, applying Lemma 7.3, we have $|faulty^r| \geq r$. □

LEMMA 7.5. *If every process that decides decides in line* 29 *of round* $t + 1$ *or line* 33 *of round* $t + 1$*, then* $|faulty^t| = t$.

Proof. The proof is trivial when $t = 0$. Thus we consider the case $t \geq 1$. Suppose that every process that decides decides in line 29 of round $t+1$ or line 33 of round $t+1$. Consider any correct process p_i. Since p_i does not decide in line 18 of round $t + 1$, $lastRound_i^t = false$. Thus $newhalt_i^t \neq halt_i^t$ (from lines 27 and 30). Furthermore, as p_i does not halt by round t, from Lemma 7.2 it follows that for every $g \in [1, t - 1]$, $newhalt_i^g \neq halt_i^g$. Thus for every $g \in [1, t]$, $newhalt_i^g \neq halt_i^g$. Since no process decides (and hence, halts) by round t, by applying Lemma 7.3 (with $r - 1 = t - 1$), we have $|faulty^t| \geq t$. As at most t processes can crash in a run, $|faulty^t| = t$. □

LEMMA 7.6 (uniform agreement). *No two processes decide differently.*

Proof. If no process decides, then the lemma trivially holds. Suppose some process decides. Consider the lowest round number r in which some process decides. Let p_i be a process that decides in round r, say, on some n-tuple d. We divide the proof into two parts: (a) p_i does not decide in line 33 of round $t + 1$, and (b) $r = t + 1$ and p_i decides in line 33 of round $t + 1$.

(a) p_i *does not decide in line* 33 *of round* $t + 1$: Thus, process p_i decides either in (1) line 18 or in (2) line 29 of round $r \leq t + 1$. In both cases, we show the following: no process can decide an n-tuple different from d in round r, and any process that completes round r without deciding in line 18 and line 29 does so with $newest^r = d$. This implies uniform agreement because every process that decides in round r has decision value the same as its $newest^r$, and in subsequent rounds, d is the only surviving *newest* and *est* value. (Note that, even if $r = t + 1$, and another process p_j decides in line 33 of round r, p_j decides on $newest_j^r = d$.)

Process p_i *decides in line* 18 *of round* r: Notice that $r > 1$ because no process can decide at line 18 in round 1 (as $lastRound^0 = false$). Since p_i decides in line 18, $lastRound_i^{r-1} = true$ and p_i sends a DEC message in round r. We claim that every DEC message sent in round r is (r, DEC, d). Suppose that another process p_j sends a (r, DEC, $d1$) message. Then $lastRound_j^{r-1} = true$. Since no process decides by round $r - 1$, applying Lemma 7.1 twice we have $d1 = newest_j^{r-1} \preceq newest_i^{r-1} = d$ and $d = newest_i^{r-1} \preceq newest_j^{r-1} = d1$, i.e., $d1 = d$. As p_i completes the send phase of round r, every process receives at least one (r, DEC, d) message and either decides d in line 18 or adopts d as *newest* in line 21.

Process p_i *decides in line* 29 *of round* r: Thus $est_i = newest_i$ is d in line 28 of round r, and p_i sent (r, EST, d) in round r. We claim that no process decides a value different from d in round r. Clearly, p_i does not receive any DEC message in round r (otherwise, p_i would not have executed line 29). Suppose some process p_j decides $d1$ in round r. If process p_j decides in line 18, then p_j sends a DEC message in round r, and p_i receives that message (as p_j completes the send phase of round r, none of its messages are lost), which is a contradiction. Suppose that p_j decides in line 29. Thus $est_j = newest_j$ is $d1$ in line 28 of round r, and p_i sent (r, EST, $d1$) in round r. Since p_i receives a round r message from p_j and vice versa, $d1 \preceq d$ and $d \preceq d1$; i.e., $d = d1$. If p_j decides in line 33, then it decides on the *newest* value adopted in round $t + 1$. We show below that every process that updates its *newest* in round k updates it to d.

We now show that any process that completes round r without deciding in line 18 or line 29 does so with $newest = d$. Suppose by contradiction that some process p_j completes round r with $newest = d2 \neq d$ and without deciding in line 18 and line 29.

Process p_j updates its variable *newest* in line 21 or line 26. Suppose p_j updates its *newest* in line 21. Then p_j has received a DEC message from some process p_m. Since p_i decides at line 29, it does not receive any DEC message in round r. Thus $p_m \in \mathit{newhalt}_i^r$. Since p_m completes round $r-1$, $p_m \notin \mathit{newhalt}_i^{r-1} = \mathit{halt}_i^r$. (If $r = 1$ then obviously $p_m \notin \mathit{halt}_i^r = \emptyset$.) Hence, the predicate in line 27 evaluates to false at p_i, and p_i cannot decide in line 29, which is a contradiction. Thus, p_j updates its *newest* in line 26. Since p_i completes round r by deciding d and evaluates the condition in line 28 to true, p_i sends a (r, EST, d) message in round r. Thus p_j receives (r, EST, d) from p_i, and hence, $d \preceq d2$. As $d2 \neq d$, it follows that $d2 \npreceq d$. Consequently, there is a process p_m such that p_j receives $d3 \npreceq d$ from p_m, and p_i does not receive any message from p_m in round r. Thus, $p_m \in \mathit{newhalt}_i^r$. However, p_m completes round $r-1$ and hence, $p_m \notin \mathit{newhalt}_i^{r-1} = \mathit{halt}_i^r$. (If $r = 1$ then obviously $p_m \notin \mathit{halt}_i^r = \emptyset$.) Hence, the predicate in line 27 evaluates to false at p_i, and p_i cannot decide in line 29, which is a contradiction.

(b) $r = t+1$ *and* p_i *decides in line* 33 *of round* $r = t+1$: From the definition of r, every process that decides decides in round $t+1$. We have shown above that if any process decides in line 18 or line 29 of round $t+1$ then every process that decides in round $t+1$, decides the same value. Therefore, we need only to consider the case where every process that decides does so at line 33 of round $t+1$. From Lemma 7.5, we have $|\mathit{faulty}^t| = t$. Hence, every process that enters round $t+1$ is a correct process. Consequently, every process that enters round $t+1$ receives the same set of messages in round $t+1$. Observe that no process sends a DEC message in round $t+1$ (otherwise, that process decides in line 18 of round $t+1$ or line 29 of round t, which is a contradiction). Thus every process that enters round $t+1$ updates *newest* to the same value in line 26 and decides on identical values in line 33. □

7.3. Time-complexity. We now discuss the time-complexity of our IC algorithm. We show through the following lemma that, in runs with at most $f \geq 1$ failures, the algorithm achieves local decision in $f+1$ rounds and global decision in $f+2$ rounds. However, when $f = 0$, the local decision takes the same number of rounds as the global decision (2 rounds); recall that we showed in Proposition 6.4 that NBAC (and hence, IC) algorithms require 2 rounds for local decision when $f = 0$. (In section 7.4, we show a UC algorithm that achieves local decision in round 1 when $f = 0$.)

We say that a process p_i *learns* index $l \in [1,n]\backslash\{i\}$ in round k if $\mathit{newest}_i^{k-1}[l] = \bot$ and $\mathit{newest}_i^k[l] \neq \bot$. (In other words, p_i learns about the proposal value of p_l in round k.) We say that p_i learns index i in round 0. Also, we say that p_i learns index l from p_j in round k if $\mathit{newest}_i^{k-1}[l] = \bot$ and p_i receives a round k message from p_j containing an *est* such that $\mathit{est}[l] \neq \bot$. On the other hand, if p_j sends an *est* such that $\mathit{est}[l] \neq \bot$ in round k, then we say that p_j propagates index l in round k. (Note that there may be more than one process from which a process learns the same index in a round.) Clearly, if p_i propagates l in round k, then p_i learns l in a lower round.

LEMMA 7.7. *In every run with at most f faulty processes, the following properties hold:*

(a) *If $f \in [1,t]$, then there is a correct process that decides by round $f+1$.*

(b) *If $f \in [0, t-2]$, then any process that halts halts by round $f+2$.*

(c) *Any process that halts does so by round $t+1$.*

Proof. (a) For $f = t$, the proof is trivial because every correct process decides by round $t+1$. Consider a run in which at most $f \in [1, t-1]$ processes crash, and suppose by contradiction that no correct process decides by round $f+1$. Thus, no process halts by round $f+1 \leq t$. It follows from Lemma 7.4 that $|\mathit{faulty}^f| \geq f$. Since at most f processes crash in the run, $|\mathit{faulty}^f| = f$ and every process that enters round $f+1$ is correct. Furthermore, since no correct process halts by round f, Lemma 7.4 implies that $|\mathit{faulty}^{f-1}| \geq f-1$. Since $|\mathit{faulty}^f| = f$, *at most* one process crashes in round f.

Let S be the set of processes that enter round $f+1$. Since every process in S is correct, all of them complete round $f+1$. We establish a contradiction by showing that some process in S decides in line 29 of round $f+1$. We demonstrate this fact indirectly by showing the following four claims for processes in S in round $f+1$: (1) every process has $\mathit{lastRound} = \mathit{false}$ in line 16, (2) no process receives a DEC message in round $f+1$, (3) every process evaluates the predicate in line 27 to true, and (4) some process evaluates the predicate in line 28 to true.

Claim 1. Suppose by contradiction that, at some process in S, $\mathit{lastRound} = \mathit{true}$ in line 16 of round $f+1$. Then that process halts in round $f+1$. This leads to a contradiction because we know that every process in S is correct, and (from our initial assumption) correct processes do not decide (and hence, do not halt) by round $f+1$.

Claim 2. Suppose by contradiction that some process $p_i \in S$ receives a DEC message from some process p_j in round $f+1$. Since every process that enters round $f+1$ is correct, p_j is a correct process, and hence, p_j decides in line 18 of round $f+1$ or line 29 of round f, which is a contradiction. Thus no process in S receives a DEC message in round $f+1$.

Claim 3. Suppose by contradiction that some process $p_i \in S$ evaluates the predicate at line 27 to false; i.e., $\mathit{halt}_i^{f+1} \neq \mathit{newhalt}_i^{f+1}$. Since p_i does not halt by round $f+1 \leq t$, from Lemma 7.2 we have $\mathit{halt}_i^k \neq \mathit{newhalt}_i^k$ for every k in $[1, f]$. Thus $\mathit{halt}_i^k \neq \mathit{newhalt}_i^k$ for every k in $[1, f+1]$. As no correct process halts by round $f+1$, from Lemma 7.3 (with $r-1 = f \leq t-1$) it follows that $|\mathit{faulty}^{f+1}| \geq f+1$, which is a contradiction.

Claim 4. Suppose by contradiction that every process in S evaluates the predicate in line 28 to false. It follows that, in round $f+1$, every process in S learns an index. (Recall that every process that enters round $f+1$ is correct and is in set S.)

Consider any process $p_i \in S$ which learns index $l1$ in round $f+1$ from some process p_x. Suppose p_x learns index $l2$ in round $f+1$ from process p_y. Since p_i learns from p_x and p_x learns from p_y, $p_i \neq p_x$ and $p_x \neq p_y$. (Note that p_i and p_y may not be distinct.) Since p_x propagates $l1$ and learns $l2$, $l1 \neq l2$.

Since p_x is a correct process, p_x learns $l1$ in round f (otherwise, if p_x learned $l1$ in a round lower than f, p_x would have propagated $l1$ to p_i by round f). Similarly, p_y learns $l2$ in round f. Consider the process p'_x from which p_x learns $l1$ in round f. Process p'_x must have crashed in round f; otherwise, on receiving the round f message from p'_x, p_i would have learned $l1$ in round f. Similarly, the process p'_y from which p_y learns $l2$ in round f must have crashed in round f; otherwise, p_x would have learned $l2$ from p'_y in round f. We claim that p'_x and p'_y are distinct processes. Otherwise, if $p'_x = p'_y$, then p'_x propagates both $l1$ and $l2$ in round f, and when p_x receives a message from p'_x in round f, p_x learns both $l1$ and $l2$ in round f, which is a contradiction. (Recall that we assumed p_x learned $l2$ in round $f+1$.)

Thus two processes, p'_x and p'_y, crash in round f. However, recall that we have already shown (in the first paragraph of this proof) that *at most* one process crashes in round f, which is a contradiction.

(b) Consider a run in which at most $f \in [0, t-2]$ processes crash, and suppose by contradiction that a process p_i completes round $f+2$ without halting. Observe that if any process p_j halts at round $k \leq f+1$, then p_j sends a DEC message in round k. Since p_j completes round k, p_i receives the DEC message, sets *lastRound* to *true* in round k, and halts in round $k+1 \leq f+2$. Thus no process halts by $f+1$. As p_i does not halt by round $f+2 \leq t$, from Lemma 7.2, for every $g \in [1, f+1]$, we have $newhalt_i^g \neq halt_i^g$. Applying Lemma 7.3 (with $r-1 = f \leq t-1$) we have $|faulty^{f+1}| \geq f+1$, which is a contradiction.

(c) This part of the proof is obvious from the algorithm. □

7.4. Deriving NBAC and UC algorithms. In section 6.3, we showed how to transform any IC algorithm to an NBAC algorithm without any additional communication. An equally straightforward transformation generates a UC algorithm from an IC algorithm: on UC-propose(v), a process invokes IC-propose(v), and if a process IC-decides an n-tuple d, then it UC-decides $d[l]$, where l is the lowest index such that $d[l] \neq \perp$.

The IC algorithm of Figure 7.1 does not locally decide in round 1 in a failure-free run ($f = 0$). Therefore, to match the local decision lower bound for UC when $f = 0$, we modify the UC algorithm obtained from our IC algorithm by adding the following: p_1 UC-decides on its proposal value v_1 in the receive phase of round 1. This modification does not violate UC agreement because, if p_1 completes the send phase of round 1, then every process that completes round 1 has $newest[1] = v_1$ at the end of round 1. At the beginning of round 2, the processes set *est* to *newest*. Subsequently, at all processes, $newest[1]$ and $est[1]$ are always v_1. Thus, in our transformation of IC algorithm to UC algorithm, no process can UC-decide a value different from v_1.

7.5. Synchronous results summary. Combining our lower bound results with the time-complexity of the IC algorithm, the derived NBAC and UC algorithms, and the simple NC algorithm sketched in the introduction, we get the following tight bounds:

1. $\forall t \in [1, n-1]$, $\forall f \in [0, t]$, (SCS_t, SCS_f, NC, ld) $= f$. Local decision bound for consensus.
2. $\forall t \in [1, n-1]$, $\forall f \in [0, t-1]$, (SCS_t, SCS_f, UC, ld) $= f+1$. Local decision bound for uniform consensus.
3. (a) $\forall t \in [1, n-1]$, $\forall f \in [1, t-1]$, $\forall P \in$ {NBAC, IC}, (SCS_t, SCS_f, P, ld) $= f+1$. (b) $\forall t \in [1, n-1]$, $\forall P \in$ {NBAC, IC}, (SCS_t, SCS_0, P, ld) $= 2$. Local decision bounds for nonblocking atomic commit and interactive consistency.

8. Eventually synchronous lower bound. In this section we investigate lower bounds for UC in eventually synchronous models ES_t. We do not consider lower bounds for NBAC and IC in ES_t because they are impossible to solve in ES_t if $t \geq 1$. Furthermore, any algorithm that solves consensus also solves uniform consensus in ES_t [16]. Thus, in ES_t, we investigate only lower bounds for uniform consensus.

We know from [14] that every UC algorithm in ES_t has a run that requires an arbitrary number of rounds for any correct process to decide (because a run may remain "asynchronous" for an arbitrary number of rounds). Thus, we focus on *synchronous* runs of ES_t, i.e., runs in which $GSR = 1$. (In other words, a run of ES_t is synchronous if it is also a run of SCS_t.)

As all runs of SCS_t are synchronous runs of ES_t, the local and global decision lower bounds for UC in SCS_t also hold for synchronous runs of ES_t; i.e., roughly speaking, the local decision lower bound is $f+1$ and the global decision lower bound is $f+2$. However, we knew of no algorithm that showed that the bounds are tight, except when $f = 0$ and $f = t$ (the best and the worst case): the global decision tight bound is 2 rounds in runs with $f = 0$ crashes [19, 28, 26], and $t+2$ rounds in runs with at most $f = t$ crashes [9].

In the following proposition, we show that, for most values of f, the local decision lower bound is $f+2$ rounds, which is the same as the lower bound for global decision. (We give a matching algorithm in section 9.) The proposition states that every UC algorithm in ES_t has a run in SCS_f (i.e., a synchronous run with at most f crashes) in which every correct process decides in round $f+2$ or a higher round.

PROPOSITION 8.1. *For all t s.t. $1 \leq t < n/2$, for all $f \in [0, t-3]$,* (ES_t*,* SCS_f*, UC, ld*) $\geq f+2$.

Remarks. We exclude the following two cases. (1) $t = 0$: In this case, processes can decide after exchanging proposal values in the very first round in synchronous runs (e.g., decide always on the proposal value of p_1). (2) $t \geq n/2$: In this case, we know that there is no UC algorithm in ES_t.

Proof. Suppose by contradiction that there is a UC algorithm A in ES_t and an integer f in $[0, t-3]$ such that in every synchronous run of A with f crashes some correct process decides by round $f+1$. Since SCS_t is a submodel of ES_t, A solves UC in SCS_t as well. We also observe that any UC algorithm also solves WA. Thus A solves WA in SCS_t. Thus from Lemma 5.2 we know that there are two runs of A in SCS_t such that their round f configurations, y and y', satisfy the following: (1) at most f processes have crashed in each configuration, (2) the configurations differ at exactly one process, say, p_i, and (3) $val(y) = 0$ and $val(y') = 1$. (Recall that, given a configuration C, $r(C)$ and $val(C)$ are defined only if C is a configuration of a run in a synchronous model.)

We note that in y or y', any alive process p_j that is distinct from p_i has not yet decided. Otherwise, as y and y' differ only at p_i, process p_j would decide the same value v in y and y', and hence, p_j is a correct process that decides v in both $r(y)$ and $r(y')$, which is a contradiction.

Let z and z' denote the configurations at the end of round $f+1$ of $r(y)$ and $r(y')$, respectively. Runs $r(y)$ and $r(y')$ are runs of A in SCS_t and hence synchronous runs of A in ES_t. As at most f processes crash in each run, $r(y)$ and $r(y')$, it follows from our assumption about algorithm A that some correct process decides by round $f+1$ in each run. Thus, there is at least one alive process in z, say, q_1, that has decided 0. Similarly, there is at least one alive process in z', say, q_3, that has decided 1. There are three cases to consider. (We now consider runs of A in ES_t.)

Case 1. $p_i \notin \{q_1, q_3\}$. Thus we have (1) a round $f+1$ configuration z and a process q_1 such that at most f processes have crashed in z, and q_1 is alive and has decided 0 in z, (2) a round $f+1$ configuration z' and a process q_3 such that at most f processes have crashed in z', and q_3 is alive and has decided 1 in z', and (3) process p_i is distinct from both q_1 and q_3. (Processes q_1 and q_3 might not be distinct.) There are two subcases to consider.

Case 1a. Process p_i is alive in y and y'. Consider the following two synchronous runs of A.

$R1$ is a run such that (1) the round f configuration is y, (2) p_i crashes in the send phase of round $f+1$ such that only q_1 and q_3 receive the message from p_i, (3) q_1 and q_3 crash in round $f+2$ before sending any message, and (4) no process distinct from p_i, q_1, and q_3 crashes after round f. Notice that q_1 cannot distinguish the round $f+1$ configuration of $R1$ from z and therefore decides 0 at the end of round $f+1$ in $R1$. By uniform agreement, every correct process decides 0. Since $t \leq n-1$, there is at least one correct process in $R1$, say, p_l.

$R2$ is a run such that (1) the round f configuration is y', (2) p_i crashes in the send phase of round $f+1$ such that only q_1 and q_3 receive the message from p_i, (3) q_1

and q_3 crash in round $f+2$ before sending any message, and (4) no process distinct from p_i, q_1, and q_3 crashes after round f. Notice that q_3 cannot distinguish the round $f+1$ configuration of $R2$ from z' and therefore decides 1 at the end of round $f+1$ in $R2$. However, p_l cannot distinguish $R1$ from $R2$: at the end of round $f+1$, the two runs are different only at p_i, q_1, and q_3, and none of the three processes sends messages after round $f+1$ in both runs. Thus (as in $R1$) p_l decides 0 in $R2$, which is a contradiction to uniform agreement.

Case 1b. Process p_i has crashed in either y or y'. (Process p_i has not crashed in both y and y' because p_i has different states in y and y'.) Without loss of generality, we can assume that p_i has crashed in y, and hence, p_i is alive in y'. Consider the following two synchronous runs of A.

$R12$ is a run such that (1) the round f configuration is y (and hence, p_i has crashed before round $f+1$), (2) no process crashes in round $f+1$, (3) q_1, and q_3 crash in round $f+2$ before sending any message, and (4) no process distinct from p_i, q_1, and q_3 crashes after round f. Observe that the round $f+1$ configuration of $R12$ is z, and hence, q_1 decides 0 at the end of round $f+1$ in $R12$. Due to uniform agreement, every correct process decides 0 in $R12$. Since $t \leq n-1$, there is at least one correct process in $R12$, say, p_l.

$R21$ is a run such that (1) the round f configuration is y', (2) p_i crashes in the send phase of round $f+1$ such that only q_1 and q_3 receive the message from p_i, (3) q_1 and q_3 crash in round $f+2$ before sending any message, and (4) no process distinct from p_i, q_1, and q_3 crashes after round f. Notice that q_3 cannot distinguish the round $f+1$ configuration of $R21$ from z' because it receives the round $f+1$ message from p_i in both runs. Thus (as in z') q_3 decides 1 at the end of round $f+1$ in $R21$. However, p_l cannot distinguish $R12$ from $R21$: at the end of round $f+1$, the two runs are different only at p_i, q_1, and q_3, and none of them sends messages after round $f+1$ in both runs. Thus (as in $R12$), p_l decides 0 in $R21$, which is a contradiction to uniform agreement.

Case 2. $p_i \in \{q_1, q_3\}$ and p_i is alive in both y and y'.

Remark. To see why we cannot reuse the proof of Case 1, observe that if $p_i = q_1$, then run $R1$ is not a valid run of A in SCS_t: in SCS_t, p_i cannot decide in the receive phase of round $f+1$ while some of its messages from that round are lost. Similarly, if $p_i = q_3$, then run $R2$ is not a valid run in SCS_t. Hence, in this case, we construct some runs of A in ES_t that are not in SCS_t (i.e., nonsynchronous runs) to derive a contradiction.

Without loss of generality we can assume that $p_i = q_1$. (Note that the proof holds even if $p_i = q_1 = q_3$.) Consider the following three runs ($R3$ is a synchronous run, whereas $R4$ and $R5$ are nonsynchronous runs. We would like to point out that, as required by the properties of ES_t, in all nonsynchronous runs that we construct, we ensure that in every round, processes received at least $n-t$ messages of the current round, and channels are reliable):

$R3$ is a run such that (1) the round f configuration is y, (2) p_i crashes in round $f+1$ before sending any message, (3) if $q_3 \neq p_i$ then q_3 crashes in round $f+2$ before sending any message, and every message sent by q_3 in round $f+1$ is received in round $f+1$, (4) no process distinct from p_i and q_3 crashes in round $f+1$ or in a higher round, and (5) no message is delayed. Since $t < n/2 < n-1$, there is at least one correct process in $R3$, say, p_l. Suppose p_l decides $v \in \{0,1\}$ in some round $K1 \geq f+1$. (To see why p_l cannot decide before round $f+1$ in $R3$, notice that the state of p_l at the end of round f is the same in runs $r(y)$, $r(y')$, and $R3$, because $p_l \neq p_i$. If p_l

decides v before round $f+1$ in $R3$, then it also decides v in $r(y)$ and $r(y')$. However, $val(y) \neq val(y')$.)

$R4$ is a run such that (1) the round f configuration is y, (2) p_i and q_3 crash in round $f+2$ before sending any message, and only p_i and q_3 receive the round $f+1$ message from p_i (all other round $f+1$ messages from p_i are lost[1]), (3) if $q_3 \neq p_i$, every process that completes round $f+1$ receives a round $f+1$ message from q_3, (4) no process distinct from p_i and q_3 crashes in round $f+1$ or in a higher round, and (5) no message is delayed. Notice that p_i cannot distinguish the configuration at the end of round $f+1$ in $R4$ from z, and thus, p_i decides 0 at the end of round $f+1$ in $R4$ (because $p_i = q_1$ decides 0 in z). However, p_l cannot distinguish the round $K1$ configuration of $R4$ from that of $R3$ because (a) at the end of round f, the two runs are different only at p_i, (b) all round $f+1$ messages sent by p_i to processes distinct from p_i and q_3 are lost, and (c) p_i and q_3 do not send messages after round $f+1$. Thus (as in $R3$) p_l decides v in round $K1$.

$R5$ extends y' in the same way as $R4$ extends y. Namely, $R5$ is a run such that (1) the round f configuration is y', (2) p_i and q_3 crash in round $f+2$ before sending any message, and only p_i and q_3 receive the round $f+1$ message from p_i (all other round $f+1$ messages from p_i are lost), (3) if $q_3 \neq p_i$, then every process that completes round $f+1$ receives the round $f+1$ message from q_3, (4) no process distinct from p_i and q_3 crashes in round $f+1$ or in a higher round, and (5) no message is delayed. Notice that q_3 cannot distinguish the configuration at the end of round $f+1$ in $R5$ from z' (because in both runs, q_3 receives the round $f+1$ message from p_i), and thus, q_3 decides 1 at the end of round $f+1$ in $R5$. However, p_l cannot distinguish the round $K1$ configuration of $R5$ from that of $R3$ because, (a) at the end of round f the two runs are different only at p_i, (b) all round $f+1$ messages sent by p_i to processes distinct from p_i and q_3 are lost, and (c) p_i and q_3 do not send messages after round $f+1$. Thus (as in $R3$) p_l decides v in round $K1$.

Clearly, either $R4$ or $R5$ violates uniform agreement: p_l decides v in both runs; however, p_i decides 0 in $R4$ and q_3 decides 1 in $R5$.

Case 3. $p_i \in \{q_1, q_3\}$ and p_i has crashed in either y or y'. (Process p_i has not crashed in both y and y' because p_i has different states in y and y'.) Notice that the case $p_i = q_1 = q_3$ is not possible because, in that case, p_i is alive in both z and z' and hence in y and y'. We show the contradiction for the case when $p_i = q_1 \neq q_3$. (The contradiction for $p_i = q_3 \neq q_1$ is symmetric.)

Since, $p_i = q_1$, p_i is alive in z and hence alive in y. Thus p_i has crashed in y'. Consider the following nonsynchronous run.

$R6$ is a run such that (1) the round f configuration is y, (2) in round $f+1$, only p_i receives the round $f+1$ message from itself (all other messages sent by p_i in round $f+1$ are lost), (3) p_i crashes in round $f+2$ before sending any message, (4) no process distinct from p_i crashes in round $f+1$ or in a higher round, and (5) no message is delayed. At the end of round $f+1$ in $R6$, p_i cannot distinguish the configuration from z and therefore decides 0 (because $p_i = q_1$ decides 0 in z). However, q_3 does not receive the round $f+1$ message from p_i in $R6$, and hence, q_3 cannot distinguish the configuration at the end of round $f+1$ in $R6$ from z'. (Observe that, in z', q_3 does not receive the round $f+1$ message from p_i because p_i has crashed in y'.) Consequently, q_3 decides 1 in $R6$, which is a contradiction to uniform agreement. □

Remark. A closer look at the proof of Proposition 8.1 reveals that the non-

[1] From the definition of ES_t, messages sent by a faulty process (p_i) may be lost in a nonsynchronous run.

synchronous runs we construct ($R4$, $R5$, and $R6$) require only a small amount of nonsynchrony in the model. The three runs are valid in a weakened synchronous model where the following holds: even if some message from process p_i is lost in round $f+1$, then p_i might complete round $f+1$. (Recall that, in a synchronous model, if some message from p_i is lost in round $f+1$, then p_i has necessarily crashed in the send phase of round $f+1$.) It is easy to see that such runs are also valid in the synchronous send-omission model [17] as well as in an asynchronous round based model enriched with a *Perfect* failure detector [2]. Thus the $f+2$ local decision lower bound in synchronous runs also extends to these two models.

9. A matching eventually synchronous algorithm.

In this section, we present a UC algorithm in ES_t that matches the local and global decision lower bounds in synchronous runs. We assume that $t < n/2$, as UC is impossible to solve in ES_t if $t \geq n/2$ [11]. As we pointed out earlier, [19, 28, 26] give a UC algorithm in ES_t that matches the global decision bound for synchronous runs with $f = 0$ crashes, and [9] gives a UC algorithm in ES_t that matches the global decision bound for synchronous runs with $f = t$ crashes. We knew of no UC algorithm that matches the bounds for $1 \leq f \leq t-1$.

Figure 9.1 presents a uniform consensus algorithm A_{es} in ES_t that globally decides (and hence, locally decides) within $f+2$ rounds in every synchronous run with at most f crashes for $0 \leq f \leq t$. In other words, our algorithm matches the $f+2$ round global (and local) decision lower bound for synchronous runs of UC algorithms in ES_t.

9.1. Overview.

Algorithm A_{es} is a generalization of the UC algorithm of [9] modified for early decision. A_{es} assumes the following: (1) the model ES_t with $0 \leq t < n/2$ (i.e., a majority of processes are correct), (2) any message sent by a process p_i to itself in any round k is received in round k, or p_i crashes in round k, and (3) the set of proposal values in a run is a totally ordered set, e.g., every process p_i can tag its proposal value with its index i and then the values can be ordered based on this tag. (A matching algorithm that does not rely on each process receiving at least $n-t$ messages in every round is described in [8].)

The algorithm A_{es} proceeds in sessions, where each session is composed of $t+2$ rounds of message exchange. A run globally decides within $f+2$ rounds in a "synchronous" session, provided at most f processes crash in the run. In each round of a session, processes exchange their estimate (of the decision value), and, roughly speaking, adopt the minimum estimate value seen in the round as the estimate for the next round. In this respect, a session of A_{es} is similar to the IC algorithm presented in section 7: if the model was synchronous, then a process p_i could simply monitor the set of processes from which p_i did not receive any message (set $Halt_i$), and then, p_i could decide on its own estimate when $Halt_i$ did not change for a round. Basically, p_i could do so because, in a synchronous model, $Halt_i$ would be equal to the set of crashed processes, and hence, if $Halt_i$ did not change for a round, then p_i would have the smallest estimate among all alive processes.

However, in ES_t, even if p_i does not receive a message from some process p_j, p_j might not have crashed, and p_j can continue sending messages in subsequent rounds. Thus, even if $Halt_i$ does not change for a round, p_i might not have the lowest estimate among all alive processes. Therefore, in A_{es}, in addition to the estimate values, processes also exchange the $Halt$ sets to detect whether the current session is synchronous. Furthermore, to ensure early decision, p_i maintains and exchanges a variable STATE_i which indicates if p_i considers the current session to be synchronous ($\text{SYNC}1$), or if p_i

```
at process $p_i$:
propose($v_i$)
$est_i \leftarrow v_i$
for round $s_i$ from 1 to $\infty$ do
    $k_i \leftarrow ((s_i - 1) \bmod (t+2)) + 1$                    {$k_i$ varies from 1 to $t+2$}
    if $k_i = 1$ and STATE$_i \neq$ DECIDE then
        $Halt_i \leftarrow \emptyset$
        STATE$_i \leftarrow$ SYNC1          {STATE$_i$ is either SYNC1, SYNC2, NSYNC, or DECIDE}

    Send phase
    send($s_i$, $est_i$, STATE$_i$, $Halt_i$) to all

    Receive phase
    wait until received messages in round $s_i$
    if STATE$_i$ = DECIDE then
        return
    if received any ($s_i$, $est'$, DECIDE, $*$) then
        $est_i \leftarrow est'$; decide($est_i$); STATE$_i \leftarrow$ DECIDE; go to the next round      {decision}
    if STATE$_i \in$ {SYNC1, SYNC2} then
        $Halt_i \leftarrow Halt_i \cup \{p_j \mid (p_i$ received($s_i$, $*$, NSYNC, $*$) from $p_j$) or
        ($p_i$ received($s_i$, $*$, $*$, $Halt_j$) from $p_j$ s.t. $p_i \in Halt_j$) or ($p_i$ did not receive any round $s_i$ message
        from $p_j$)}
        $msgSet_i \leftarrow \{ m \mid m$ is a round $s_i$ message received from $p_j \notin Halt_i\}$
        $est_i \leftarrow$ Min{$est \mid (*, est, *, *) \in msgSet_i$}
        if (STATE$_i$ = SYNC2) and ($|Halt_i| \leq t$) and (STATE  = SYNC2 for every message in $msgSet_i$) then
            decide($est_i$); STATE$_i \leftarrow$ DECIDE; go to the next round      {decision}
        if $|Halt_i| \leq k_i - 1$ then
            STATE$_i \leftarrow$ SYNC2
        if $k_i \leq |Halt_i| \leq t$ then
            STATE$_i \leftarrow$ SYNC1
        if $|Halt_i| > t$ then
            STATE$_i \leftarrow$ NSYNC
    if (STATE  = NSYNC) and (received any ($s_i$, $est'$, SYNC2, $*$)) then
        $est_i \leftarrow est'$
```

FIG. 9.1. *A uniform consensus algorithm A_{es} in ES_t.*

considers the session to be synchronous with the possibility of a decision in the next round (SYNC2), or whether p_i considers the session to be asynchronous (NSYNC).

9.2. Description. The processes invoke propose($*$) with their respective proposal values as a parameter, and the propose procedure progresses in *sessions*: a session consists of $t+2$ rounds, and session sn contains rounds from $((sn-1)*(t+2))+1$ to $sn*(t+2)$. We call the kth round in a session sn (i.e., round $((sn-1)*(t+2))+k$) *step k* of session sn. Recall that, for every run R in ES_t, there is an unknown round number GSR from which the system is *synchronous* (eventual synchrony property of ES_t). We say that a session is synchronous if the session starts in round GSR or in a higher round.

Every process p_i maintains the following variables:

k_i is the current round number;

STATE$_i$ at p_i reflects its view on how much progress is made toward achieving a decision in the current session: (1) if STATE$_i$ is updated to NSYNC then p_i considers the current session to be asynchronous, (2) if STATE$_i$ is updated to SYNC1 then p_i considers the session to be synchronous but p_i cannot decide in the next round, (3) if STATE$_i$ is updated to SYNC2 then p_i considers the session to be synchronous with the possibility of a decision in the next round, and (4) p_i updates STATE$_i$ to DECIDE upon decision;

est_i is the estimate of the possible decision value, and, roughly speaking, the minimum value seen by p_i;

$Halt_i$ is a set of processes p_j such that, in the current round or a lower round of this session, at least one of the following occurred: p_i received STATE = NSYNC from p_j, p_i did not receive a message from p_j, or p_i received a messages from p_j with $p_i \in Halt_j$;

$msgSet_i$ is a set of messages received by p_i from processes that are not in $Halt_i$.

The variables are initialized as follows. Round number s_i starts from 1 and est_i is initialized to the proposal value of p_i. Variables STATE_i and $Halt_i$ are initialized to SYNC1 and $\emptyset$, respectively and, if p_i has not yet decided, are reset to their initial values at the beginning of each session. In each round, processes exchange est, STATE, and $Halt$ variables, update their own variables depending upon the messages received, and possibly decide. In step k, p_i updates its variables as follows.

1. If p_i receives a DECIDE message, then p_i decides on the decision value received.
2. If STATE_i is SYNC1 or SYNC2, then the following hold:
   - p_i updates $Halt_i$ to include all processes already in $Halt_i$ and also includes the set of processes p_j such that (a) p_i has received an NSYNC message from p_j in step k, (b) p_i has received a message from p_j with $p_i \in Halt_j$ in step k, or (c) p_i has not received any message from p_j in step k.
   - p_i includes in $msgSet_i$ every message received in step k whose sender is not in $Halt_i$, and p_i computes est_i to be the minimum est value among messages in $msgSet_i$.
   - If STATE_i is SYNC2, $Halt_i$ is of size at most t, and all messages in $msgSet_i$ contain STATE = SYNC2, then p_i decides on its estimate.
   - Depending on the size h of the set $Halt_i$, p_i updates STATE_i as follows: if h is lower than the current step number, then STATE_i is set to SYNC2; else if h is at most t, then STATE_i is set to SYNC1; otherwise, STATE_i is set to NSYNC.
3. If STATE = NSYNC and p_i receives a message with STATE = SYNC2, then p_i adopts the estimate contained in that message.
4. Upon decision in round k, p_i sends the decision value to all processes in round $k + 1$ and then halts.

9.3. Correctness. The validity property of the algorithm follows from the following three simple observations: (1) the est value of a process is initialized to the proposal value of the process, (2) the est value of a process at the beginning of round $s \geq 2$ is the est value of some process at the beginning of round $s - 1$, and (3) every process decides on the est value of some process. In the rest of the section, we prove the uniform agreement property of the algorithm. We defer the proof of the termination property to the next subsection, where we prove termination along with the time-complexity property of the algorithm.

For a given session, we introduce the following notation. For every variable val_i at process p_i, we denote by $val_i[k]$ ($k \geq 1$) the value of the variable val_i immediately after the completion of step k; $val_i[0]$ denotes the value of val_i immediately before sending messages in step 1. We assume that there is a symbol *undefined* that is distinct from any possible value of the variables in the algorithm. If p_i crashes before completing step k, then $val_i[k] =$ *undefined*; if p_i crashes before sending messages in step 1, then $val_i[0] =$ *undefined*. For every process p_l that completes step k with $\text{STATE}_l[k] \in \{\text{SYNC1, SYNC2}\}$, let $senderMS_l[k]$ denote the set of processes that have sent the messages in $msgSet_l[k]$. We first prove the following lemma.

LEMMA 9.1. *Consider any session and a process p_l that completes step k with $\text{STATE}_l[k] \in \{\text{SYNC1, SYNC2}\}$. Then, $senderMS_l[k] = \Pi - Halt_l[k]$.*

Proof. Process p_l completes step k with STATE = SYNC1 or STATE = SYNC2 and hence updates $Halt$ and $msgSet$ at line 17 and line 18 of step k, respectively. Consider any process $p_m \in \Pi$. There are two cases concerning the message from p_m to p_l in step k.

- If p_l does not receive the messages from p_m in step k, then from the third condition in line 17, $p_m \in \mathit{Halt}_l[k]$, and from line 18, $p_m \notin \mathit{senderMS}_l[k]$.
- If p_l receives the step k message from p_m, then from line 18, $p_m \in \mathit{senderMS}_l[k]$ if and only if $p_m \notin \mathit{Halt}_l[k]$. □

LEMMA 9.2 (uniform agreement). *No two processes decide differently.*

Proof. If no process ever decides, then the lemma is trivially true. Thus, consider the lowest session sn in which some process decides. In session sn, consider the lowest step in which some process decides, say, step $k' + 1 \geq 2$. (It is easy to see that no process can decide in step 1 of sn.) If some process decides in line 15, then some other process has decided in a lower step of sn or in a lower session, which is a contradiction to the definition of $k' + 1$ and sn. Thus some process decides in line 21 of step $k' + 1$. We claim the following.

CLAIM 9.3 (elimination). *Consider the lowest session sn in which some process decides. If $k' + 1 \geq 2$ is the lowest step in sn in which some process decides, then, if there are two processes p_x and p_y such that* $\text{STATE}_x[k'] \in \{\text{SYNC1},\text{SYNC2}\}$ *and* $\text{STATE}_y[k'] = \text{SYNC2}$*, then* $est_x[k'] \geq est_y[k']$*.*

Proof of Lemma 9.2 *continued.* For now, we assume the above claim and prove uniform agreement. We later give a proof of Claim 9.3. Suppose that some process p_w decides d at line 21 of step $k' + 1$. From lines 19 and 20 it follows that there is a message in $\mathit{msgSet}_w[k' + 1]$ that has $\text{STATE} = \text{SYNC2}$ and $est = d$, say, from process p_v. Consider another process p_u that completes step k' with $\text{STATE} = \text{SYNC2}$ and $est = d'$. Applying Claim 9.3 twice with $p_x = p_v$ and $p_y = p_u$, and vice-versa, we get $d' = d$. It follows that, every process that completes step k' with $\text{STATE} = \text{SYNC2}$ does so with $est = d$. Notice that every process that decides at line 21 in step $k' + 1$ (that includes p_w) has only messages with $\text{STATE} = \text{SYNC2}$ in the $\mathit{msgSet}[k' + 1]$, and hence, all these messages have $est = d$. Consequently, every process that decides in line 21 of step $k' + 1$ sets its est to d in line 19 and then decides on its $est = d$ in line 21. Thus, every process that decides in step $k' + 1$ decides d. (Recall that no process can decide in line 15 of step $k' + 1$.) It remains to be shown that no process decides a different value in a higher step of sn or in a higher session.

From line 20 we have $|\mathit{Halt}_w[k' + 1]| \leq t$, and hence, Lemma 9.1 implies that $\mathit{msgSet}_w[k' + 1]$ contains at least $n - t$ messages, i.e., messages from a majority of processes. Furthermore, the last condition in line 20 requires that every message in $\mathit{msgSet}_w[k'+1]$ has $\text{STATE} = \text{SYNC2}$. Thus, a majority of the processes sent a message with $\text{STATE} = \text{SYNC2}$ in step $k' + 1$. As the round $k' + 1$ message from process p_v has $\text{STATE} = \text{SYNC2}$ and $est = d$, by applying Claim 9.3, it follows that for messages in round $k' + 1$, (1) every message with $\text{STATE} = \text{SYNC2}$ in round $k' + 1$ has $est = d$, and (2) every message with $\text{STATE} = \text{SYNC1}$ has $est \geq d$.

Now consider the est value of any process p_i at the end of step $k'+1$. If $\text{STATE}_i[k'+1] = \text{NSYNC}$, then p_i has received at least one message with $\text{STATE} = \text{SYNC2}$ and $est = d$ (because a majority of processes have sent such messages and, in every step, p_i receives messages from at least $n - t$ processes, a majority) and therefore updates its est to d (line 28). On the other hand, if $\text{STATE}_i[k' + 1] \in \{\text{SYNC1}, \text{SYNC2}\}$ then $\mathit{Halt}_i[k' + 1] \leq t$ (line 22 and line 24). Therefore, $\mathit{msgSet}_i[k' + 1]$ contains at least $n - t$ messages (Lemma 9.1). Furthermore, $\mathit{msgSet}_i[k' + 1]$ contains no message with $\text{STATE}_i[k'+1] = \text{NSYNC}$ (line 17 and line 18). Therefore, from Claim 9.3, every message in $\mathit{msgSet}_i[k' + 1]$ has $est \geq d$ and there is at least one message with $\text{STATE} = \text{SYNC2}$

and $est = d$ (because, in step $k' + 1$, a majority of processes sent messages with $\text{STATE} = \text{SYNC2}$ and $est = d$). Therefore, in line 19, p_i updates est to d.

Thus every process that completes step $k' + 1$ updates its est to d, and every process that decides in step $k' + 1$ decides d. Suppose by contradiction that some process decides a value different from d in a higher step of sn or in a higher session. Consider the lowest session sn'' and the lowest step k'' in sn'' in which some process p_j decides a value different from d, say, d''. Observe that if p_j decides in line 15 of step k'', then from line 14 it follows that some process has decided d'' in a lower step of sn'' or in a lower session. Thus p_j has decided on its est in line 21. Again observe that, given a session sn', the est value of a process at the end of some step $k \geq 2$ is the est value of some process at the end of step $k - 1$, and the est value of a process at the end of the step $k = 1$ is the est value of some process at the end of step $t + 2$ of the previous session $sn' - 1$. Therefore, the est value of any process in a step higher than $k' + 1$ in session sn, or in a higher session, cannot be different from d. Thus p_j cannot decide d'' in step k'' of sn'', which is a contradiction. □

CLAIM 9.3. *Consider the lowest session sn in which some process decides. If* $k' + 1 \geq 2$ *is the lowest step in sn in which some process decides, then, if there are two processes* p_x *and* p_y *such that* $\text{STATE}_x[k'] \in \{\text{SYNC1}, \text{SYNC2}\}$ *and* $\text{STATE}_y[k'] = \text{SYNC2}$*, then* $est_x[k'] \geq est_y[k']$.

Proof. Suppose by contradiction that there are two processes p_x and p_y such that the following holds.

Assumption A1. $\text{STATE}_x[k'] \in \{\text{SYNC1}, \text{SYNC2}\}$, $\text{STATE}_y[k'] = \text{SYNC2}$, $est_x[k'] = c$, $est_y[k'] = d$, and $c < d$.

In the context of session sn, we show Claims 9.3.1–9.3.7 based on the definition of k' and the Assumption A1. Claim 9.3.4 contradicts Claim 9.3.7, which completes the proof of Claim 9.3 by contradiction. □

Let us define the following sets for $k \in [1, k' + 1]$:

- $C[k] = \{p_i | est_i[k] \leq c\}$ (the set of processes that complete step k with $est \leq c$).
- $crashed[k]$ = the set of processes that crashed before completing step k.
- $NSYN[k] = \{p_i | \text{STATE}_i[k] = \text{NSYNC}\}$.
- $Z[k] = C[k] \cup crashed[k] \cup NSYN[k]$.

Additionally, let us define $C[0]$ to be the set of processes that start step 1 with est less than or equal to c, $crashed[0]$ to be the set of processes that crash before sending any message in step 1, $NSYN[0] = \emptyset$, and $Z[0] = C[0] \cup crashed[0] \cup NSYN[0]$. We make the following observation.

Observation A2. $|C[0]| \geq 1$, and hence, $|Z[0]| \geq 1$. Otherwise, if every process starts step 1 with a value greater than c, then $est_x[k'] > c$ (which contradicts Assumption A1).

Proof sketch. Before presenting Claims 9.3.1–9.3.7, we give a rough sketch of the overall proof. Recall that Z is the set of processes that crashed, entered state NSYNC, or have estimate less than or equal to c at the end of a step. $Z[k]$ denotes the set Z at the end of step k. We derive a contradiction on the size of set Z by showing that (1) for p_y to complete step k' with $\text{STATE} = \text{SYNC2}$ and $est = d$, we need $|Z[k' - 1]| \leq k' - 1$, but (2) for Assumption A1 to be satisfied, $|Z|$ should increase in every step, and hence, $|Z[k' - 1]| > k' - 1$.

We first note that, if a process is in set $Z[k]$, then it remains in that set in all higher steps. To see why, note that once a process crashes or enters state NSYNC, it stays in those states. In addition, if a process has $est \leq c$, then, unless it crashes or enters state NSYNC, the process updates its estimate to the lowest estimate seen in that step, which cannot be more than c.

Now, from Assumption A1, process p_y completes step k' with state SYNC2. From the algorithm, this requires that $Halt_y$ set of p_y at the end of step k' is of size at most $k'-1$. Now consider the message from a process p_j in $Z[k'-1]$ to p_y in step k'. Either p_y does not receive a message from p_j or it receives one with state NSYNC, or with $est \leq c$. In the first two cases, p_y puts p_j in its $Halt_y$ set, and the last case is not possible because it requires p_y to update its est to a value lower than d. Thus the set $Z[k'-1]$ is a subset of $Halt_y$ at the end of step k', and hence, $|Z[k'-1]| \leq k'-1$.

From the definition of Z and Assumption A1, process p_x is in $Z[k']$. We also show that p_x is not in $Z[k'-2]$. To see why, assume otherwise. Then p_x sends step $k'-1$ messages with $est \leq c$, and therefore, processes in $\Pi - Z[k'-1]$ do not receive any message from p_x (otherwise, they would update their estimate to a value at most c and hence be in set $Z[k'-1]$). Note that the number of processes that are in $\Pi - Z[k'-1]$ is more than t as we have already shown $|Z[k'-1]| \leq k'-1 \leq t < n/2$. Thus, in step k', more than t processes send messages with $p_x \in Halt$. From the algorithm, p_x puts all such processes in its $Halt_x$ set. However, a $Halt_x$ set of size more than t requires p_x to enter state NSYNC, which is a contradiction.

We next show that at least one process enters the set Z in every step (until step $k'-2$). For ease of presentation, in this proof sketch, we ignore crashed processes and processes with state NSYNC. Suppose by contradiction that no process enters the set Z in some step g; i.e., $Z[g] = Z[g+1]$. Then, arguing as above, processes in $\Pi - Z[g+1]$ do not receive any message from processes in $Z[g]$ (otherwise, they would update their estimate to a value at most c, and hence be in set $Z[g+1]$). It follows from the algorithm that, in subsequent steps, every process in $\Pi - Z[g+1] = \Pi - Z[g]$ ignores estimate values received from any process in $Z[g]$. Thus no process in $\Pi - Z[g]$ adopts an est less than or equal to c. Thus set Z does not change after round g. This contradicts our earlier observation that p_x is in $Z[k']$ but not in $Z[k'-2]$. (The actual proof of this claim is bit involved because we need to consider crashed processes and processes with state NSYNC.)

As $|Z|$ increases by at least 1 in every step until step $k'-2$, we have $|Z[k'-2]| \geq k'-1$. Using a slightly different argument we can show that $|Z|$ increases by 1 in step $k'-1$ as well. Thus, $|Z[k'-1] > k'-1$, which contradicts our earlier observation. We now give the detailed proof of the claims.

CLAIM 9.3.1. (1) For all $k \in [0, k'-1]$, $(crashed[k] \cup NSYN[k]) \subseteq (crashed[k+1] \cup NSYN[k+1])$. (2) For all $k \in [1, k']$, if $p_i \notin (NSYN[k] \cup crashed[k])$ then p_i sends messages with STATE $\in \{$SYNC1, SYNC2$\}$ in step k, and in all steps lower than k, of this session.

Proof. (1) Suppose by contradiction that there is a process p_i such that $p_i \in crashed[k] \cup NSYN[k]$ and $p_i \notin crashed[k+1] \cup NSYN[k+1]$. Since a crashed process does not recover, $crashed[k] \subseteq crashed[k+1]$, and hence, $p_i \notin crashed[k+1] \cup NSYN[k+1]$ implies that $p_i \notin crashed[k]$. Thus, $p_i \in crashed[k] \cup NSYN[k]$ implies that $p_i \in NSYN[k]$; i.e., p_i completes step k with STATE $=$ NSYNC. Notice that by the definition of k' (i.e., $k'+1$ is the lowest step in which some process decides), the conditions of line 12 and line 14 cannot be true in step $k+1 < k'+1$ for any process. Thus the STATE of p_i remains NSYNC at the end of step $k+1$, i.e., $p_i \in NSYN[k+1]$, which is a contradiction.

(2) If $p_i \notin (NSYN[k] \cup crashed[k])$, then, from Claim 9.3.1.1, it follows that $p_i \notin (NSYN[k_1] \cup crashed[k_1])$ for all $k_1 \leq k$; i.e., p_i completes every step lower than or equal to k with STATE $\neq$ NSYNC. Thus p_i has not sent any message with

STATE = NSYNC in step k or in a lower step. □

CLAIM 9.3.2. For all $k \in [0, k'-1]$, $Z[k] \subseteq Z[k+1]$.

Proof. Suppose by contradiction that there is a process p_i and some $k \in [0, k'-1]$ such that $p_i \in Z[k]$ and $p_i \notin Z[k+1]$. Since $p_i \notin Z[k+1]$, then $p_i \notin \mathit{crashed}[k+1] \cup \mathit{NSYN}[k+1]$. Applying Claim 9.3.1(1), we get $p_i \notin \mathit{crashed}[k] \cup \mathit{NSYN}[k]$. However, $p_i \in Z[k] = C[k] \cup \mathit{crashed}[k] \cup \mathit{NSYN}[k]$, and hence, $p_i \in C[k]$.

We first observe that p_i sends messages with STATE ≠ NSYNC in the first $k+1$ steps: this follows from $p_i \notin \mathit{crashed}[k+1] \cup \mathit{NSYN}[k+1]$ and Claim 9.3.1(2). As p_i always receives messages from itself and does not send any message with STATE ≠ NSYNC in the first $k+1$ steps, it follows that $p_i \notin \mathit{Halt}_i[k+1]$ (line 17). Furthermore, $p_i \notin \mathit{crashed}[k+1] \cup \mathit{NSYN}[k+1]$ implies that p_i completes step $k+1$ with STATE = SYNC1 or STATE = SYNC2. Applying Lemma 9.1 we have $p_i \in \mathit{senderMS}_i[k+1]$. Thus, the step $k+1$ message from p_i is in $\mathit{msgSet}_i[k+1]$. However, as $p_i \in C[k]$, the step $k+1$ message from p_i contains $\mathit{est}_i[k] \leq c$. Thus, when p_i evaluates *est* in line 19 of step $k+1$, p_i considers its own message with $\mathit{est}_i[k] \leq c$ and hence adopts a value less than or equal to c as $\mathit{est}_i[k+1]$. Thus $p_i \in C[k+1] \subseteq Z[k+1]$, which is a contradiction. □

CLAIM 9.3.3. For all $k \in [0, k'-1]$, for all $p_i \notin Z[k+1]$, $Z[k] \subseteq \mathit{Halt}_i[k+1]$.

Proof. Consider a process $p_j \in Z[k]$ and a process $p_i \notin Z[k+1]$. In step $k+1$, $\mathit{msgSet}_i[k+1]$ either contains a message from p_j or does not contain any message from p_j. In the second case, Lemma 9.1 implies that $p_j \in \mathit{Halt}_i[k+1]$. Consider the case where $\mathit{msgSet}_i[k+1]$ contains a message m from p_j. From line 17 and line 18, it follows that m contains STATE ≠ NSYNC, and hence, $p_j \notin \mathit{NSYN}[k]$. Furthermore, p_j has sent a message in step $k+1$, and so, $p_j \notin \mathit{crashed}[k]$. Thus $p_j \notin \mathit{crashed}[k] \cup \mathit{NSYN}[k]$, but we have assumed $p_j \in Z[k]$. So, $p_j \in C[k]$, and hence, message m from p_j contains an *est* less than or equal to c. Since $m \in \mathit{msgSet}_i[k+1]$, in step $k+1$, p_i evaluates *est* to a value less than or equal to c. Thus $p_i \in C[k+1] \subseteq Z[k+1]$, which is a contradiction. Thus $\mathit{msgSet}_i[k+1]$ does not contain any message from p_j. □

CLAIM 9.3.4. $|Z[k'-1]| \leq k'-1$.

Proof. From Assumption A1, it follows that $p_y \notin Z[k']$. Therefore, from Claim 9.3.3, $Z[k'-1] \subseteq \mathit{Halt}_y[k']$. On the other hand, $\text{STATE}_y[k'] = \text{SYNC2}$ implies that $|\mathit{Halt}_y[k']| \leq k'-1$ (line 22 and line 23). Thus, $|Z[k'-1]| \leq k'-1$. □

CLAIM 9.3.5. $p_x \in Z[k']$ and $p_x \notin Z[k'-2]$.

Proof. As $\mathit{est}_x[k'] = c$, we have $p_x \in C[k'] \subseteq Z[k']$.

For the second part of the claim, suppose by contradiction that $p_x \in Z[k'-2]$. Then, from Claim 9.3.3, for every process $p_i \notin Z[k'-1]$, $p_x \in \mathit{Halt}_i[k'-1]$. Therefore, in step k', if any process in $\Pi - Z[k'-1]$ sends a message m, then $p_x \in m.\mathit{Halt}$ (where $m.\mathit{Halt}$ denotes the *Halt* field of m). If p_x receives m in step k', then it includes the sender of m in Halt_x (because of condition 2 in line 17), and if p_i does not receive m in step k', then p_i includes the sender of m in Halt_x (because of condition 3 in line 17). Thus $\Pi - Z[k'-1] \subseteq \mathit{Halt}_x[k']$. Using Claim 9.3.4, $|\mathit{Halt}_x[k']| \geq |\Pi - Z[k'-1]| \geq n - (k'-1)$. Since $k'+1 \leq t+2$ and $t < n/2$, we have $|\mathit{Halt}_x[k']| \geq n - t > t$. However, $|\mathit{Halt}_x[k']| > t$ implies that $\text{STATE}_x[k'] = \text{NSYNC}$ (line 26 and line 27), which is a contradiction. □

CLAIM 9.3.6. (1) For all $k \in [0, k'-3]$, $Z[k] \subset Z[k+1]$. ($Z[k]$ is a proper subset of $Z[k+1]$.) (2) $|Z[k'-2]| \geq k'-1$.

Proof. (1) From Claim 9.3.2, $Z[k] \subseteq Z[k+1]$ ($k \in [0, k'-1]$). Suppose by contradiction that there is some $g \in [0, k'-3]$ such that $Z[g] = Z[g+1]$.

We first show by induction on the step number k that, for all $k \in [g+1, k'-1]$, $C[k] - (\mathit{NSYN}[k] \cup \mathit{crashed}[k]) \supseteq C[k+1] - (\mathit{NSYN}[k+1] \cup \mathit{crashed}[k+1])$. (This statement corresponds to the brief argument presented in the proof sketch where we showed that if we ignore crashed processes and processes with NSYNC state, then the set Z does not increase after step g.)

Base case $(k = g+1)$. $C[g+1] - (NSYN[g+1] \cup crashed[g+1]) \supseteq C[g+2] - (NSYN[g+2] \cup crashed[g+2])$. Suppose by contradiction that there is a process p_i such that $p_i \in C[g+2] - (NSYN[g+2] \cup crashed[g+2])$ (*Assumption* A3) and $p_i \notin C[g+1] - (NSYN[g+1] \cup crashed[g+1])$ (*Assumption* A4).

Assumption A3 implies that $p_i \notin NSYN[g+2] \cup crashed[g+2]$. Applying Claim 9.3.1.1, we have $p_i \notin NSYN[g+1] \cup crashed[g+1]$, and therefore, from Assumption A4, it follows that $p_i \notin C[g+1]$. Thus p_i completes step $g+1$ with $est > c$ and STATE $\neq$ NSYNC. Furthermore, Assumption A3 implies that p_i completes step $g+2$ with $est \leq c$ and STATE $\neq$ NSYNC. So, $msgSet_i[g+2]$ contains a message with $est \leq c$ from some process p_j, i.e., $p_j \in senderMS_i[g+2]$ (*Observation* A5). As p_j sends a message with $est \leq c$ in step $g+2$, it follows that $p_j \in C[g+1] \subseteq Z[g+1]$.

As $p_i \notin NSYN[g+1] \cup crashed[g+1]$ and $p_i \notin C[g+1]$, from the definition of $Z[g+1]$ we have $p_i \notin Z[g+1]$. Claim 9.3.3 implies that $Z[g] \subseteq Halt_i[g+1]$. Recall that we assumed $Z[g] = Z[g+1]$ and, from line 17, $Halt_i[g+1] \subseteq Halt_i[g+2]$. Therefore, $Z[g+1] \subseteq Halt_i[g+2]$. Thus $p_j \in C[g+1] \subseteq Z[g+1]$ implies that $p_j \in Halt_i[g+2]$. From Observation A5, $p_j \in senderMS_i[g+2] \cap Halt_i[g+2]$.

As $p_i \notin NSYN[g+2] \cup crashed[g+2]$, it follows that p_i completed step $g+2$ with STATE = SYNC1 or STATE = SYNC2. From Lemma 9.1 it follows that $senderMS_i[g+2] \cap Halt_i[g+2] = \emptyset$. However, $p_j \in senderMS_i[g+2] \cap Halt_i[g+2]$, which is a contradiction.

Induction hypothesis $(k \in [g+1, r])$. $C[k] - (NSYN[k] \cup crashed[k]) \supseteq C[k+1] - (NSYN[k+1] \cup crashed[k+1])$ for some $r < k' - 1$.

Induction step $(k = r+1)$. $C[r+1] - (NSYN[r+1] \cup crashed[r+1]) \supseteq C[r+2] - (NSYN[r+2] \cup crashed[r+2])$. Suppose by contradiction that there is a process p_i such that $p_i \in C[r+2] - (NSYN[r+2] \cup crashed[r+2])$ (*Assumption* A6) and $p_i \notin C[r+1] - (NSYN[r+1] \cup crashed[r+1])$ (*Assumption* A7).

Similar to the base case, applying Assumptions A6 and A7, and Claim 9.3.1 gives us $p_i \notin NSYN[r+2] \cup crashed[r+2]$, $p_i \notin NSYN[r+1] \cup crashed[r+1]$, and $p_i \notin C[r+1]$. Thus $p_i \notin Z[r+1]$. Since $g+1 < r+1$, from Claim 9.3.2, we have $Z[g+1] \subseteq Z[r+1]$, and therefore, $p_i \notin Z[g+1]$.

Applying Claim 9.3.3 on $p_i \notin Z[g+1]$ implies that $Z[g] \subseteq Halt_i[g+1]$. Recall that we assumed $Z[g] = Z[g+1]$, and from line 17 and $g+1 < r+2$, $Halt_i[g+1] \subseteq Halt_i[r+2]$. Therefore, $Z[g+1] \subseteq Halt_i[r+2]$ (*Observation* A8).

From the induction hypothesis, we have $(C[g+1] - (NSYN[g+1] \cup crashed[g+1])) \supseteq (C[r+1] - (NSYN[r+1] \cup crashed[r+1]))$. From the definition of $Z[g+1]$, $C[g+1] - (NSYN[g+1] \cup crashed[g+1]) \subseteq C[g+1] \subseteq Z[g+1]$, and therefore, $C[r+1] - (NSYN[r+1] \cup crashed[r+1]) \subseteq Z[g+1]$. Applying Observation A8, we have $(C[r+1] - (NSYN[r+1] \cup crashed[r+1])) \subseteq Halt_i[r+2]$ (*Observation* A9).

As $p_i \notin Z[r+1]$, p_i completes step $r+1$ with $est > c$ and STATE $\neq$ NSYNC. Furthermore, Assumption A6 implies that p_i completes step $r+2$ with $est \leq c$ and STATE $\neq$ NSYNC. Therefore, $msgSet_i[r+2]$ contains a message with $est \leq c$ from some process p_j, i.e., $p_j \in senderMS_i[r+2]$ (*Observation* A10). As p_j sends a message with $est \leq c$ in step $r+2$, $p_j \in C[r+1] \subseteq Z[r+1]$.

As the step $r+2$ message of p_j is in $msgSet_i[r+2]$, from line 17 it follows that the message sent by p_j had STATE $\neq$ NSYNC. Therefore, $p_j \notin NSYN[r+1]$, and clearly, $p_j \notin crashed[r+1]$. Therefore, $p_j \in C[r+1] - (NSYN[r+1] \cup crashed[r+1])$.

From Observation A9 it follows that $p_j \in Halt_i[r+2]$. From Observation A10, $p_j \in senderMS_i[r+2] \cap Halt_i[r+2]$.

As $p_i \notin NSYN[r+2] \cup crashed[r+2]$ (from Assumption A6), p_i completed step $r+2$ with STATE = SYNC1 or STATE = SYNC2. Lemma 9.1 implies that $senderMS_i[r+2] \cap Halt_i[r+2] = \emptyset$. However, $p_j \in senderMS_i[r+2] \cap Halt_i[r+2]$, which is a contradiction.

From the above result (that we proved by induction), we have $C[k'-2]-(NSYN[k'-2] \cup crashed[k'-2]) \supseteq C[k'] - (NSYN[k'] \cup crashed[k'])$. From Assumption A1, $p_x \in C[k'] - (NSYN[k'] \cup crashed[k']))$. From Claim 9.3.5, we have $p_x \notin Z[k'-2] \supseteq (C[k'-2] - (NSYN[k'-2] \cup crashed[k'-2])$. In other words, p_x is in $C[k'] - (NSYN[k'] \cup crashed[k'])$ but not in $C[k'-2]-(NSYN[k'-2] \cup crashed[k'-2])$, which is a contradiction.

(2) Part (1) of this claim implies that for every $k \in [0, k'-3]$, $|Z[k+1]|-|Z[k]| \geq 1$. From Observation A2, $|Z[0]| \geq 1$. Therefore, $|Z[k'-2]| \geq k'-1$. □

CLAIM 9.3.7. $|Z[k'-1]| > k'-1$.

Proof. Suppose by contradiction that $|Z[k'-1]| \leq k'-1$. Since $Z[k'-2] \subseteq Z[k'-1]$ (Claim 9.3.2) and $|Z[k'-2]| \geq k'-1$ (Claim 9.3.6.2), we have $Z[k'-2] = Z[k'-1]$ and $|Z[k'-2]| = |Z[k'-1]| = k'-1$ (*Assumption* A11).

From Claim 9.3.5, we know that $p_x \notin Z[k'-2] = Z[k'-1]$. Applying Claim 9.3.3, we have $Z[k'-2] \subseteq Halt_x[k'-1]$. As $Z[k'-2] = Z[k'-1]$ (from Assumption A11), it follows that $Z[k'-1] \subseteq Halt_x[k'-1]$.

Since $p_x \notin Z[k'-1]$, p_x completes step $k'-1$ with $est > c$ and STATE $\neq$ NSYNC. From Assumption A1, we also know that p_x completes step k' with $est \leq c$ and STATE $\neq$ NSYNC. Therefore, $msgSet_x[k']$ contains a message, say, from process p_j, with $est \leq c$, i.e., $p_j \in senderMS_x[k']$. From the definition of $C[k'-1]$, $p_j \in C[k'-1] \subseteq Z[k'-1]$. However, we showed earlier that $Z[k'-1] \subseteq Halt_x[k'-1]$, and from line 17, it follows that $Halt_x[k'-1] \subseteq Halt_x[k']$. Thus $Z[k'-1] \subseteq Halt_x[k']$ and $p_j \in Halt_x[k']$.

From Assumption A1, we know that p_x completed step k' with STATE = SYNC1 or STATE = SYNC2. Therefore, Lemma 9.1 implies that $senderMS_x[k'] \cap Halt_x[k'] = \emptyset$. However, $p_j \in senderMS_x[k'] \cap Halt_x[k']$, which is a contradiction. □

9.4. Time-complexity. We now discuss the termination and the time-complexity of the algorithm. From the definition of ES_t, for every run R in ES_t, there is an unknown round number GSR from which the system is *synchronous* (eventual synchrony property of ES_t). Define a *synchronous session* as a session that starts in round GSR or in a higher round. Let sn be the lowest synchronous session, and let f be the number of processes that crash in R.

LEMMA 9.4. *Consider any process p_i that completes step $k \in [1, t+2]$ of session sn. If no correct process decides before step k, then every process in $Halt_i[k]$ has crashed by step k.*

Proof. Suppose no correct process decides before step k in session sn. For every step $l \in [0, k]$ in sn, let $H[l]$ be the union of all $Halt_j[l]$ such that $Halt_j[l] \neq$ *undefined*. We claim the following, which immediately implies the lemma: *Every process in $H[l]$ (for all $l \in [0, k]$) crashes by step l.*

We prove the claim by induction on step number l. For $l = 0$, the claim is trivially true, because $H[0] = \emptyset$ (base case). Suppose that the claim is true for $l \in [0, l'-1]$ (for some $l'-1 \leq k-1$): every process in $H[l]$ crashes by step l (induction hypothesis). Consider the set $H[l']$ (induction step). If $H[l']-H[l'-1] = \emptyset$ then the induction step is trivial. Suppose by contradiction that there is a process $p_j \in H[l']-H[l'-1]$ such that

p_j has not crashed by step l'. Thus there is a process p_a such that $p_j \notin Halt_a[l'-1]$ and $p_j \in Halt_a[l']$.

As p_j has not crashed by step l', no correct process has decided before round k, and sn is a synchronous session, so p_a must have received the step l' message m of p_j. Since, $p_j \in Halt_a[l']$, m contains either (a) STATE = NSYNC or (b) set $Halt_j$ such that $p_a \in Halt_j$. Now, we show both cases to be impossible and thus prove the induction step by contradiction.

From our induction hypothesis, for every step $l'' < l'$, every process in $Halt_j[l'']$ has crashed by step l''. Since no more than t processes can crash in a run, in rounds lower than l', $|Halt_j|$ is never higher than t. Thus p_j cannot update its STATE to NSYNC in rounds lower than l' (line 26). Thus the round l' message from p_j does not contain STATE = NSYNC.

If the round l' message from p_j contains $Halt_j$, such that $p_a \in Halt_j$, then $p_a \in Halt_j[l'-1] \subseteq H[l'-1]$. However, from our induction hypothesis, every process in $H[l'-1]$ has crashed before completing round $l'-1$, which implies that p_a has crashed before completing round $l'-1$, which is a contradiction. □

LEMMA 9.5 (time-complexity). *In every run of the algorithm in SCS_f (for any $f \in [0,t]$), every process that decides does so by round $f+2$.*

Proof. Consider any run R of the algorithm in SCS_f. (Note that, for a run in SCS_f, the first session is synchronous.) If some correct process decides by round $f+1$, then every process receives a DECIDE message (and decides) by round $f+2$. Therefore, suppose by contradiction that no correct process decides by round $f+1$ in R, and some correct process p_i completes round $f+2$ without deciding.

Since at most f processes may crash in R, from Lemma 9.4, in every round, $|Halt|$ at every alive process is less than or equal to f. As p_i does not decide in round $f+2$ and $|Halt_i[f+2]| \leq f$, one of the following is true: (1) STATE$_i[f+1]$ = NSYNC, (2) STATE$_i[f+1]$ = SYNC1, or (3) some other process p_j sent a message in round $f+2$ with STATE = SYNC1. Case 1 requires $|Halt_i| > t$ in round $f+1$ or in a lower round (line 26), which is a contradiction. Cases 2 and 3 are not possible because $|Halt[f+1]| \leq f$ at p_i and p_j, and therefore, p_i and p_j set STATE to SYNC2 in round $f+1$. □

LEMMA 9.6 (termination). *Every correct process eventually decides.*

Proof. Suppose by contradiction that some correct process p_i does not decide in a run R. If some correct process decides, then every correct process receives a DECIDE message and decides. Thus, no correct process decides. Consider the lowest synchronous session sn. Since no correct process decides in R, from Lemma 9.4, in every step, $|Halt|$ at every alive process in session sn is less than or equal to t (as t is the maximum number of processes that may crash in R).

As p_i does not decide by step $t+2$ of session sn, from line 20, one of the following is true: (1) STATE$_i[t+1]$ = NSYNC, (2) STATE$_i[t+1]$ = SYNC1, or (3) some other process p_j sent a message in step $t+2$ with STATE = SYNC1. Case 1 requires $|Halt_i| > t$ in step $t+1$ or in a lower step (line 26),which is a contradiction. Cases 2 and 3 are not possible because $|Halt[t+1]| \leq t$ at p_i and p_j, and therefore, p_i and p_j set STATE to SYNC2 in round $t+1$. □

9.5. Eventually synchronous results summary. Combining Proposition 8.1, the global decision lower bounds in [19, 9], and the time-complexity of algorithm A_{es}, we get the following tight bounds in eventually synchronous models:

1. $\forall t \in [1,(n-1)/2]$, $\forall f \in [0, t-3]$, $(ES_t,\ SCS_f,\ \text{UC},\ ld) = f+2$. Local decision bound for uniform consensus.

2. $\forall t \in [1, (n-1)/2]$, $\forall f \in [0, t]$, $(ES_t,\ SCS_f,\ \text{UC},\ gd) = f + 2$. Global decision bound for uniform consensus.

10. Concluding remarks.

The time-complexity of local decisions is a natural measure in many agreement-based distributed systems. As pointed out in the introduction, in a replication or a transactional system, it may be sufficient for a client to receive the decision value from any process executing the agreement algorithm. Besides, studying the local decision metric helps uncover fundamental differences between problems and between models that were not apparent with other metrics. For example, in a synchronous model, uniform consensus and nonblocking atomic commit have the same tight bound in terms of global decision but have different bounds when we consider local decision. Similarly, considering a local decision metric allows us to infer that early deciding uniform consensus algorithms are faster in a synchronous model than in synchronous runs of an eventually synchronous model.

REFERENCES

[1] M. K. Aguilera and S. Toueg, *A simple bivalency proof that t-resilient consensus requires $t+1$ rounds*, Inform. Process. Lett., 71 (1999), pp. 155–158.

[2] T. D. Chandra and S. Toueg, *Unreliable failure detectors for reliable distributed systems*, J. ACM, 43 (1996), pp. 225–267.

[3] B. Charron-Bost and F. Le Fessant, *Validity conditions in agreement problems and time complexity*, in SOFSEM, Lecture Notes in Comput. Sci. 2932, Springer-Verlag, New York, 2004, pp. 196–207.

[4] B. Charron-Bost and A. Schiper, *Uniform consensus is harder than consensus*, J. Algorithms, 51 (2004), pp. 15–37.

[5] D. Dolev, C. Dwork, and L. J. Stockmeyer, *On the minimal synchronism needed for distributed consensus*, J. ACM, 34 (1987), pp. 77–97.

[6] D. Dolev, R. Reischuk, and H. R. Strong, *Early stopping in Byzantine agreement*, J. ACM, 37 (1990), pp. 720–741.

[7] D. Dolev and H. R. Strong, *Authenticated algorithms for Byzantine agreement*, SIAM J. Comput., 12 (1983), pp. 656–666.

[8] P. Dutta, *Time-Complexity Bounds on Agreement Problems*, Ph.D. thesis, Thesis 3261, Swiss Federal Institute of Technology, Lausanne (EPFL), Switzerland, 2005.

[9] P. Dutta and R. Guerraoui, *The inherent price of indulgence*, Distributed Computing, 18 (2005), pp. 85–98.

[10] P. Dutta, R. Guerraoui, and B. Pochon, *Fast non-blocking atomic commit: An inherent trade-off*, Inform. Process. Lett., 91 (2004), pp. 195–200.

[11] C. Dwork, N. A. Lynch, and L. J. Stockmeyer, *Consensus in the presence of partial synchrony*, J. ACM, 35 (1988), pp. 288–323.

[12] C. Dwork and Y. Moses, *Knowledge and common knowledge in a Byzantine environment: Crash failures*, Inform. and Comput., 88 (1990), pp. 156–186.

[13] M. J. Fischer and N. A. Lynch, *A lower bound for the time to assure interactive consistency*, Inform. Process. Lett., 14 (1982), pp. 183–186.

[14] M. J. Fischer, N. A. Lynch, and M. Paterson, *Impossibility of distributed consensus with one faulty process*, J. ACM, 32 (1985), pp. 374–382.

[15] J. Gray, *Notes on data base operating systems*, in Advanced Course: Operating Systems, Lecture Notes in Comput. Sci. 60, Springer-Verlag, New York, 1978, pp. 393–481.

[16] R. Guerraoui, *Revisiting the relationship between non-blocking atomic commitment and consensus*, in WDAG, Lecture Notes in Comput. Sci. 972, Springer-Verlag, New York, 1995, pp. 87–100.

[17] V. Hadzilacos, *Byzantine Agreement under Restricted Types of Failures (Not Telling the Truth Is Different from Telling Lies)*, Tech. report 19-83, Aiken Computation Laboratory, Harvard University, Boston, MA, 1983.

[18] V. Hadzilacos, *On the relationship between the atomic commitment and consensus problems*, in Fault-Tolerant Distributed Computing, Lecture Notes in Comput. Sci. 448, Springer-Verlag, New York, 1986, pp. 201–208.

[19] I. Keidar and S. Rajsbaum, *A simple proof of the uniform consensus synchronous lower bound*, Inform. Process. Lett., 85 (2003), pp. 47–52.

[20] L. Lamport, *The part-time parliament*, ACM Trans. Comput. Syst., 16 (1998), pp. 133–169.

[21] L. Lamport and M. J. Fischer, *Byzantine Generals and Transaction Commit Protocols*, Tech. report 62, SRI International, Menlo Park, CA, 1982.

[22] L. Lamport, R. E. Shostak, and M. C. Pease, *The Byzantine generals problem*, ACM Trans. Program. Lang. Syst., 4 (1982), pp. 382–401.

[23] N. A. Lynch, *Distributed Algorithms*, Morgan Kaufmann, San Francisco, CA, 1996.

[24] Y. Moses and S. Rajsbaum, *A layered analysis of consensus*, SIAM J. Comput., 31 (2002), pp. 989–1021.

[25] Y. Moses and M. R. Tuttle, *Programming simultaneous actions using common knowledge*, Algorithmica, 3 (1988), pp. 121–169.

[26] A. Mostéfaoui and M. Raynal, *Solving consensus using Chandra-Toueg's unreliable failure detectors: A general quorum-based approach*, in DISC, Lecture Notes in Comput. Sci. 1693, Springer-Verlag, New York, 1999, pp. 49–63.

[27] M. C. Pease, R. E. Shostak, and L. Lamport, *Reaching agreement in the presence of faults*, J. ACM, 27 (1980), pp. 228–234.

[28] A. Schiper, *Early consensus in an asynchronous system with a weak failure detector*, Distributed Computing, 10 (1997), pp. 149–157.

[29] D. Skeen, *Nonblocking commit protocols*, in ACM SIGMOD International Conference on Management of Data, ACM, New York, 1981, pp. 133–142.

SIAM J. COMPUT.
Vol. 37, No. 3, pp. 757–782

A NEARLY LINEAR-TIME APPROXIMATION SCHEME FOR THE EUCLIDEAN *k*-MEDIAN PROBLEM*

STAVROS G. KOLLIOPOULOS[†] AND SATISH RAO[‡]

Abstract. This paper provides a randomized approximation scheme for the k-median problem when the input points lie in the d-dimensional Euclidean space. The worst-case running time is $O(2^{O((\log(1/\epsilon)/\varepsilon)^{d-1})} n \log^{d+6} n)$, which is nearly linear for any fixed ε and d. Moreover, our method provides the first polynomial-time approximation scheme for k-median and uncapacitated facility location instances in d-dimensional Euclidean space for any fixed $d > 2$. Our work extends techniques introduced originally by Arora for the Euclidean traveling salesman problem (TSP). To obtain the improvement we develop a structure theorem to describe hierarchical decomposition of solutions. The theorem is based on an *adaptive decomposition* scheme, which guesses at every level of the hierarchy the structure of the optimal solution and accordingly modifies the parameters of the decomposition. We believe that our methodology is of independent interest and may find applications to further geometric problems.

Key words. approximation algorithms, approximation schemes, k-median, facility location, Euclidean space, linear time

AMS subject classifications. 68Q25, 90B10, 90B12, 90B35

DOI. 10.1137/S0097539702404055

1. Introduction.

In the *k-median* problem we are given a set N of n points in a metric space and a positive integer k. The objective is to locate k *medians* (facilities) among the points so that the sum of the distances from each point in N to its closest median is minimized. The version of the problem we study is sometimes called the discrete k-median: the facilities must be located at input points. In the continuous version facilities may be located anywhere in the underlying space. The k-median problem is a well-studied, NP-hard problem which falls into the general class of *clustering* problems: partition a set of points into clusters so that the points within a cluster are close to each other with respect to some appropriate measure. Moreover, k-median is closely related to *uncapacitated facility location,* a basic problem in the operations research literature (see, e.g., [13]). In the latter problem, in addition to the set N of points, we are given also a cost c_i for *opening* a facility at point i. The objective is to open an unspecified number of facilities at a subset of N so as to minimize the sum of the cost to open the facilities (*facility cost*) plus the cost of assigning each point to the nearest open facility (*service cost*). In this paper we provide a fast approximation scheme for the problem when the input lies in a Euclidean space.

A *ρ-approximation algorithm* for a minimization problem, $\rho > 1$, computes in time polynomial in the input size a feasible solution of cost at most ρ times the optimum. An *approximation scheme* computes, for any fixed $\varepsilon > 0$, a $(1+\varepsilon)$-approximate feasible solution in time polynomial in the input size and $1/\varepsilon$.

[*] Received by the editors March 14, 2002; accepted for publication (in revised form) January 5, 2007; published electronically July 4, 2007.

http://www.siam.org/journals/sicomp/37-3/40405.html

[†] Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Athens, Greece (sgk@di.uoa.gr, http://www.di.uoa.gr/∼sgk). The work of this author was partially supported by the European Social Fund and National Resources - (EPEAEK-II) ARCHIMIDIS II.

[‡] Computer Science Division, University of California Berkeley, Berkeley, CA 94720 (satishr@cs.berkeley.edu).

1.1. Previous work. The succession of results for k-median is as follows. Lin and Vitter [23] used their filtering technique to obtain a solution of cost at most $(1+\varepsilon)$ times the optimum but using $(1+1/\varepsilon)(\ln n+1)k$ medians. They later refined their technique to obtain a solution of cost $2(1+\varepsilon)$ while using at most $(1+1/\varepsilon)k$ medians [22]. The first nontrivial approximation algorithm that achieves feasibility as well, i.e., uses k medians, combined the powerful randomized algorithm by Bartal for approximation of metric spaces by trees [5, 6] with an approximation algorithm by Hochbaum for k-median on trees [19]. The ratio thus achieved is $O(\log n \log\log n)$. This algorithm was subsequently refined and derandomized by Charikar et al. [8] to obtain a guarantee of $O(\log k \log\log k)$. Charikar and Guha and independently Tardos and Shmoys reported the first constant-factor approximations [10]. In contrast, the uncapacitated facility location problem, in which there is no a priori constraint on the number of facilities, seems to be better understood. Shmoys, Tardos, and Aardal [27] gave a 3.16 approximation algorithm. This was later improved by Guha and Khuller [16] to 2.408 and to 1.736 by Chudak [12]. After a preliminary, and by now obsolete, abstract[1] of this work appeared in [21] some additional results on k-median and facility location appeared by Charikar and Guha [9] and Jain and Vazirani [20] and on the local search approach by Arya et al. [4]. Follow-up work on the Euclidean case includes that by Har-Peled and Mazumdar [18], and work focusing on improving the dependence on the dimension includes that by Chen [11].

In this paper we focus on the case when the underlying metric is Euclidean. Until the work of Arora, Raghavan, and Rao [3], this case was not known to be any easier to approximate than a general metric. These authors gave a randomized polynomial-time approximation scheme for k-median when the points lie on the Euclidean plane [3]. For any fixed $\varepsilon > 0$, their algorithm outputs a $(1+\varepsilon)$-approximation with probability $1-o(1)$ and runs in $O(nkn^{O(1/\varepsilon)}\log n)$ time in the worst case. For facility location they gave an approximation scheme with running time $O(n^{1+O(1/\varepsilon)}\log n)$. This development followed the breakthrough approximation schemes of Arora [2] for the traveling salesman problem (TSP) and other geometric problems. While the work in [3] used techniques from the TSP approximation scheme, the different structure of the optimal solutions for k-median and TSP necessitated the development of a new structure theorem to hierarchically decompose solutions. We elaborate further on this issue during the exposition of our results in the next paragraph. The dependence of the running time achieved by the methods of Arora, Raghavan, and Rao on $1/\varepsilon$ is particularly high. For example, the approximation scheme can be extended to higher-dimensional instances but runs in quasi-polynomial time $O(n^{(\log n/\varepsilon)^{d-2}})$ for a set of points in R^d with fixed $d > 2$.

1.2. Results and techniques. Results. We provide a randomized approximation scheme for k-median on the Euclidean plane. For any fixed $\varepsilon > 0$, our scheme outputs in expectation a $(1+\varepsilon)$-approximate solution, in time

$$O\left(2^{O\left(\frac{\log(1/\varepsilon)}{\varepsilon}\right)} n \log^8 n\right)$$

in the worst case.[2] Our time bound represents a drastic improvement on the result in [3]. For any fixed accuracy ε desired, the dependence of the running time on $1/\varepsilon$

[1] In the conference version [21] we had erroneously claimed a running time of $O(2^{1/\varepsilon^d} n \log n \log k)$.

[2] For large ε, the term $\frac{\log(1/\varepsilon)}{\varepsilon}$ should be interpreted throughout the paper as $\frac{\max\{\log(1/\varepsilon),\Theta(1)\}}{\varepsilon}$. We omit the $\Theta(1)$ factor to avoid cumbersome notation.

translates to a (large) constant hidden in the near-linear asymptotic bound $O(n \log^8 n)$ compared to the exponent of a term polynomial in n in the bound of Arora, Raghavan, and Rao. Moreover, for inputs in R^d, our algorithm extends to yield a running time of

$$O\left(2^{O\left(\left(\frac{\log(1/\varepsilon)}{\varepsilon}\right)^{d-1}\right)} n \log^{d+6} n\right),$$

which yields for the first time a polynomial-time approximation scheme for any fixed $d > 2$. The ideas behind the new k-median algorithm yield also improved, nearly linear-time, approximation schemes for uncapacitated facility location. Our time bounds for the latter problem hold under the assumption that a polynomial in n-approximation is available for the value of the service cost. An example of such a case is when all the interpoint distances are polynomially related. We now elaborate on the techniques we use to obtain our results.

Techniques. Our main motivation for improving the running times of [3] was to gain a better understanding of approximation schemes for geometric problems and in particular the issues arising from Euclidean facility location problems. The actual running time we obtain is mainly of theoretical interest. We believe that the main contribution of the present paper lies in the new ideas we introduce to overcome the limitations of the approach employed by Arora, Raghavan, and Rao in [3]. To describe these ideas we sketch first some previous developments, starting with the breakthrough results by Arora [2] (see also the work of Mitchell [24] for a different approximation scheme for the Euclidean TSP).

A basic building block for Arora's results on TSP [2] was a structure theorem providing insight into how much the cost of an optimal tour could be affected in the following situation. Roughly speaking, the plane is recursively dissected into a collection of rectangles of geometrically decreasing area, represented by a quadtree data structure. For every box in the dissection one places a fixed number, dependent on the desired accuracy ε, of equidistant *portals* on the boundary of the box. The optimal TSP tour can cross between adjacent rectangles any number of times; a *portal-respecting* tour is allowed to cross only at portals. How bad can the cost of a deflected, portal-respecting tour be compared to the optimum? Implicitly, Arora used a charging argument on the edges in an optimal solution to show that the edges could be made to be portal-respecting. We now sketch his approach, which was made explicit and applied to k-median in [3].

Given a set of at most k open facilities, a k-median solution is a set of edges assigning every point to an open facility. A portal-respecting solution is one in which the assignment edges are actually paths that cross rectangle boundaries only at portals. We can assume that the input is surrounded by a rectangle with sidelength polynomial in n (cf. section 2). At level i of the dissection, the rectangles at this level with sidelength 2^i are cut by vertical and horizontal lines into rectangles of sidelength 2^{i-1}. The x- and y-coordinates of the dissection are randomly shifted at the beginning so that the probability that an edge e in a solution is cut at level i is $O(\text{length}(e)/2^i)$. Let m denote the number of portals along the dissection lines. If e is cut at level i, it must be deflected through a portal, paying additional cost $O(2^i/m)$. Summing over all the $O(\log n)$ levels of the decomposition, the expected deflection cost of edge e in a portal-respecting solution is at most

$$\sum_{i=1}^{O(\log n)} O\left(\frac{\text{length}(e)}{2^i}(2^i/m)\right). \tag{1}$$

Selecting $m = \Theta(\log n/\varepsilon)$ and applying the dissection to the optimal solution, we obtain the existence of a portal-respecting solution of cost $(1+\varepsilon)OPT$. Once the existence has been shown, dynamic programming can be used to compute the best portal-respecting solution. The running time of the dynamic programming contains a $k2^{O(m)}$ term; hence we have the $kn^{1/\varepsilon}$ term in the overall running time.

Arora additionally used a "patching lemma" argument to show that the TSP tour could be made to cross each box boundary $O(1/\epsilon)$ times. This yielded an $O(n(\log n)^{O(1/\varepsilon)})$ time algorithm for the TSP. (This running time was subsequently improved by Rao and Smith to $O(2^{O(1/\varepsilon^2)} + n \log n)$ [25] while still using $\Theta(\log n/\varepsilon)$ portals.) The k-median method did not, however, succumb to a patching lemma argument; thus the running time for the algorithms in [3] remained $O(kn^{O(1/\epsilon)})$.

Our method reduces the number m of portals to $O(\log(1/\epsilon)/\epsilon)$. *That is, we remove the* $\log n$ *factor in the number of portals that appears to be inherent in Arora, Raghavan, and Rao's charging based methods and even in Arora's charging plus patching based methods.*

Adaptive dissection. We outline some of the ideas behind the reduced value for m. The computation in (1) exploits linearity of expectation by showing that the "average" dissection line cutting an edge is short enough. The complicated dependencies among the dissection lines across all $O(\log n)$ levels seem too complicated to reason about directly. On the other hand, when summing the expectations across all levels an $O(\log n)$ factor creeps in, which apparently has to be offset by setting m to $\log n/\varepsilon$. We provide a new structure theorem to characterize the structure of near-optimal solutions. In contrast to previous approaches, given a rectangle at some level in the decomposition, it seems a good idea to choose several possible "cuts" hoping that one of them will hit a small number of segments from the optimum solution. This approach gives rise to the *adaptive dissection* idea, in which the algorithm "guesses" the structure of the part of the solution contained in a given rectangle and tunes accordingly the generation of the subrectangles created for the next level of the dissection. In the k-median problem the guess consists of the area of the rectangle which is empty of facilities. Let L be the maximum sidelength of the subrectangle containing facilities. Cutting close to the middle of this subrectangle with a line of length L should, in a probabilistic sense, mostly dissect segments from the optimal solution of length $\Omega(L)$, forcing them to deflect by $L/m = \varepsilon L$. A number of complications arise by the fact that a segment may be cut by both horizontal and vertical dissection lines. We note that the cost of "guessing" the empty area is incorporated into the size of the dynamic programming lookup table by trying all possible configurations. Given the preeminence of recursive dissection in approximation schemes for Euclidean problems [2, 3, 25] we believe that the adaptive dissection technique is of independent interest and may prove useful in other geometric problems as well.

Although the adaptive dissection technique succeeds in reducing the required number of portals to $\Theta(\log(1/\varepsilon)/\varepsilon)$ and thus asymptotically improves the dependence of the running time on $1/\varepsilon$, the dynamic program has still to enumerate all possible rectangles. Compared to the algorithm in [3], we apparently have to enumerate even more rectangles due to the "guess" for the areas without facilities and some amount of randomization we introduce in the choice of subrectangle boundaries. We bound the size of the lookup table by showing that the boundaries of the possible rectangles can be appropriately spaced and still capture the structure of a near-optimal solution.

The outline of the paper is as follows. In section 2 we give definitions and preprocessing steps. In section 3 we prove the new structure theorem and obtain the reduced number of portals. In section 4 we provide a modified structure theorem which yields a small size for the dynamic program table. In section 5 we present the dynamic program and some extensions of the main result.

2. Preliminaries. An *edge* (u, v) is a line segment connecting input points u and v. Given a selection of open facilities, an *assignment edge* is an edge (u, v) such that exactly one of u or v is an open facility. An *assignment* is a set E of assignment edges such that every point which does not host a facility appears exactly once as an endpoint. In order to minimize the cost every point must of course be assigned to its closest open facility. The *sidelength* of a rectangle with sides parallel to the axes is the length of its largest side. For any two points p, p' their Euclidean distance is denoted by $d(p, p')$.

We assume that the input points are on a unit grid of size polynomial in the number of input points. This assumption is enforced by a preprocessing phase described in [3]; the preprocessing incurs an additive error of $O(1/n^c)$ times the optimal value for some constant $c > 0$. The assumption is needed to ensure the depth of the recursive dissection is $O(\log n)$. Within the same additive error we can assume that no two input points lie on the same vertical grid line. The latter assumption simplifies the presentation.

In [3] it is shown that if a polynomial-factor approximation is available for the value of the service cost, i.e., a range $[D, n^c D]$ where this value must lie, then the above assumptions on the points can be enforced by a simple plane sweep. An algorithm to compute this range for the k-median problem is the 2-approximation minmax clustering algorithm of Gonzalez [15]. A faster algorithm for the same problem was given by Feder and Greene [14]. The latter runs in $O(n \log k)$ time on the plane. The total preprocessing time for $d = 2$ is $O(n \log n)$. For general dimension d the preprocessing time is $O(dn \log n + kd^d \log(kd^d))$ [14]. A faster 2-approximation algorithm that runs in $O(n + k \log k)$ with high probability was given by Har-Peled in [17].

3. The Structure Theorem. In this section we prove our basic Structure Theorem that shows the existence of approximately optimal solutions with a simple structure. This theorem can yield directly a dynamic programming algorithm whose running time dependence on ε is no worse than $2^{O(\log(1/\varepsilon)/\varepsilon)}$. Our exposition focuses on 2-dimensional Euclidean instances. It is easy to generalize to d dimensions. Given a set of points N and a set of facilities $F \subset N$, we define the *greedy cost under* F to be the cost of assigning each point to its closest facility. If F is the set of facilities open in the optimal solution, the greedy cost under F is the optimal cost.

We proceed to define a recursive randomized decomposition. The decomposition uses two processes: the Subrectangle process, and the Cutrectangle process.

Subrectangle:
Input: a rectangle B containing at least one facility.
Process: Find the minimal rectangle B' containing all the facilities. Let its maximum sidelength be s. Grow the rectangle by $s/3$ in each dimension. We call the grown rectangle B''.
Output: $B_s = B'' \cap B$.

Notice that $B - B_s$ contains no facilities. If $B_s \neq B$, we call B_s a *proper subrectangle.*

Cutrectangle:
Input: a rectangle B containing at least one facility.
Process: Randomly cut the rectangle into two rectangles with a line that is orthogonal

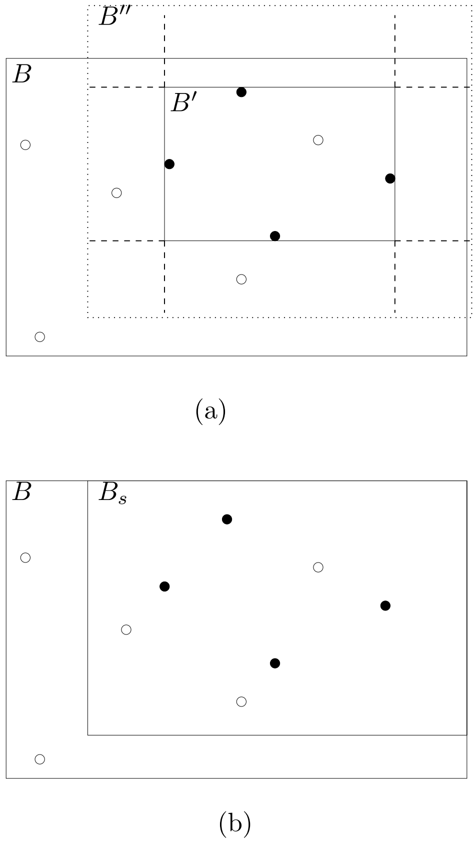


Fig. 1. *Illustration of the* Subrectangle *process. The circles are input points. The solid black ones indicate points with open facilities.* (a) *The intermediate rectangles created.* (b) *The input B and the output B_s of the process.*

to the middle third of the maximal side of the rectangle.

Output: The two created rectangles.

Remark 1. Given as input a rectangle with constant aspect ratio, both processes output rectangles with constant aspect ratio. This remark will be useful in section 4.

See Figure 1 for an illustration of the Subrectangle process. The recursive method alternately applies the Subrectangle and Cutrectangle processes to produce a decomposition of the original rectangle containing the input. The method stops when either the current rectangle contains one point or the current rectangle contains no facilities—whichever happens earlier. We emphasize that in this section we do not use an a priori random shift of the coordinate system as Arora does in [2]. The Subrectangle process would diminish any randomization introduced at the beginning of the dissection. The randomization required by the upcoming Facts 1 and 2 is introduced by Cutrectangle. We also observe that the original rectangle is not necessarily covered by leaf rectangles in the decomposition, due to the subrectangle steps.

We place $m+1$ evenly spaced points on each side of each rectangle in the dissection, where m will be defined later and depends on the accuracy ε of the sought approximation. We call these points *portals.* We define a *portal-respecting path* between two points to be a path between the two points that crosses only rectangles that enclose one of the points at portals. We define the *portal-respecting distance* between two points to be the length of the shortest portal-respecting path between the points. We begin by giving three technical lemmas which will be of use in the main Structure Theorem. Lemma 1 has a straightforward proof and gives the motivation behind the decomposition. We want short assignment edges in a given solution to be separated by rectangles of small sidelength.

LEMMA 1. *If the first rectangle R in the dissection to separate points v and w has sidelength D, the difference between the portal-respecting distance and the geometric distance between v and w is $O(D/m)$.*

We define a *cutting* line segment in the decomposition to be either (i) a line segment l that is used in the CUTRECTANGLE process to divide a rectangle R into two rectangles or (ii) a line segment l used to form the boundary of a proper subrectangle R in the SUBRECTANGLE procedure. In both cases we say that l *cuts* R. We define the *sidelength* of a cutting line l as the sidelength of the rectangle cut by l. Observe that the length of a cutting line is upperbounded by its sidelength.

LEMMA 2. *If any two parallel cutting line segments produced by the application of* CUTRECTANGLE *are within distance L, one of the line segments has sidelength at most $3L$.*

Proof. Let l_1, l_2 be the two cutting segments at distance L. Assume without loss of generality that they are both vertical, l_1 is the longer of the two lines, and l_1 is on the left of l_2 and cuts a rectangle R of sidelength greater than $3L$ into R_1 and R_2. Then l_1 is produced first in the decomposition. Thus l_2 is contained within R_2 (see Figure 2) and since it comes second can only cut a rectangle R'_2 contained within R_2. By the definition of CUTRECTANGLE, if s is the sidelength of R'_2, l_2 is drawn at least $s/3$ away from the left boundary of R'_2, which implies $s/3 \leq L$. Thus $s < 3L$. □

The next lemma relates the length of a cutting line segment produced by SUBRECTANGLE to the length of any assignment edges it intersects. We slightly abuse terminology and say that an edge (v, f) is *separated* when a cutting line separates v and f.

LEMMA 3. *Let f be a facility and (v, f) an assignment edge of length D. If a cutting line segment σ produced by* SUBRECTANGLE *separates v and f for the first time, then σ has sidelength at most $5D$.*

Proof. Let B_s be the rectangle cut by σ; let s be its sidelength. Assume without loss of generality that σ is a vertical line and that the horizontal dimension of B_s is maximal. Let B_s be produced by a SUBRECTANGLE process with input B. We use the notation of the process for the intermediate rectangles produced. Let $y \leq s$ be the sidelength of the rectangle B'. Then the sidelength of B'' is $(5/3)y > s$. The two vertical strips at the side of B'' are empty of facilities. Part of those strips may lie outside B; see Figure 3. The intersection of the vertical strip which is crossed by (v, f) with B must be a rectangle whose projection on the x-axis has length $y/3$. Therefore, $y/3 < D$, which implies that $s < 5D$. □

Consider the optimal solution with k given medians among the points. In the Structure Theorem we will prove the existence of a near-optimal solution in which a point v is always assigned to its closest or second-closest median. The *modified cost* of an assignment E is the sum over all the points of the portal-respecting distances to their respective assigned facilities. Since the Structure Theorem assumes that the set of open facilities is given, it applies more generally to uncapacitated facility location.

We first provide two calculations that will be of use in the proof of the theorem.

FACT 1. *Let $\mathcal{E}(\Delta, Z)$ denote the event that an edge of length Δ is separated by a*

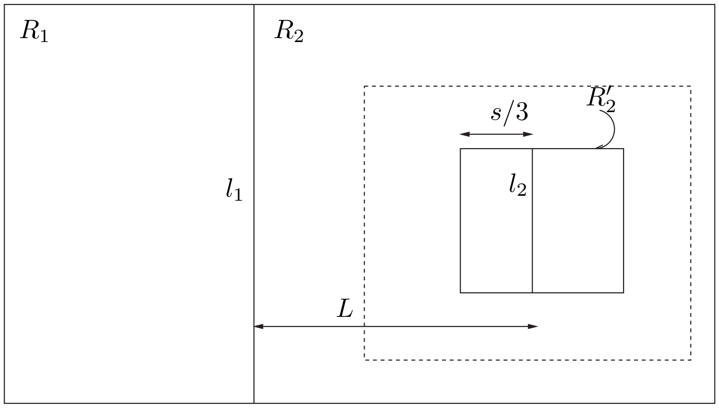


FIG. 2. *Illustration of the proof of Lemma* 2. *Rectangle R_2' is a descendant but not necessarily a child of R_2 in the decomposition.*

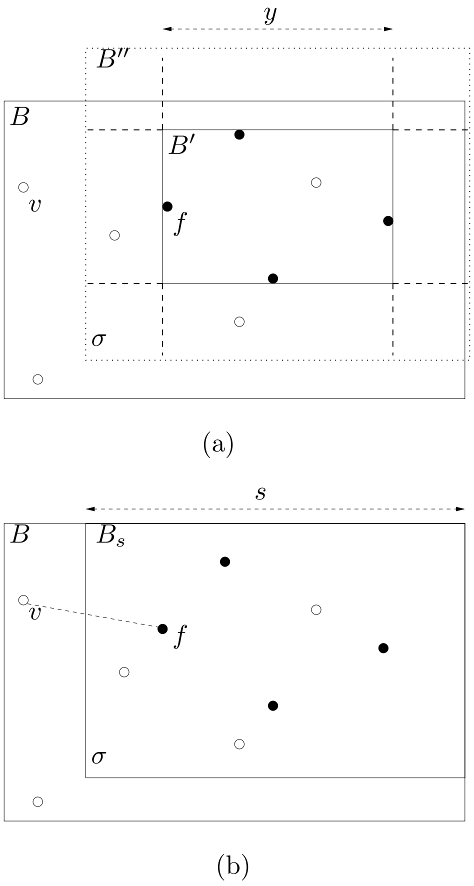


FIG. 3. *Illustration of the proof of Lemma* 3. *The circles are input points. The solid black ones indicate points with open facilities.* (a) *The intermediate rectangles created by* SUBRECTANGLE. (b) *The input B and the output B_s of the process.*

cutting line of sidelength Z that is produced by Cutrectangle. *We have that*

$$Pr[\mathcal{E}(\Delta, Z)] \leq 3\Delta/Z.$$

Intuitively, sidelengths increase geometrically in the dissection. Therefore, a consequence of Fact 1 is that the probability that an edge of length Δ is separated by a cutting line of sidelength Z or more that is produced by Cutrectangle is at most $O((\Delta/Z))$.

Fact 2. *Let $\mathcal{E}_{\geq}(\Delta, Z)$ denote the event that an edge of length Δ is separated by a cutting line of sidelength Z or more that is produced by* Cutrectangle. *Then*

$$Pr[\mathcal{E}_{\geq}(\Delta, Z)] = O(\Delta/Z).$$

The following lemma will also be of use.

Lemma 4. *Let the distance from a point v to its closest open facility f be D and the distance from v to its second closest open facility l be L. In the decomposition, v and f are separated for the first time by a cutting line that has sidelength either* (i) *larger than $L/2$ or* (ii) *smaller than $8D$.*

Proof. We know that v and l are separated by the time the sidelength of any enclosing rectangle is at most L. Observe that by Lemma 3, a cutting line of sidelength $> 5D$ separating v and f can only be produced by Cutrectangle. We will show that Cutrectangle cannot produce such a cutting line with sidelength in $(8D, L/2)$.

Without loss of generality we assume the relative positioning of v, f, l given above. See Figure 4 for an illustration. For any rectangle of sidelength $L/2$ containing v, there are no facilities to the left of v. Thus, by the Subrectangle process on input of sidelength $s < L/2$, the left boundary of any rectangle of sidelength $s < L/2$ that contains v is within a distance of at most $s/5$ of v. This is because in Subrectangle the maximal rectangle empty of facilities is grown by at most a third in each direction. By the Cutrectangle process on input of sidelength s, any cutting line for a rectangle box is at least $s/3$ to the right of its left boundary, which implies that the cutting line is at least $s/3 - s/5 = 2s/15$ to the right of v. Thus, the length D line segment cannot be cut until the sidelength of the enclosing rectangle is at most $(15/2)D$. □

Theorem 1 (Structure Theorem). *Let $m > 0$ be any integer and $F \subset N$ be a set of open facilities. There is an assignment E such that the expected difference between the modified cost of E and the greedy cost C under F is $O(C \max\{1, \log(m)\}/m)$.*

Proof. By linearity of expectation, it suffices to bound the expected cost increase for a given assignment edge. For a point v we define $f \in F$ as v's closest facility and assume f to be to the right of v (without loss of generality). We define $l \in F$ as the closest facility to the left of v. We denote the distance from v to f by D, and the distance from v to l by L. The idea behind the analysis of the portal-respecting solution is that assigning v to either f or l (a decision based on the amount by which the decomposition distorts each distance) will be enough to show near-optimal modified cost. The proof of the theorem shows how to actually construct the assignment E.

We assume without loss of generality that v and f are separated for the first time by a vertical cutting line. We can turn the configuration on its side and use the same argument if this condition does not hold.

The semicircle of diameter $2L$ centered at v and lying entirely to the left of the vertical line passing through v is empty in its interior. Therefore Lemma 4 applies.

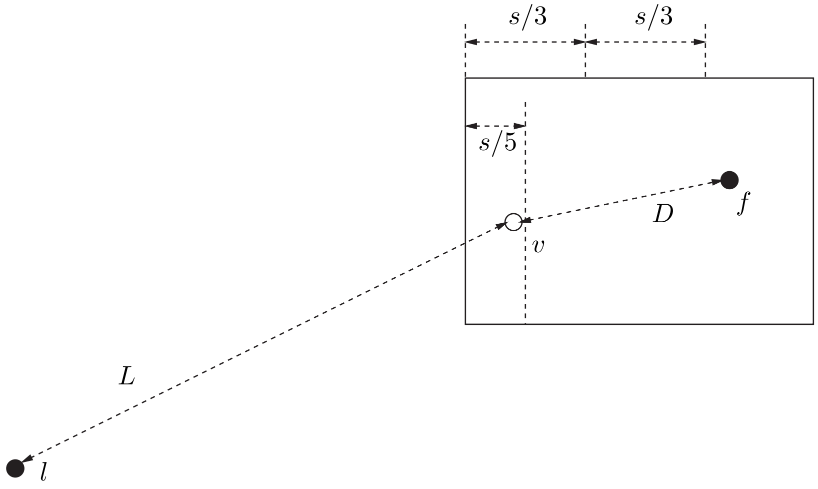


FIG. 4. *Illustration of the proof of Lemma* 4. *In this case we depict* $s < 2D$.

We proceed to a case analysis based on which of the two edges (v, l) or (v, f) is separated first by the decomposition. Observe that (v, l) can be separated for the first time by either a vertical or a horizontal line. If v and f are separated for the first time by a line produced by SUBRECTANGLE, we assign v to f in E and the increase in cost is $5D/m$ by Lemma 3. Therefore we can assume for the remainder of the proof that v and f are first separated by a vertical cutting line produced by CUTRECTANGLE. Let μ, ξ denote the lines separating for the first time (v, l) and (v, f), respectively.

CASE A. *Edge* (v, l) *is separated before* (v, f).

We will now calculate the expectation of the cost increase for the two possible subcases.

CASE A1. *Edge* (v, l) *is separated for the first time by a vertical cutting line.* We assign v to f in E. With some probability p, ξ has sidelength $L/2$ or more. By Lemma 4, ξ has sidelength at most $8D$ with probability $(1 - p)$. Therefore, by Lemma 1, the cost increase is $O(D/m)$ with probability $(1 - p)$. Now we turn to the case in which ξ has length $L/2$ or more. If μ is produced by SUBRECTANGLE, by Lemma 3, μ has sidelength at most $5L$. If μ is produced by CUTRECTANGLE, by Lemma 2 either ξ or μ has length at most $6L$. Moreover, ξ always has length smaller than μ since it is produced second in the dissection. By Lemma 1 the cost increase is $O(L/m)$ regardless of the operation producing μ. By Fact 2, probability p is $O(D/L)$. Therefore, the expected cost increase for CASE A1 is at most

$$(1 - p)O(D/m) + pO(L/m) = O(D/m) + O((D/L)(L/m)) = O(D/m).$$

Remark 2. The probability calculation for Case A1 depended only on the choice of which vertical line first cuts (v, f). This will be useful in section 4, when we restrict our choices for vertical cutting lines.

CASE A2. *Edge* (v, l) *is separated for the first time by a horizontal cutting line.* We assign v to f in E. We compute first the expectation of the cost increase conditioned upon the sidelength X of line μ. See Figure 5. By Fact 1, (v, f) is cut by a line of sidelength Y with probability at most $3D/Y$. Observe that this is true regardless of the value of X. Moreover, $L/2 \leq Y \leq X$ or, by Lemma 4, $Y \leq 8D$. The upper bound of X holds since (v, f) is contained in the rectangle cut by μ. For some constant c,

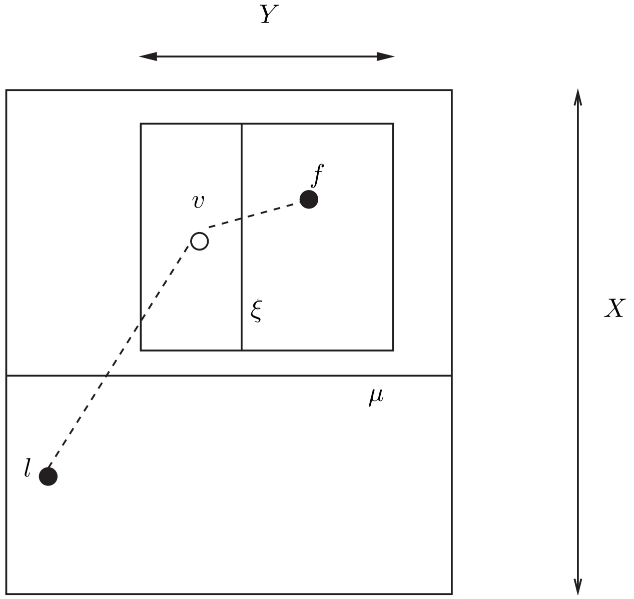


FIG. 5. *Case* A2 *in the proof of Theorem* 1.

the conditional expected cost increase is bounded by

$$\sum_{L/2\leq Y\leq X|\exists i, Y=2^i} (cD/Y)(Y/m) = O\left((D/m)\log(X/L)\right).$$

We now remove the conditioning on X. If line μ was produced by SUBRECTANGLE, by Lemma 3 it has length at most $5L$. No matter how large the probability of this event is, by the sum above the cost increase is $O(D/m)$. If line μ is produced by CUTRECTANGLE, by Fact 1 it has sidelength X with probability at most $3L/X$. The expectation of the cost increase is at most

$$\sum_{X\geq L|\exists i, X=2^i} 3(L/X)\log(X/L)O(D/m) = O(D/m).$$

Remark 3. To compute the probability of the event B that ξ has a given sidelength Y, we used only the total probability theorem. We partitioned the event space into the possible events $A_1, A_2, \ldots$ based on the different possible values of X and then applied $Pr[B] = \sum_i Pr[B|A_i]Pr[A_i]$. This remark will be useful in section 4.

CASE B. *Edge* (v, f) *is separated before* (v, l).

CASE B1. *Edge* (v, l) *is separated for the first time by a vertical cutting line.* The difference from CASE A1 is that we cannot argue that ξ has smaller sidelength than μ; therefore, we follow a different strategy. If the sidelength of ξ is at most mL we assign v to f; else we assign v to l in E.

By Lemma 4, if the cost increase exceeds $8D/m$ the sidelength of ξ is $L/2$ or higher. By Fact 1, there is a constant c such that assigning v to f yields an expected cost increase of

$$c[(2D/L)(L/2m) + (D/L)(L/m) + (D/2L)(2L/m) + \cdots + (D/mL)(mL/m)]$$
$$= O(D\log(m)/m).$$

By Fact 2, the probability that ξ has a sidelength of mL or more and hence that we assign v to l is $O(\frac{D}{mL})$. In this case Lemma 2 gives that the sidelength of μ is at most $3(L + D)$. Therefore, when v is assigned to l, the expected assignment cost (and not just the increase) is

$$O\left(\frac{D}{mL}(L + L/m)\right) = O(D/m + D/m^2).$$

Therefore, regardless of whether v is assigned to f or l the expected cost increase is $O(D \log(m)/m)$.

CASE B2. *Edge* (v, l) *is separated for the first time by a horizontal cutting line.* We follow the same strategy as in Case B1: if the sidelength of ξ is at most mL we assign v to f; else we assign v to l in E.

By the same argument as in Case B1, if v is assigned to f the expected cost increase is $O(D \log(m)/m)$. Consider the case where v is assigned to l. If μ is produced by SUBRECTANGLE by Lemma 3 it has sidelength at most $5L$ and hence the expected assignment cost is $O(D/m + D/m^2)$ by a calculation similar to Case B1.

If μ is produced by CUTRECTANGLE it cannot have sidelength less than L. Moreover, by Fact 1 it has sidelength X with probability at most $3L/X$. X cannot exceed the sidelength Y of ξ but for every possible X in the range $[L, Y]$, the conditional probability that μ has sidelength X is still at most $3L/X$. Therefore, we obtain that the expected assignment cost for v is

$$O\left(\sum_{mL \le Y | \exists i, Y = 2^i} \frac{D}{mL} \sum_{L \le X \le Y | \exists i, X = 2^i} (L/X)(X + X/m)\right)$$
$$= O\left(\frac{D}{mL}(L + L/m)\right) = O(D/m + D/m^2).$$

Remark 4. To compute the probability of the event B that μ has a given sidelength X, we used only the total probability theorem. We partitioned the event space into the possible events $A_1, A_2, \ldots$ based on the different possible values of Y and then applied $Pr[B] = \sum_i Pr[B|A_i]Pr[A_i]$. This remark will be useful in section 4.

This is the end of the proof of the Structure Theorem. □

4. Modifying the Structure Theorem.

The Structure Theorem in the previous section demonstrates that a portal-respecting $(1 + O(\log(m)/m))$-approximate solution exists while only placing m portals on the boundary of the decomposition rectangles. The randomization introduced by CUTRECTANGLE is essential for the Structure Theorem. However, it does pose the problem of how to enumerate all possible outputs of CUTRECTANGLE in the context of a dynamic program such as the one in [3]. In this section we show how to effectively bound the number of rectangles to be enumerated by the dynamic program and in the process obtain a nearly linear-time algorithm. To this end we introduce some discretization on the possible outcomes of SUBRECTANGLE and CUTRECTANGLE.

We first give some definitions. Consider the square of sidelength T that surrounds the original input. We know from section 2 that $T = O(n^c)$ for some constant $c > 0$. We assume that $T = m2^\rho$ for some integer ρ and that the leftmost lower corner of the square lies at the origin of the axes. Call the vertical (horizontal) lines whose x-coordinate (y-coordinate) is an integral multiple of m *eligible.* Then, we call the vertical (horizontal) eligible lines with x-coordinate (y-coordinate) congruent to 0 mod2^i, $1 \leq i \leq \rho$, *i-allowable.* Note that the top- and leftmost eligible lines are ρ-allowable, and that any j-allowable line is i-allowable for all $i < j$. A rectangle R of sidelength s is *t-allowable* if t is the maximum value such that any two parallel sides of R of length s lie on t-allowable lines and $s \geq 2^t m$. Observe that when a rectangle is t-allowable, the aspect ratio is not bounded. The definition guarantees only that the smallest side has length $2^t m$. A rectangle which is t-allowable for some t and whose sides have length within a factor of $5 + 1/2m$ of each other is called *allowable.*

We modify the SUBRECTANGLE and CUTRECTANGLE processes as follows.

SUBRECTANGLE-NEW:
Input: An allowable rectangle containing at least one facility.
Process: Perform the SUBRECTANGLE process of the previous section. Let B_s be the computed rectangle.
Output: The minimal allowable rectangle that contains B_s.

CUTRECTANGLE-NEW:
Input: An allowable rectangle containing at least one facility.
Process: Choose a cutting line in the middle third of the maximal side of the rectangle, uniformly at random among all lines that produce two allowable subrectangles.
Output: The two allowable subrectangles.

To illuminate further CUTRECTANGLE-NEW consider an example that takes as input a t-allowable rectangle with sidelength $m2^t$. This implies that the rectangle is a square. Let us consider the horizontal side as maximal. The middle third of the maximal side has $\lfloor m/3 \rfloor$ candidate t-allowable cutting lines. Choosing one yields two subrectangles for which the horizontal side has length at least $(m/3)2^t$ and at most $(2/3)m2^t$. Accordingly the two subrectangles are each i-allowable for $i \geq t-2$.

Before proving the Modified Structure Theorem we examine the validity of the deterministic lemmas from section 3 in the new setting. Clearly Lemma 2 continues to hold. We have to be more careful with Lemmas 3 and 4 due to the extra requirement that the output of SUBRECTANGLE-NEW should be allowable. This might cause the rectangle output by SUBRECTANGLE to be "stretched." We first show that this stretch is small.

LEMMA 5. *Each side of the rectangle output by* SUBRECTANGLE-NEW *has length at most $1 + 2/m$ the length of the corresponding side of the rectangle B_s computed during the process.*

Proof. Within the SUBRECTANGLE process, the rectangle B' of sidelength s is grown by $s/3$ in each dimension. It follows that the lengths of the sides of B_s are within a factor of $(s + 2s/3)/(2s/3) = 5$ of each other. To meet the definition of an allowable rectangle, we have only to move the boundaries of B_s so that they fall on t-allowable lines. Here t is the maximum integer so that each side of B_s has length at least $2^t m$. To achieve this, we have to extend each side by at most 2^t in each direction for a total increase of a $1 + 2/m$ factor. □

Lemma 4 continues to hold with the modified decomposition. In fact it becomes slightly stronger since (cf. proof of Lemma 4) the left boundary of any allowable rectangle produced by SUBRECTANGLE-NEW that has sidelength $s < L/2$ and contains v is within a distance of at most $s(1/5 + 1/m)$ from v. We now give the equivalent to Lemma 3 in the modified setting.

LEMMA 6. *Let f be a facility and (v, f) be an assignment edge of length D. If a cutting line segment σ produced by* SUBRECTANGLE-NEW *separates v and f for the* *first time, then σ has sidelength at most* $(5 + 6/m)D$.

Proof. We outline only the changes from the proof of Lemma 3. Keeping the same notation, by Lemma 5 $(5/3)y \leq s \leq (5/3 + 2/m)y$. Since $y/3 < D$, we obtain $s < (5 + 6/m)D$. □

The difference between Lemmas 3 and 6 is negligible from the point of view of the Modified Structure Theorem since it introduces only an additive $O(D/m^2)$ error term. We can now focus on the probabilistic part of the modified decomposition.

The new dissection is similar to the one from section 3. The primary difference is that the randomization in the CUTRECTANGLE process has been diminished. If we add back some randomization up front by shifting the original rectangle containing the input, we can get the same result as in the Structure Theorem on the expected increased cost of a portal-respecting solution. We define an (a, b)*-shifted* coordinate system, $0 \leq a, b \leq T$, as one in which the x and y coordinates are shifted by a and b, respectively. The shifting uses wraparound as in [2]: a vertical line which had x-coordinate x_1 before the shifting has coordinate $x_1 + a \bmod T$ after. If a and b are chosen at random, any vertical or horizontal line is equally likely to be i-allowable. The *new modified cost* of an assignment is defined with respect to the new dissection given above.

THEOREM 2 (Modified Structure Theorem). *Let $m > 0$ be any integer and $F \subset N$ be a set of open facilities. If the coordinate system is randomly shifted by (a, b), where a and b are chosen independently in $[0, T]$, then there is an assignment E such that the expected difference between the new modified cost of E and the greedy cost C under F is $O(C \max\{1, \log(m)\}/m)$.*

Proof. As mentioned, minor variations of the deterministic lemmas from section 3 continue to hold in the restricted version of the dissection. The randomized portion in the proof of Theorem 1 reasons only about two types of events. Moreover, it reasons about each event in isolation (cf. Cases A1 and A2 in the proof; the rest are similar). Thus, we need only be concerned with the probability of each event. The random shift up front of the coordinate system along with the randomization inside the process will ensure that these two types of events occur in our decomposition into allowable rectangles with approximately the same probability as in the previous decomposition.

The first event (cf. Case A1) is that a line produced by CUTRECTANGLE-NEW of length X cuts a line segment of length D. The probability of this event is required to be at most $3D/X$ in the proof of the Structure Theorem. We show that this continues to hold albeit with a constant larger than 3.

We assume for simplicity that the line segment is in a rectangle R of sidelength exactly $X = m2^i$ at some point. (This will be true to within a constant factor.) If $D < 2^i$, we know that the segment is cut by at most one i-allowable line. The probability of this event is at most $D/2^i$ due to the random shift. The CUTRECTANGLE-NEW process chooses from $m/3$ i-allowable lines uniformly at random. Thus, the line segment is cut with probability $1/(m/3)$ times $D/2^i$, which is $3D/X$ as required. If $D > 2^i$, we notice that D intersects at most $\lceil D/2^i \rceil < 2D/2^i$ i-allowable lines. By the union bound the probability that this line segment is cut during CUTRECTANGLE-NEW on R is upperbounded by $2D/2^i$ times $3/m$, i.e., $6D/X$.

The second event (cf. Case A2) is the intersection of two events; a horizontal line of length X cuts a segment of length L and a vertical line of length Y cuts a segment of length D. In the proof of the Structure Theorem, the probability was shown by the total probability theorem to be upperbounded by the product of the probability bounds of the two events, i.e., $3L/X$ times $3D/Y$. The second term represented the conditional probability that edge (v, f) is cut by a line of sidelength Y given that (v, l) was cut by a horizontal line of sidelength X. We argued that the conditional probability does not depend on X.

For our restricted decomposition, the probability of each event in isolation can be bounded by $6L/X$ and $6D/Y$ as argued above. Moreover, we chose the horizontal and vertical shifts independently and we choose the horizontal and vertical cut-lines in different processes. Thus, we can also argue that the probability of the intersection of the two events is at most $6L/X$ times $6D/Y$. □

We now prove a lemma bounding the number of allowable rectangles.

Lemma 7. *The number of allowable rectangles that contain l or more points from the input set N is $O(m^4(n/l)\log n)$.*

Proof. Our proof uses a charging argument. Let R_l be a rectangle on the plane that has minimum sidelength, say, L, and contains l points. We bound the cardinality of the set S_l of allowable rectangles, which are distinct from R_l, contain at least l points and have at least one point in common with R_l. Let R_a be such a rectangle. Then R_a has sidelength at least L; otherwise it would have been chosen instead of R_l.

We bound the number of allowable rectangles in S_l with sidelength $X \in [2^{i-1}, 2^i]m$ by $O(m^4)$ as follows. The corners must fall on the intersection of two t-allowable lines that are within distance X from some side of R_l. Since allowable rectangles have bounded aspect ratio, the possible values for t are 2^{i-k}, $k = 0, 1, 2, 3$.

The number of j-allowable lines that are within distance X from some side of R_l is $O(X/2^{j-1})$ since $X \geq L$. Thus, the number of corner choices is $O(m^2)$. Two corners must be chosen, so the number of rectangles in S_l of sidelength $X \in [2^{i-1}, 2^i]m$ is $O(m^4)$. Since there are $O(\log n)$ values of i, $|S_l| = O(m^4 \log n)$.

Now, we remove R_l and its points from the decomposition and repeat the argument on the remaining $n - l$ points. The number of repetitions until no points are left is $O(n/l)$; therefore, by induction, we get a bound of $O(m^4(n/l)\log n)$ on the number of allowable rectangles that contain at least l points. □

5. The dynamic program. We have structural theorems relative to a particular decomposition. Unfortunately, the decompositions are defined with respect to the facility locations. However, in reality they use only the facility locations in the Subrectangle steps. Moreover, the number of subrectangles is at most polynomial in the size of the original rectangle. Indeed, the number of allowable subrectangles is polynomial in m and n. Thus, we can perform dynamic programming to find the optimal solution. The structure of the lookup table is similar to the one used in [3]. We exploit our Structure Theorem and the analysis on the total number of allowable rectangles to obtain a smaller number of entries.

We will develop two versions of our dynamic program, in order of increasing sophistication, each with a somewhat different time complexity. Our dynamic programs have much in common with the one in [3]. For the sake of completeness we include here all the relevant definitions from [3].

5.1. The table definition. In all our approaches the table will consist of a set of entries for each allowable rectangle that contains at least one point. There is a one-to-one correspondence between entries of the table and subproblems on the corresponding rectangle. Subproblems on a given rectangle are generated by what amounts to a guessing (enumeration) of the location of facilities that serve the points in the rectangle. These facilities could be either inside the rectangle or outside. In either case, since by the Structure Theorem it suffices to consider portal-respecting solutions, we store only the *interaction pattern* (defined by items 2 and 3 below) of the portals to the facilities. For each allowable rectangle, we will enumerate the following:

1. the number of facilities in the rectangle,
2. a distance for each portal p to the nearest facility in the rectangle, and
3. a distance for each portal p to the nearest facility outside the rectangle, if one exists that is nearer to p than all facilities inside the rectangle.

Every triple of this sort defines a subproblem. Given a subproblem which is indexed by f for the number of facilities and is defined on a rectangle R, a solution will be a set of f facilities among the points in R and a portal-respecting path from each point in R to one of the facilities in R or to a portal on the boundary. The table entry corresponding to the subproblem will store the minimum service cost under the given constraints. Recall from section 3 that the decomposition stops as soon as a rectangle is produced that contains no facilities. Formally, a *leaf* of the decomposition is a rectangle R which has been produced by the decomposition, such that either (i) R contains one point and this point is an open facility or (ii) R is a rectangle that contains at least one point and no facilities.

The enumeration of the distances happens by using the *inside* and *closest* functions on the portals as defined in [3]. In order to save on the size of the table the definitions of the two functions follow three rules, as in [3]. We present them first, establish their validity, and then provide the definitions that implement them.

Rule 1. We will only approximate the distances to the nearest facility inside and outside of a rectangle R to a precision of s/m for a rectangle of sidelength s.

Rule 2. As far as the distance from a portal p to the closest facility is concerned we do not consider distances of more than $6s$. Clearly the distance from a portal p on the boundary of R to the nearest facility within R cannot exceed $2s$. Therefore this rule affects only assignments to the outside of R, i.e., the upcoming definition of the *closest* function. We want to ensure that no assignments are lost for the dynamic program. A key remark is that a portal p is assigned to a facility η outside R *only if all* facilities within R (if such exist) are further from p than η. Therefore if R contains some facility, any facility outside R at a distance of more than $2s$ is of no interest to the points in R. If R contains no facilities, it is a leaf of the decomposition. This leaf rectangle must have been produced by CUTRECTANGLE-NEW. Let R' be the parent rectangle that was cut. $R' - R$ must contain some facility, or else R' itself must have been a leaf. The sidelength s of R is at least one third the sidelength of R'. Hence by a generous estimate there is a facility at distance at most $6s$ from every portal of R.

Rule 3. It suffices for neighboring portals to represent distances as offsets from the previous distance. This is o.k. because the distance value at a portal changes only by a constant from the distance value at an adjacent portal.

We are now ready to give the definitions of the *inside* and *closest* functions as in [3]. Let Π be a subproblem defined on rectangle R, and let Π be indexed by a number of facilities $f \in [0, k]$. For a portal p, *inside* (*closest*) encodes the distance to the nearest facility to p within (outside) R, subject to Rules 1, 2, and 3.

- If $f = 0$, we set $inside(p) = \infty$ for every portal p on the boundary of R. This should happen for all R that are leaves of the decomposition.
- If $f > 0$, we define an assignment *inside* of numbers $[1, \ldots, 4m]$ to the portals of R, such that $|inside(p) - inside(p')| \leq 1$ if portals p and p' are successive along the boundary of R. The domain of the assignment expresses Rule 1, i.e., the precision of the distances. The condition on the difference expresses Rule 3; i.e., we represent distances as offsets from the previous distance.

- If $f < k$, we define an assignment *closest* of numbers $[1, \ldots, 4m]$ to the portals of R, such that (i) $|closest(p) - closest(p')| \leq 1$ if portals p and p' are successive along the boundary of R and (ii) $closest(p) < inside(p)$. Since by Rule 2 above, we can always assume that there is a facility within distance $6s$ from p, the definition can fail only if one such facility within R is the nearest to p, i.e., when condition (ii) fails. In this case we set $closest(p) = \infty$.
- If $f = k$, we set $closest(p) = \infty$ for every portal p on the boundary of R.

By the three rules above, as far as the distances are concerned we need only to guess $O(m)$ bits per entry. Taking into account the guess for the number of facilities the total number of table entries (subproblems) corresponding to each allowable rectangle is bounded by $k2^{O(m)}$. We bound the table size by noting that the total number of allowable rectangles is, by Lemma 7, at most $O(m^4 n \log n)$. Thus, we can bound the total number of entries in the table by $k2^{O(m)} n \log n$.

5.2. Computing the table entries. With respect to the recursive decomposition defined by the Modified Structure Theorem every subproblem defined on a rectangle R has one or two *children* subproblems, the number depending on which process, Cutrectangle-new or Subrectangle-new, is applied. The solution for a subproblem Π defined on rectangle R is computed by looking at children subproblems whose interaction patterns match with the interaction pattern of Π. The *recursion height* of any subproblem defined on R is equal to the maximum length of a path leading to a leaf of the decomposition in the tree representing the dissection defined by the Modified Structure Theorem. The maximum recursion height is obviously $O(\log n)$. Based on which process is applied, we are looking for either (i) a pair of subproblems defined on two allowable subrectangles that cover R and whose total number of facilities equals that of Π or (ii) a subproblem defined on a single allowable subrectangle of R whose number of facilities equals that of R. Among the matching combinations we select the one of minimum cost, where the cost is computed as follows. After fixing the values of the *inside* and *closest* functions on the portals of Π, for every point in R, we know the location of the nearest facility, either within R or outside. For a portal p on the boundary of R let $n(p)$ be the total number of points assigned to p, i.e., the total number of points that are served by a facility on the outside via p. Similarly to [3], the *cost* of Π equals the total assignment cost within R plus

$$\sum_{\text{portals } p} closest(p) n(p) (s/m).$$

An important difference of our scheme from [3] is that we cannot a priori compute the randomly shifted dissection and then apply dynamic programming on the rectangles of the dissection. Because of the Subrectangle-new process, for any given subproblem we have to enumerate all possible outputs of either the Subrectangle-new process or all different cuttings of the Cutrectangle-new process. This gives rise to two different types of algorithmic steps, one for each case. Fortunately, by Lemma 7 the total number of rectangles to be enumerated throughout the dynamic program is $O(m^4 n \log n)$.

Let R be a rectangle of sidelength s at height i of the recursion, and assume inductively that the algorithm has solved all subproblems at height $i-1$. Consider a subproblem Π defined on R and let Π be indexed by a number f of facilities. We now give the details of the algorithm's steps.

Application of SUBRECTANGLE-NEW. To capture the outcome of SUBRECTANGLE-NEW the algorithm enumerates all possible subproblems on allowable subrectangles R_u of R such that the following hold.

S1. The number of facilities open in R and R_u is equal to f.
S2. For each portal p of R there is a portal p' of R_u such that $d(p,p')+inside(p')(s_u/m) \leq inside(p)(s/m)$. Here s_u denotes the sidelength of R_u, with $s_u \leq s$.
S3. For each portal p of R_u there is a portal p' of R such that $d(p,p')+(s/m)closest(p') \leq closest(p)(s_u/m)$.

There does not seem to be a way to look up the cost of R, say, as a function of the cost of the chosen R_u. Moreover, the number of possible candidates for R_u could be large; we need to consider candidates of arbitrarily small sidelength. We take the inverse view and enumerate the work done for each allowable rectangle S when it is considered as a subrectangle of some R. The argument that follows is due to J. Remy.

For fixed t, any rectangle S is contained in $O(1)$ t-allowable rectangles of constant aspect ratio. Therefore, a fixed S is considered as a subrectangle $O(\log n)$ times. Therefore, by Lemma 7 the total number of subrectangles, not necessarily distinct, that the dynamic program needs to enumerate is $O(m^4 n \log^2 n)$. For every such occurrence of S in the dynamic program we enumerate all the interaction patterns and the possible number of facilities, which yields $k2^{O(m)}$ subproblems. We proceed to bound the running time for each subproblem involving a rectangle R containing S.

A subproblem defined on S specifies the positions of facilities within R (up to an additive $sidelength(S)/m$ factor). We want to assign the points in R to the closest facility within S or outside R without paying time proportional to the number of points in R. The intuition behind the proof is that a higher additive error on the assignment can be tolerated for points in $R \setminus S$ that are “far” from S. The adaptive dissection suggests at a high level guessing the rectangle S and accordingly the location of facilities within R. To assign the points efficiently, the algorithm will be also adaptive at a much lower level and tune the accuracy of the assignment for the different points.

LEMMA 8 (due to J. Remy). *Let Π_R and Π_S be two subproblems defined on rectangles R and S where S is contained in R. The two subproblems agree as in* S1–S3 *above. Let G be the assignment of every point in R to its closest facility where the locations of facilities are specified by Π_R and Π_S. We can compute an assignment G' such that the service cost of each point p has an additive error of $O(1/m)$ times the modified service cost of p in G. G' and its total cost can be computed in $O(m^4 \log^2 n)$ time worst-case.*

Proof. The construction uses some ideas similar to the ones in [26]. We partition the area of $R \setminus S$ into rectangular *moats* $M_1, M_2, \ldots$. Moat M_1 is a $2d$-shape created by surrounding S by a rectangle S' that strictly contains S and then removes S. Moat M_{i+1} is the area remaining from a rectangle surrounding moat M_i after $(\bigcup_{j \leq i} M_j) \cup S$ has been removed. The *radius* r_i of moat M_i is the maximum distance of an outer vertical (or horizontal) side from the nearest vertical (horizontal) side of S. The radii are set to be geometrically increasing with $r_i = sidelength(S)(1+1/m)^i$, $i = 1, 2, \ldots$. Any outer vertical (horizontal) side of M_i is at the maximum distance that does not exceed r_i from the closest vertical (horizontal) side of S. Hence a side at a distance less than r_i will fall on the boundary of R. See Figure 6 for an illustration. If the area left uncovered in R at some point of the process is not wide enough to become a moat by itself, we append it to the last moat produced and terminate. The number of moats is $O\left(m \log(sidelength(R) - sidelength(S))\right) = O(m \log n)$.

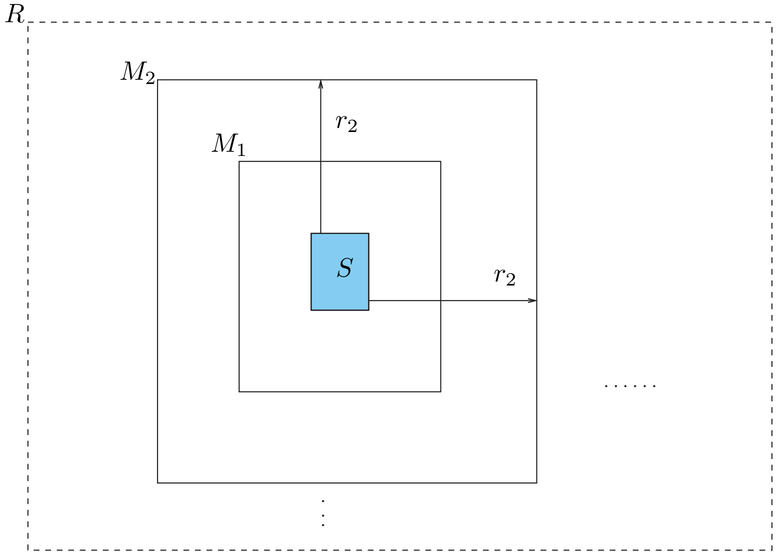


Fig. 6. *Illustration of the moat construction process. If all sides of S are equidistant from the sides of R, each of the moats will be topologically equivalent to an annulus.*

Each moat M_i is divided into $O(m^2)$ rectangular cells whose sides are each $\Theta(r_i/m)$ long. See Figure 7. The total number of cells across all moats is $O(m^3 \log n)$. For every point in $R \setminus S$ we have to try $O(m)$ portals on the boundaries of S and R to find the closest facility. The algorithm identifies *all points within a cell* C with the center of the cell and assigns them to the same portal. We calculate now the error induced by this collapsing of points. For a point p in cell C of moat M_l, let d_1 be its distance from the nearest portal on the boundary of S. The estimated distance d_1' for all points in the cell to the closest portal on the boundary of S will be at most

$$d_1 + O(sidelength(C)) \leq d_1 + O\left(\frac{\max\{(1+1/m)d_1, (1+1/m)sidelength(S)\}}{m}\right)$$

$$= d_1 + O(1/m)\max\{d_1, sidelength(S)\}.$$

The second term in the max expression above is needed for points in M_1. For the same point p let d_2 be its distance from the nearest portal on the boundary of R. In the worst case the distortion is maximized when M_l is the outer moat, i.e., the one touching the boundary of R. The estimated distance after collapsing p will be at most $d_2 + O(sidelength(C)) \leq d_2 + O(sidelength(R)/m)$ which is fine since the modified service cost through a portal on the boundary of R is already distorted by $O(sidelength(R)/m)$.

The total cost of assigning the points in $R \backslash S$ is the weighted sum of the assignment costs of the cells where the cost for each cell is weighted by the number of points in the cell. Computing this number can be done by orthogonal range searching techniques in $O(\log n)$ time per cell (see [1]) after an $O(n \log n)$ global preprocessing. Therefore, the total running time is $O(m^4 \log^2 n)$. □

Remark 5. The analogue of Lemma 8 holds in d-dimensional space. The number of cells is $O(M^{d+1} \log n)$, where M is the number of portals maintained on each face of a parallelepiped. The range searching query time increases by a multiplicative $O(\log^{d-2} n)$ factor [1]. The running time becomes $O(M^{d+2} \log^d n)$.

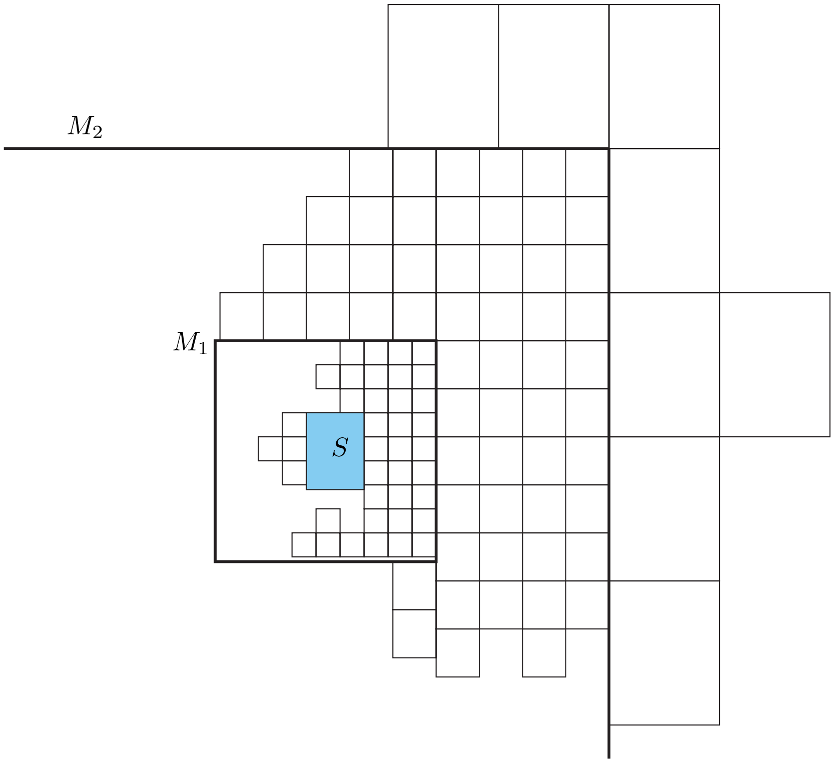


FIG. 7. *Illustration of the cell construction process.*

By Lemma 8 and the previous discussion the total computation time of the dynamic program due to SUBRECTANGLE-NEW is

$$\begin{aligned} &\text{Total time due to SUBRECTANGLE-NEW} \\ &\quad = (\text{Time per entry}) \ \times \ (\text{number of entries}) \\ &\quad = O(m^4 \log^2 n) \times k2^{O(m)} n \log^2 n = O(k2^{O(m)} n \log^4 n). \end{aligned} \tag{2}$$

This bound can be improved by amortization. By Lemma 7, the number of allowable rectangles that contain k or more points is $O(m^4 (n/k) \log n)$. If an allowable rectangle S contains fewer than $l < k$ points from N, we need only to keep $l2^{O(m)}$ table entries for it, since at most l facilities can be placed inside it. Moreover, the number of allowable rectangles containing between l and $2l$ points is shown by Lemma 7 to be $O(m^4 (n/l) \log n)$. We can now bound the total time due to SUBRECTANGLE-NEW by

$$\begin{aligned} &O(m^4 \log^2 n) \times \left(k2^{O(m)} (n/k) \log^2 n + \sum_{l=2^i}^{l<k} l2^{O(m)} (n/l) \log^2 n \right) \\ &\quad = 2^{O(m)} n \log^4 n \log k. \end{aligned} \tag{3}$$

Application of CUTRECTANGLE-NEW. The algorithm enumerates also all possible cuttings of R according to the CUTRECTANGLE-NEW process. It looks for combinations of subrectangles R_1, R_2 of R whose interaction patterns match those of Π. The details are similar to the method outlined in [3], where four children rectangles are considered instead of two. In particular, the algorithm enumerates pairs of subproblems defined on allowable rectangles R_1, R_2 that partition R, fulfill the sidelength requirements of CUTRECTANGLE-NEW, and in addition fulfill the following.

C1. The sum of the number of facilities in R_1 and R_2 equals f.

C2. For each portal p of R there is a portal p' on one of R_1, R_2 (call it R') such that $d(p,p') + \mathit{inside}(p')(s'/m) \leq \mathit{inside}(p)(s/m)$. Here s' is the sidelength of R', with $s' \geq s/3$.

C3. For each portal p of R_j, $j = 1, 2$, there is either (i) a portal p' on one of R_1, R_2 (call it R') such that $d(p,p') + \mathit{inside}(p')(s'/m) \leq \mathit{closest}(p)(s(j)/m)$ or (ii) a portal p' of R such that $d(p,p') + \mathit{closest}(p')(s/m) \leq \mathit{closest}(p)(s(j)/m)$. Here $s(j)$ is the sidelength of R_j.

The total number of combinations of subrectangles and interaction patterns to consider for Π, due to CUTRECTANGLE-NEW, is $2^{O(m)}$. We also need to search over all possible splits of the facilities across pairs of subrectangles which gives an additional k factor. Therefore, the total number of subproblems needed to determine the value for the entry of subproblem Π is $k2^{O(m)}$. In contrast to the SUBRECTANGLE-NEW case, computing the cost of every combination can be done via lookup. For every pair of children subproblems that fulfill C1–C3, the cost of Π is equal to the sum of the two costs. We have that

$$\begin{aligned} &\text{Total time due to CUTRECTANGLE-NEW} \\ &\quad = \text{(Time per entry)} \times \text{(number of entries)} \\ &\quad = k2^{O(m)} \times k2^{O(m)} n \log n = O(k^2 2^{O(m)} n \log n). \end{aligned} \tag{4}$$

Again we can improve the total time to $O(k2^{O(m)} n \log k \log n)$ by amortization. By Lemma 7, the number of allowable rectangles that contain k or more points is $O(m^4 (n/k) \log n)$. If an allowable rectangle contains fewer than $l < k$ points from N, we need only to keep $l2^{O(m)}$ entries for it, since at most l facilities can be placed inside it. Moreover, the number of allowable rectangles containing between l and $2l$ points is shown by Lemma 7 to be $O(m^4(n/l) \log n)$. We can now bound the total number of table entries by

$$k2^{O(m)} O(m^4 (n/k) \log n) + \sum_{l=2^i}^{l<k} 2^{O(m)} O(m^4 (n/l) \log n) l = O(2^{O(m)} n \log k \log n). \tag{5}$$

By (4), (5) the total running time due to CUTRECTANGLE-NEW is

$$O(k2^{O(m)} n \log^2 n). \tag{6}$$

By (3) and (6) the total running time of the algorithm is $O(\max\{k, \log^3 n\} 2^{O(m)} n \log^2 n)$ in the worst case.

5.3. Improving the dependence on *k*. In a second approach, we can further replace the factor of k with a factor that depends only on m and $\log n$. The idea is to use indexing according to approximate cost. In this approach, the table entries are indexed by

1. the total cost of the solution corresponding to the service cost required by the clients (points) of the rectangle (this cost includes the possible assignments of points to facilities outside the rectangle via the appropriate portals),

2. a distance for each portal p to the nearest facility in the rectangle, and
3. a distance for each portal p to the nearest facility outside the rectangle if one exists that is nearer to p than all facilities inside the rectangle.

The value of a table entry is the minimum number of facilities required inside the rectangle to find a solution with the corresponding cost and interaction pattern. Naturally, we disregard subproblems indexed by cost values that are too small to be attainable with no more than k facilities.

How many different cost indexes are there? Given our initial assumption about the points lying on a grid of polynomial size, the total number of interesting cost values is polynomial in n. We quantize them further by considering only cost values that are powers of $(1+\delta)$. Then the number of distinct cost values becomes $\Theta(\log_{1+\delta} n)$. The effect of this quantizing is that for a single subproblem we approximate only its cost value to within a factor of $(1+\delta)$. These approximations pile up multiplicatively over the $O(\log n)$ levels of the decomposition. To offset this effect we choose δ to be $\Theta(m^{-1}/\log n)$. The total number of cost indexes is then $\Theta(m \log^2 n)$. Accordingly, the total number of table entries is $(\log^2 n)2^{O(m)} n \log n = 2^{O(m)} n \log^3 n$.

Under the cost quantizing, the intended meaning of the table values is the following. Let an entry be indexed by cost $C = (1+\delta)^y$ and some interaction pattern. Let this entry correspond to height i of the recursion. The value f in the entry means that the minimum number of facilities to achieve cost at most $C/(1+\delta)^i$ is at least $f+1$. In Lemma 9 below we will show that this intention can indeed be achieved. Computing an entry involves searching over combinations of subrectangles, matching interaction patterns, and cost splits, and choosing the combination that requires the minimum number of facilities in order to be realized. As before, the number of matching interaction patterns and subrectangles to be considered is $2^{O(m)}$ per table entry. The dynamic program computes entries defined on the same rectangle and with the same interaction pattern by increasing order of cost index.

For the number of cost splits we reason as follows. Given a subproblem on a rectangle R indexed by interaction pattern P and cost C one needs to consider all three cases that follow. *Cost-Split Case* (i): All subproblems on R indexed by the same interaction pattern P and with costs less than C. *Cost-Split Case* (ii): Due to the SUBRECTANGLE-NEW process, all subproblems defined on subrectangles of R with matching interaction pattern and costs less than or equal to C. Note that we might also consider cost equal to C if the subrectangle might contain all the points of R. *Cost-Split Case* (iii): Due to the CUTRECTANGLE-NEW, all cost splits C_1, C_2, across a pair of rectangles with appropriate interaction patterns, such that $C_1 + C_2 = C$. The minimum number of facilities between (i), (ii), and (iii) will give the value for the entry. Because everything is quantized at powers of $(1+\delta)$, sums get distorted. Therefore, for (iii), instead of considering all cost splits such that $C_1 + C_2 = C$, we consider C_1, C_2 such that

$$C/(1+\delta) < C_1 + C_2 \leq C.$$

The number of such splits is proportional to the total number of distinct cost values, i.e., $O(m \log^2 n)$.

To obtain the new running times we make the necessary changes to (2) and (4). In (4) the k factor is replaced throughout with $O(m \log^2 n)$, yielding time $O(2^{O(m)} n \log^5 n)$. In (2) the corresponding number of entries is proportional to the square of the total number of cost values. Therefore the total time due to SUBRECTANGLE-NEW is $O(2^{O(m)} n \log^8 n)$. Taking the maximum over the SUBRECTANGLE-NEW and the CUTRECTANGLE-NEW processes yields a worst-case running time of $O(2^{O(m)} n \log^8 n)$.

Now, we should argue that at each level the table entry corresponds to the minimum number of facilities that achieve the specified connection cost or better. This follows easily from the following lemma.

LEMMA 9. *If the table entry for a subproblem* Π, *which is indexed by cost* C, *is filled with the number* f, *then there exists a set of* f *facilities within the corresponding rectangle and a portal-respecting assignment of points to facilities that satisfies the interaction patterns such that the total cost is no more than* C. *Furthermore, there is no portal-respecting solution for* Π *that* (i) *uses fewer facilities and* (ii) *has actual service cost less than or equal to* $C/(1+\delta)^i$, *where* i *is the recursion height of the subproblem* Π.

Proof. We use induction on i. It is easy to compute table entries for rectangles that are leaves of the decomposition. Then, we inductively assume the lemma and note that the first part holds by construction. For the second part we consider the way the dynamic program computed the entry for Π. Let R be the corresponding rectangle. For the sake of contradiction assume that there is a better solution for Π, i.e., one with $g < f$ facilities and cost $C^* \le C/(1+\delta)^i$. Among these solutions, choose the one that minimizes g. By our (Modified) Structure Theorem it must be decomposable either via SUBRECTANGLE-NEW into a subproblem Π_u or via CUTRECTANGLE-NEW into two subproblems Π_1, Π_2.

In the first case, let R_u be the subrectangle of R on which Π_u is defined, and let $C_u \le C$ be the minimum cost for which Π_u admits a solution with g facilities. The cost C^* equals C_u plus the assignment cost of the points in $R - R_u$. By induction there is a computed entry for Π_u filled with the value g and indexed by cost $C'_u \le (1+\delta)^{i-1}C_u \le (1+\delta)^{i-1}C^* \le C/(1+\delta)$. Π_u will be examined by the dynamic program as per Cost-Split Case (ii) above. This contradicts the fact that the dynamic program missed the solution with the g facilities.

In the second case, this better solution is defined by combining the solutions to the two subproblems Π_1 and Π_2. Then, by induction we have solutions with the correct number of facilities in the entries for Π_1 and Π_2 which are indexed by the appropriate approximate cost; i.e., the table entries are indexed by costs C_1 and C_2 that are at most $(1+\delta)^{i-1}$ times the actual assignment cost for each subproblem. We show that at combination one can further lose an additional factor of at most $(1+\delta)$. Let Γ be the smallest power of $(1+\delta)$ such that $C_1 + C_2 \le \Gamma$. The idea is that this combination of Π_1 and Π_2 with actual service cost at most $C_1 + C_2$ will be considered by *any* subproblem, with appropriate interaction pattern, defined on R at recursion height i and whose cost index is at least Γ. In particular, we have that

$$\Gamma \le (1+\delta)(C_1 + C_2) \le (1+\delta)^i C^* \le C.$$

If $\Gamma < C$ (recall that C is also a power of $(1+\delta)$), the induction step gives us that the combination was examined as per Cost-Split Case (iii) when filling out the entry of the subproblem on R, with the same interaction pattern as Π but indexed by cost Γ. Therefore, it will be available when filling out the entry for Π and will be examined as per Cost-Split Case (i), which is a contradiction.

If $\Gamma = C$, we have that $C/(1+\delta) < C_1 + C_2 \le C$. Therefore, the combination will be examined when filling out the entry for Π as per Cost-Split Case (iii). Again this contradicts the fact that the dynamic program missed this solution. □

We are now ready to state the main result of the paper.

THEOREM 3. *Given an instance of the* k*-median problem in the* 2*-dimensional Euclidean space, and any fixed* $m > 0$, *there are randomized algorithms that compute* *a* $(1 + O(\log(m)/m))$*-approximation, in expectation, with worst-case running times* $O(2^{O(m)} n \log^8 n)$ *and* $O(2^{O(m)} \max\{k, \log^3 n\} n \log^2 n)$*, respectively.*

Repeating the algorithm $O(m \log n)$ times gives a $(1 + O(\log(m)/m))$ approximation guarantee with probability $1 - o(1)$. The algorithm can be easily extended to instances in the d-dimensional Euclidean space. The main difference is that now we work with rectangular parallelepipeds whose faces lie on i-allowable hyperplanes. As a consequence the bound given in Lemma 7 increases by an $O(m^{2d-4})$ factor and in the dynamic program we have to maintain data for each of the $2d$ faces of each parallelepiped where each face contains m^{d-1} portals. Moreover, the increased cost for range searching in Lemma 8 has to be taken into account. Following the same steps as in the proof of Theorem 3, we obtain the following.

THEOREM 4. *Given an instance of the* k*-median problem in the* d*-dimensional Euclidean space, and any fixed* $m > 0$*, there are randomized algorithms that compute a* $(1 + O(\log(m)/m))$*-approximation, in expectation, with worst-case running times* $O(2^{O(m^{d-1})} n \log^{d+6} n)$ *and* $O(2^{O(m^{d-1})} \max\{k, \log^{d+1} n\} n \log^2 n)$*, respectively.*

For any given accuracy $\varepsilon > 0$, we want $\ln(m)/m \le c\varepsilon$, where m is the number of portals and c an appropriately small constant. Let us assume $c = 1$, since we can always decrease the given value of ε. Using the computer algebra package Maple [7] we obtain that for $m = \ln(1/\varepsilon)/(0.5\varepsilon)$, $\ln(m)/m \le \varepsilon$ for any $\varepsilon \in (0, 0.5)$, while at the same time $m > 2$. On the other hand, setting $m = 3$ always yields an error less than 0.5. Therefore, the asymptotic number of portals required for an accuracy of ε in the objective is

$$O(\log(1/\varepsilon)/\varepsilon).$$

We obtain the following reinterpretation of Theorem 4.

THEOREM 5. *Given an instance of the* k*-median problem in the* d*-dimensional Euclidean space, and any fixed* $\varepsilon > 0$*, there are randomized algorithms that compute a* $(1 + \varepsilon)$*-approximation, in expectation, with worst-case running times*

$$O\left(2^{O\left((\frac{\log(1/\varepsilon)}{\varepsilon})^{d-1}\right)} n \log^{d+6} n\right) \text{ and } O\left(2^{O\left((\frac{\log(1/\varepsilon)}{\varepsilon})^{d-1}\right)} \max\{k, \log^{d+1} n\} n \log^2 n\right).$$

5.4. Uncapacitated facility location. In this section we extend the results to uncapacitated facility location. Our structure theorems clearly hold as far as the service cost is concerned. We need to implement a dynamic programming algorithm that also keeps track of the facility cost. In order to be able to preprocess the points so that they lie on a polynomial-size grid, we need the existence of a polynomial-factor approximation for the service cost as explained in section 2. *The discussion below is predicated on such an estimate being available.*

The second approach for the k-median problem enumerated the service cost and computed for each service cost value the corresponding minimum number of facilities. The analogue here is to compute the minimum facility cost. This yields an algorithm that computes a $(1 + \varepsilon)$-approximation in expectation with worst-case running time

$$O\left(2^{O\left((\frac{\log(1/\varepsilon)}{\varepsilon})^{d-1}\right)} n \log^{d+6} n\right).$$

Note that the latter time bound holds without any assumption on the range of the facility costs.

Acknowledgments. Many thanks to Sariel Har-Peled for valuable comments on a draft of this paper. The authors are grateful to Jan Remy for identifying a gap in the dynamic program and allowing us to include in the paper the fix he proposed.

REFERENCES

[1] P. K. Agarwal and J. Erickson, *Geometric range searching and its relatives*, in Advances in Discrete and Computational Geometry, B. Chazelle, J. E. Goodman, and R. Pollack, eds., Contemp. Math. 223, AMS, Providence, RI, 1999, pp. 1–56.

[2] S. Arora, *Polynomial-time approximation schemes for Euclidean TSP and other geometric problems*, J. ACM, 45 (1998), pp. 753–782.

[3] S. Arora, P. Raghavan, and S. Rao, *Approximation schemes for Euclidean k-medians and related problems*, in Proceedings of the 30th Annual ACM Symposium on Theory of Computing, ACM, New York, 1999, pp. 106–113.

[4] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit, *Local search heuristics for k-median and facility location problems*, SIAM J. Comput., 33 (2004), pp. 544–562.

[5] Y. Bartal, *Probabilistic approximation of metric spaces and its algorithmic applications*, in Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 1996, pp. 184–193.

[6] Y. Bartal, *On approximating arbitrary metrics by tree metrics*, in Proceedings of the 30th Annual ACM Symposium on Theory of Computing, ACM, New York, 1998, pp. 161–168.

[7] B. W. Char, K. O. Geddes, G. H. Gonnet, B. L. Leong, M. B. Monagan, and S. M. Watt, *Maple V Language Reference Manual*, Springer-Verlag, New York, 1991.

[8] M. Charikar, C. Chekuri, A. Goel, and S. Guha, *Rounding via trees: Deterministic approximation algorithms for group Steiner tree and k-median*, in Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 1998, pp. 114–123.

[9] M. Charikar and S. Guha, *Improved combinatorial algorithms for the facility location and k-median problems*, in Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 1999, pp. 378–388.

[10] M. Charikar, S. Guha, E. Tardos, and D. Shmoys, *A constant factor approximation algorithm for the k-median problem*, in Proceedings of the 31st Annual ACM Symposium on Theory of Computing, ACM, New York, 1999, pp. 1–10.

[11] K. Chen, *On k-median clustering in high dimensions*, in Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, SIAM, Philadelphia, 2006, pp. 1177–1185.

[12] F. A. Chudak, *Improved approximation algorithms for uncapacitated facility location*, in Proceedings of the 6th Conference on Integer Programming and Combinatorial Optimization, R. E. Bixby, E. A. Boyd, and R. Z. Ríos-Mercado, eds., Lecture Notes in Comput. Sci. 1412, Springer-Verlag, Berlin, 1998, pp. 180–194.

[13] G. Cornuéjols, G. L. Nemhauser, and L. A. Wolsey, *The uncapacitated facility location problem*, in Discrete Location Theory, P. Mirchandani and R. Francis, eds., John Wiley and Sons, Inc., New York, 1990, pp. 119–171.

[14] T. Feder and D. H. Greene, *Optimal algorithms for approximate clustering*, in Proceedings of the 20th Annual ACM Symposium on Theory of Computing, ACM, New York, 1988, pp. 434–444.

[15] T. Gonzalez, *Clustering to minimize the maximum intercluster distance*, Theoret. Comput. Sci., 38 (1985), pp. 293–306.

[16] S. Guha and S. Khuller, *Greedy strikes back: Improved facility location algorithms*, in Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, SIAM, Philadelphia, 1998, pp. 649–657.

[17] S. Har-Peled, *Clustering motion*, Discrete Comput. Geom., 31 (2004), pp. 545–565.

[18] S. Har-Peled and S. Mazumdar, *Coresets for k-means and k-median clustering and their applications*, in Proceedings of the 36th Annual ACM Symposium on Theory of Computing, ACM, New York, 2004, pp. 291–300.

[19] D. S. Hochbaum, *Heuristics for the fixed cost median problem*, Math. Programming, 22 (1982), pp. 148–162.

[20] K. Jain and V. V. Vazirani, *Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation*, J. ACM, 48 (2001), pp. 274–296.

[21] S. G. Kolliopoulos and S. Rao, *A nearly linear-time approximation scheme for the Euclidean k-median problem*, in Proceedings of the 7th Annual European Symposium on Algorithms, J. Nešetřil, ed., Lecture Notes in Comput. Sci. 1643, Springer-Verlag, Berlin, 1999, pp. 378–389.

[22] J. H. Lin and J. S. Vitter, *Approximation algorithms for geometric median problems*, Inform. Process. Lett., 44 (1992), pp. 245–249.

[23] J. H. Lin and J. S. Vitter, *ϵ-approximations with minimum packing constraint violation*, in Proceedings of the 24th Annual ACM Symposium on Theory of Computing, 1992, pp. 771–782.

[24] J. S. B. Mitchell, *Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems*, SIAM J. Comput., 28 (1999), pp. 1298–1309.

[25] S. Rao and W. D. Smith, *Improved approximation schemes for geometrical graphs via spanners and banyans*, in Proceedings of the 30th Annual ACM Symposium on Theory of Computing, ACM, New York, 1998, pp. 540–550.

[26] J. Remy and A. Steger, *Approximation schemes for node-weighted geometric Steiner tree problems*, in APPROX/RANDOM 2005; full version to appear in Algorithmica.

[27] D. B. Shmoys, É. Tardos, and K. I. Aardal, *Approximation algorithms for facility location problems*, in Proceedings of the 29th Annual ACM Symposium on Theory of Computing, ACM, New York, 1997, pp. 265–274.

SIAM J. COMPUT.
Vol. 37, No. 3, pp. 783–803

HAMMING CODES, HYPERCUBE EMBEDDINGS, AND FAULT TOLERANCE*

WILLIAM AIELLO[†] AND F. T. LEIGHTON[‡]

Abstract. In this paper, we devise a special 1-error-correcting code that enables us to embed the graph product of an $N/\log N$-node hypercube and a $\log N$-node complete graph into in an N-node hypercube with constant load, dilation, and congestion. We apply the result to construct improved embeddings of trees and other structures in a hypercube, and to design more efficient and robust algorithms for reconfiguring a hypercube around random or worst-case faults. The result has also been used subsequently by others to show that the N-node hypercube can emulate all N-node planar graphs with constant slowdown.

Key words. hypercube, graph embeddings, fault tolerance, reconfiguration, tree compression, tree embedding, emulation, worst-case faults, random faults

AMS subject classifications. 68M10, 68M12, 68M15, 68R10, 68W10, 90B18, 94C15

DOI. 10.1137/S0097539798332464

1. Introduction.

In this paper, we use the notion of a graph product to define a graph called the *hypercube of cliques* which is the product of an $N/\log N$-node hypercube and a $\log N$-node clique (also known as a complete graph). We then show that an N-node hypercube of cliques can be embedded into an N-node hypercube with load 1 and constant dilation and congestion. As a consequence, we find that the hypercube has even stronger structural, algorithmic, and fault-tolerant properties than previously realized.

To clarify terminology, an embedding σ of a network F into a network G is a mapping of nodes of F to nodes of G and edges of F to paths in G such that the endpoints of the path $\sigma(e)$ for any edge $e = (u, v)$ in F are $\sigma(u)$ and $\sigma(v)$. The *load* of an embedding is the maximum number of nodes of F assigned to any one node of G. The *dilation* is the maximum length in G of the image of an edge of F. The *congestion of an edge* of G is the number of edges of F whose image passes through the edge of G. The *congestion* of the embedding is the maximum congestion of any edge.

We denote the N-node (or n-dimensional) hypercube by H_N where $N = 2^n$ throughout. We denote the S-node clique by K_S. We define the N-node *hypercube of cliques* P_N as the product of $H_{N/2^s}$ with K_{2^s}, where $s = \lfloor \log(n + 1)\rfloor$. Formally, this is written as $P_N = H_{N/2^s} \otimes K_{2^s}$. The $\otimes$ operator will be defined in the next section. For the purposes of the introduction, P_N can be formed from H_N by inserting an edge between any pair of nodes whose first $n - s$ bits of address are the same. Note that for any $(n - s)$-bit binary string there are 2^s such nodes which are completely connected.

*Received by the editors January 12, 1988; accepted for publication (in revised form) December 4, 2006; published electronically July 11, 2007. A preliminary abstract of this paper appeared as *Coding theory, hypercube embeddings and fault tolerance*, in Proceedings of the 1991 ACM Symposium on Parallel Algorithms and Architectures, 1991, pp. 125–136. This research was supported by the Defense Advanced Research Projects Agency under contracts N00014-91-J-1698 and N00014-92-J-1799 and by the U.S. Air Force under contract AFOSR-49620-92-J-0125.

http://www.siam.org/journals/sicomp/37-3/33246.html

[†] Department of Computer Science, University of British Columbia, 201-2366 Main Mall, Vancouver, BC V6T 1Z4, Canada (head@cs.ubc.ca).

[‡] Mathematics Department and Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 (flt@math.mit.edu).

Obviously, H_N is a subgraph of P_N. But can P_N be nicely embedded into H_N? From the above characterization of P_N, it is clear that one could embed P_N into H_N with constant load given a constant-load embedding of a 2^s-node clique into a 2^s-node hypercube.

But, of course, any constant-load embedding of K_{2^s} into H_{2^s} must have congestion at least $\Omega(2^s/s) = \Omega(n/\log n)$ and dilation at least $\Omega(s) = \Omega(\log n)$.

In spite of this, we show that it is possible to construct an embedding of P_N in H_N with load 1, dilation $O(1)$, and congestion $O(1)$. The embedding is described in section 2 and makes use of 1-error-correcting codes in order to map the cliques of P_N to "stars" in H_N. Roughly speaking, a *star* consists of a node u whose binary address is a codeword together with its neighbors in the hypercube. The fact that hypercubes can be partitioned into stars corresponding to codewords is well known and has been exploited in many previous papers [2, 14, 18, 20].

In addition to mapping the cliques of P_N to the stars of H_N, we must find paths between the stars in H_N such that the stars have the connectivity of a lower dimensional hypercube. In order to find the paths, we devise an algorithm to find 1-error-correcting codes with two properties: (1) any one bit of the cleartext affects only a constant number of bits of the codeword (i.e., the codes are length preserving), and (2) any one bit of the codeword is a function of only a constant number of bits of the cleartext. It is well known how to generate codes with the first property. This property ensures that our embedding has constant dilation. The second property is novel to this paper, and it guarantees that the embedding has constant congestion. To the best of our knowledge, codes satisfying the latter property were not previously studied prior to the preliminary version of this paper [1].

As a consequence of proving that P_N can be nicely embedded in H_N, we can resolve several problems concerning hypercube embeddings and robustness. For example, an immediate corollary is the fact that an $O(\log N)$-dilated N-node butterfly can be embedded in an N-node hypercube with constant load, congestion, and dilation. (A network is said to be *k-dilated* if every edge is replaced with k parallel edges.[1]) The $O(\log N)$-dilated butterfly is a powerful network for circuit-switched routing [2], and packet-switched routing [16]. The best previously known embeddings of this network in H_N have dilation $\Omega(\log N)$ [2, 12] or congestion $\Omega(\log N)$.

In section 3, we give several algorithms for embedding trees in hypercubes. First, in section 3.1, we prove a general tree compression lemma. A similar lemma using different techniques can be found in [10]. Then in section 3.2 we use the tree compression lemma and the embedding of P_N into H_N to improve the results of Bhatt et al. [6], and Greenberg and Bhatt [12]. In particular, we show how to (off-line) embed any $O(\log N)$-dilated M-node binary tree into an N-node hypercube with $O((M/N)+1)$ load and $O(1)$ congestion and dilation. Embedding a $O(\log N)$-dilated tree into a hypercube is useful for simulating tree algorithms on a hypercube. If the interprocessor messages of the tree algorithm are large, say of $\Omega(\log^2 N)$ length, then one might be able to achieve an $O(\log N)$ speedup by breaking up the messages into $O(\log N)$ pieces and sending each piece on a different path. Alternatively, by using Reed–Solomon codes to break up the messages into $O(\log N)$ packets and sending each packet along a different path, as done by Rabin [22] for hypercube routing, one may be able to simulate the tree algorithm in the presence of edge faults. The off-line embeddings of Bhatt et al. [6] apply only to undilated trees and require $M = O(N)$

[1] Note that k-dilated and $O(k)$ dilation sound similar but have very different meanings. An embedding has $O(k)$ dilation if every edge of the embedded graph is stretched across at most $O(k)$ edges by the embedding.

in order to achieve constant load and congestion. While Greenberg and Bhatt [12] are able to embed $\log N$-dilated grids into the hypercube with constant load, dilation, and congestion, their embedding of $\log N$-dilated trees has dilation $\Theta(\log\log N)$ and requires $M = O(N)$ to achieve constant load and $\Theta(\log\log N)$ congestion.

In section 3.3 we show how to dynamically (on-line) embed any M-node binary tree in an N-node hypercube with $O((M/N) + 1)$ load, $O((M/N \log N) + 1)$ congestion, and $O(1)$ dilation. Dynamic embeddings are important since, for many parallel algorithms with tree structure, the growth pattern and eventual shape of the tree are dependent on the input and are not known at compile-time. Our dynamic embedding improves upon the on-line embedding of Leighton et al. [18] which has the same (optimal) load and dilation but congestion $\Theta((M/N) + 1)$ which is only optimal for $M = O(N)$.

In section 4 we apply our embedding of P_N into H_N to the problem of reconfiguring a hypercube around faults. Previously, Hastad, Leighton, and Newman [14] showed that if each component of H_N fails independently with any constant probability $p < 1$, then the functioning parts of the hypercube can be reconfigured using only local control to simulate the original hypercube with constant slowdown. The reconfiguration algorithm of [14] is probabilistic and runs in polylogarithmic time. In section 4.1, we show how to embed P_N in H_N with constant load, congestion, and dilation, even if the nodes and edges of H_N fail with some small constant probability $p \le p_0$. Since H_N is a subgraph of P_N, this effectively reconfigures H_N. Our embedding algorithm is deterministic and uses only $O(\log N)$ rounds of communication.

In section 4.2 we extend the reconfiguration algorithm to work for any faulty hypercube containing up to $\log^{O(1)} N$ worst-case faults. Previously, Bruck, Cypher, and Soroker [9] had shown that an N-node hypercube with $\log^{O(1)} N$ worst-case faults can implement certain restricted hypercube computations with constant slowdown. The best previous general simulations were by Becker and Simon [4] and Livingston et al. [11] and can tolerate only $O(\log\log N)$ worst-case faults. To obtain the $\log^{O(1)} N$-fault bound we define the r-fold hypercube of cliques, $P_{N,r}$, as

$$P_{N,r} = H_{N/2^{sr}} \otimes \overbrace{K_{2^s} \otimes \cdots \otimes K_{2^s}}^{r},$$

where $s = \Theta(\log n)$ is defined in the next section. For constant r, we show that $P_{N,r}$ can be embedded in H_N with constant load, dilation, and congestion. We then prove that $P_{N,r}$ can tolerate $O(\log^{r-1} N)$ faults. No upperbound is known on the number of worst-case faults that a hypercube can reconfigure around. We leave tight bounds on the number of worst-case faults as an interesting subject for future research.

Subsequent to [1], Kaklamanis, Krizanc, and Rao [15] have used the fact that $P_{N,r}$ can be embedded into H_N with constant load, dilation, and congestion to show that H_N can emulate any constant-degree planar graph with constant slowdown.

2. Embedding $H_{N/2^s} \otimes K_{2^s}$ into H_N.

The main result of this section is an embedding of the N-node hypercube of cliques $P_N = H_{N/2^s} \otimes K_{2^s}$, $s = \lfloor \log(\log N + 1) \rfloor$ into the N-node hypercube. The embedding has load 1 and constant congestion and dilation. Hence, H_N can simulate the P_N with a constant factor slowdown.

By virtue of the embedding, the hypercube of cliques is no more powerful than the hypercube. Nonetheless, several routing, embedding, and robustness problems are conceptually easier to solve when viewed on the hypercube of cliques. These will be dealt with in later sections.

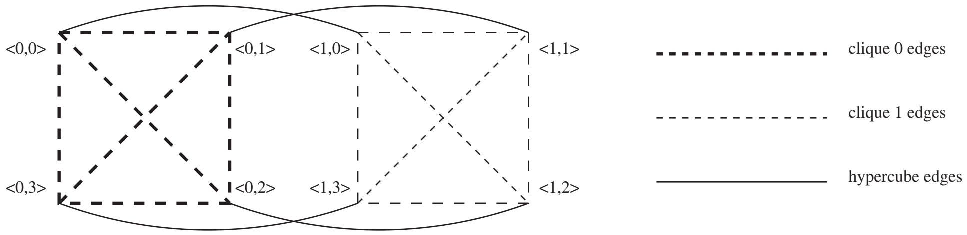


FIG. 1. $H_2 \otimes K_4 = P_8$.

Before we can state our main theorems, we need the following notation. Given two graphs, $G = (V, E)$ and $G' = (V', E')$, define the product graph $G \otimes G'$ to have a vertex set which is the cartesian product of V and V' and to have an edge set which is the union of "G edges"

$$\{(\langle v, s'\rangle, \langle u, s'\rangle) \mid (v, u) \in E,\ s' \in V'\}$$

and "G' edges"

$$\{(\langle s, v'\rangle, \langle s, u'\rangle) \mid s \in V,\ (v', u') \in E'\}.$$

$G^{(2)}$ will denote $G \otimes G$, and more generally, $G^{(r)}$ is G taken as a product with itself r times, $r \geq 1$.

We will denote the interval of consecutive integers from a to b inclusive by $[a,\ b]$. Let $N = 2^n$. The *complete graph* on S vertices K_S has vertex set $[0,\ S-1]$ and edge set $\{(i, j) \mid 0 \leq i \neq j \leq S-1\}$. The N-node *hypercube* H_N is a network with vertex set $\{0,1\}^n$. The edges of the network are between nodes whose labels differ by one bit. In what follows we will use the notation $u(j)$ to denote the jth bit of $u \in \{0,1\}^n$, $j \in [1,\ n]$. For $A \subset [1,\ n]$ we define u^A to be the node such that $u^A(k) = 1 - u(k)$ for $k \in A$ and $u^A(k) = u(k)$ for $k \notin A$. Define $A \Delta B$ to be the symmetric difference of A and B, i.e., $A \Delta B = A \cup B - A \cap B$. Then $(u^A)^B$ is equivalent to $u^{A \Delta B}$. When A has small cardinality we will often omit its brackets, e.g., $u^{\{j\}}$ will be written as u^j and denotes the node adjacent to u across the jth dimension of the hypercube. Since bit positions of a hypercube address start at 1, u^0 will mean u^ϕ which, of course, is just u. The hypercube edges are the set $\{(u, u^j)\}$, where $u \in \{0,1\}^n$ and $j \in [1,\ n]$. It is well known that $H_{2^n} = H_2^{(n)}$.

Define the *hypercube of cliques* with $N = 2^n$ nodes, P_N, as the product graph $H_{2^{n-s}} \otimes K_{2^s}$, where $s = \lfloor \log(n+1) \rfloor$. Note that if $n = 2^k - 1$ for some positive integer k, then $s = k$. Furthermore, $s = k$ for the 2^k values of n in the range $2^k - 1 \leq n < 2^{k+1} - 1$. The edges of P_N are the union of the "complete graph edges" $\{(\langle v, i\rangle, \langle v, j\rangle) \mid v \in \{0,1\}^{n-s},\ 0 \leq i \neq j \leq 2^s - 1\}$ and the "hypercube edges" $\{(\langle v, i\rangle, \langle v^j, i\rangle) \mid v \in \{0,1\}^{n-s},\ i \in [0,\ 2^s - 1],\ j \in [1,\ n-s]\}$. The hypercube of cliques for $n = 3 = 2^2 - 1$ is shown in Figure 1.

In addition to embedding the hypercube of cliques into a hypercube of the same size, we will also embed the *r-fold hypercube of cliques*, defined below, into a hypercube of the same size. The r-fold hypercube of cliques with $N = 2^n$ nodes, $P_{N,r}$, is defined as $H_{N/2^{rs}} \otimes K_{2^s}^{(r)}$, where s is the largest integer such that $2^s - 1 + (r-1)s \leq n$.

The embedding of P_N into H_N relies crucially on the notion of a star partition. This in turn relies on perfect 1-error-correcting codes which were first constructed by Hamming [13] for binary strings of length $2^k - k - 1$ for some integer k. More specifically, Hamming showed that there is a mapping g from words m of length $2^k - k - 1$ to codewords $g(m)$ of length $2^k - 1$ such that every string of length $2^k - 1$ is either a codeword or Hamming distance 1 from exactly one codeword.

To describe the mapping g, define C_k to be the k by $2^k - 1$ matrix where the columns are all the binary strings of length k except for the all-zero string. Consider strings of length l as column vectors in the vector space $GF(2)^l$. That is, vector addition is simply bitwise exclusive-or, and matrix-vector multiplication occurs with the vector on the right. It is a property of perfect 1-error-correcting codes that $C_k \cdot y = 0$ iff y is a codeword, i.e., iff $y = g(m)$ for some m. That is, the nullspace of C_k consists precisely of the codewords. It is well known that the nullspace of C_k has dimension $2^k - k - 1$ and so has a basis set $B = \{v_1, \ldots, v_{2^k-k-1}\}$ of $2^k - k - 1$ basis vectors. We will also let B denote the natural $2^k - 1$ by $2^k - k - 1$ matrix formed by the column vectors v_i. Given a basis B, the following mapping g clearly generates all the codewords of length $2^k - 1$ from strings m of length $2^k - k - 1$:

$$g(m) = B \cdot m.$$

Consider the following measures of a basis. Let the *weight* of a vector (column) be the number of 1's in the vector. Define the *height* of the basis as the maximum weight of any vector in the basis. Let the *width* of a dimension (row) be the number of basis vectors with a 1 in that dimension. Define the *width of the basis* as the maximum width of any dimension.

The lemma below shows that a basis with small height and width yields a good embedding of P_N into H_N. After proving this lemma we will show how to find such a basis.

LEMMA 1. *Given a basis of the nullspace of C_k with height h and width w for some $k \geq 2$, there is an embedding of $H_{2^{n-k}} \otimes K_{2^k}$ into H_{2^n} with load 1, dilation h, and congestion $2w + 2$ for all $n \geq 2^k - 1$.*

Proof. Assume we are given a basis B for the nullspace of C_k with height h and width w for some $k \geq 2$, and let g be the the encoding from strings of length $\eta - k$ to η defined by B. We will first describe the embedding of $H_{2^{\eta-k}} \otimes K_{2^k} = P_{2^\eta}$ into H_{2^η}, where $\eta = 2^k - 1$. We will then show that this embedding can be easily extended to an embedding of $H_{2^{n-k}} \otimes K_{2^k}$ into H_{2^n} for all $n \geq \eta$. For simplicity we will treat both H_{2^η} and P_{2^η} as directed graphs where each simple edge is replaced by two directed edges, one in each direction.

Call a node u in H_{2^η} a *star center* if u is a codeword, i.e., if $u \in \{g(m) | m \in \{0,1\}^{\eta-k}\}$. Define a *star* to be a star center together with all of its neighbors. Since g is a perfect 1-error-correcting code, every node in the hypercube is in exactly one star. That is, the stars partition the nodes of H_{2^η}: there are $2^{\eta-k}$ stars each with $\eta + 1 = 2^k$ nodes.

The $2^{\eta-k}$ cliques of size 2^k in $P_{2^\eta} = H_{2^{\eta-k}} \otimes K_{2^k}$ will be mapped to the $2^{\eta-k}$ stars of size 2^k in H_{2^η}. Furthermore, cliques which differ in one dimension in P_{2^η} will be mapped to stars which are "close" in H_{2^η}.

Our map from nodes of P_{2^η} to nodes of H_{2^η} is defined as follows:

$$\langle m, i \rangle \mapsto g(m)^i$$

for $m \in \{0, 1\}^{\eta-k}$ and $i \in [0, \eta]$. Observe that each clique is mapped to a unique star and that each node in a clique is mapped to a unique node in the corresponding star. Since the stars partition H_{2^η}, the embedding is one-to-one and onto.

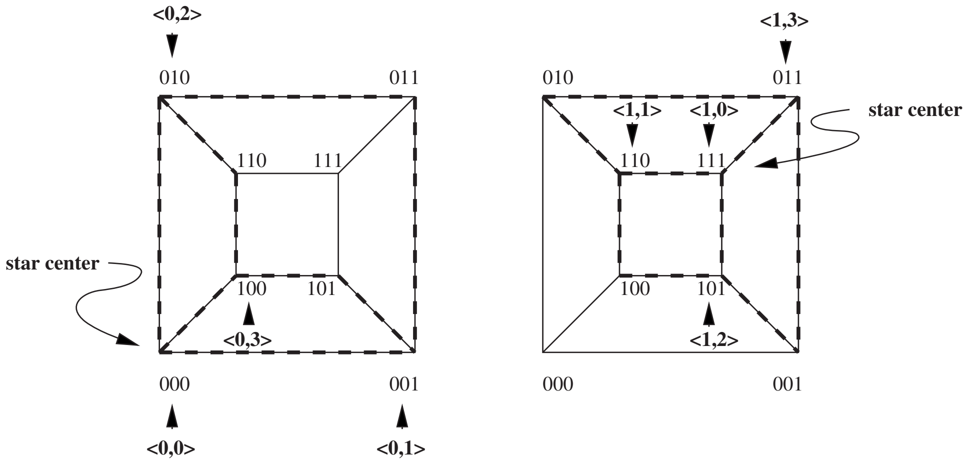


FIG. 2. *The embedding of clique* 0 *of* $H_2 \otimes K_4$ *into* H_8 *on the left, and the embedding of clique* 1 *into* H_8 *on the right.*

Embedding the clique edges of P_{2^η} is easy. The edge $\langle m, 0\rangle \rightarrow \langle m, j\rangle$, $j \in [1,\, \eta]$, is mapped to the edge $g(m) \rightarrow g(m)^j$ in H_{2^η}. The edge $\langle m, i\rangle \rightarrow \langle m, j\rangle$, $1 \le i \ne j \le \eta$, is mapped to the path $g(m)^i \rightarrow g(m)^{i,j} \rightarrow g(m)^j$. It is easy to verify that a directed edge of H_{2^η} is used at most once as the first edge of these paths and at most once as the second edge. Hence, the directed congestion due to the clique edges of P_{2^η} is at most 2. The mapping of the nodes and clique edges of P_8 to H_8 is shown in Figure 2.

To embed the hypercube edges of P_{2^η}, $\{\langle m, j\rangle \rightarrow \langle m^i, j\rangle \mid j \in [0,\, \eta],\, i \in [1,\, \eta - k]\}$, observe that $g(m^i) = g(m) \oplus v_i$ since

$$g(m^i) = B \cdot m^i = (B \cdot m) \oplus v_i = g(m) \oplus v_i.$$

This can also be written as $g(m^i) = g(m)^{V_i}$ if we let V_i be the indicies of the bits of v_i which are 1. The edge $\langle m, j\rangle \rightarrow \langle m^i, j\rangle$ is mapped to the path which starts at $g(m)^j$ and goes to $g(m^i)^j$ by changing the bits indicated by V_i. More formally, let $V_i = \{l_{i_1},\, l_{i_2}, \ldots, l_{i_{h'}}\}$, where $l_{i_1} < l_{i_2} < \cdots < l_{i_{h'}}$. Then the edge $\langle m, j\rangle \leftrightarrow \langle m^i, j\rangle$ is mapped to the path

$$g(m)^{\{j\}} \leftrightarrow g(m)^{\{l_{i_1}\}\Delta\{j\}} \leftrightarrow g(m)^{\{l_{i_1}, l_{i_2}\}\Delta\{j\}} \leftrightarrow \cdots \leftrightarrow g(m)^{V_i \Delta \{j\}} = g(m^i)^j.$$

That the dilation for embedding the hypercube edges of P_{2^η} is h follows directly from the embedding above and from the definitions of dilation and height. Note that h is an upper bound on the dilation for the entire embedding since Hamming codes must have height at least 3 and the dilation due to the embedding of the clique edges is just 2.

To bound the congestion in H_{2^η} due to the hypercube edges of P_{2^η}, consider a hypercube edge $u \leftrightarrow u^j$ across dimension j. Let v be a basis vector with a 1 in dimension j, and further suppose v has 1's in dimensions $V = \{l_1, \ldots, l_d,\, j,\, l_{d+2}, \ldots, l_{h'}\}$, where $l_1 < l_2 < \cdots < l_d < j < l_{d+2} < \cdots < l_{h'}$. If we start at node $s = u^{\{l_1, \ldots, l_d\}}$ and take a path defined by changing the dimensions of V in increasing order, we will pass through the edge $u \rightarrow u^j$ and end up at the node $t = u^{\{j, l_{d+2}, \ldots, l_{h'}\}}$. The above path may be a path of the embedding which passes through $u \leftrightarrow u^j$ if the appropriate nodes of P_{2^η} have been mapped to s and t. Note that a path, which starts at s^j, traverses the dimension of V in increasing order, and ends at t^j, traverses the edge $u^j \to u$. If the appropriate nodes of P_{2^η} have been mapped to s^j and t^j, then this latter path will also pass through $u \leftrightarrow u^j$. Hence, every basis vector with a 1 in dimension j may contribute congestion 2 to the edge $u \leftrightarrow u^j$. Obviously paths based on basis vectors which don't have a 1 in dimension j cannot cause congestion to $u \leftrightarrow u^j$ since these paths do not even cross dimension j. Hence, the congestion in H_{2^η} due to hypercube edges of P_{2^η} is at most $2w$.

This completes the embedding of $H_{2^{n-k}} \otimes K_{2^k}$ into H_{2^n} for the special case $n = \eta$. To get an embedding of all $n \geq \eta$ the following lemma will be useful.

LEMMA 2. *Given an embedding $\mathcal{E}$ of F into G with load l, congestion c, and dilation d and an embedding $\mathcal{E}'$ of F' into G' with load l', congestion c', and dilation d', there is a natural embedding of $F \otimes F'$ into $G \otimes G'$, called the product embedding, which achieves load equal to ll', congestion equal to* $\max\{cl', c'l\}$*, and dilation equal to* $\max\{d, d'\}$.

Proof. The product embedding maps nodes $\langle u, u' \rangle$ to $\langle \mathcal{E}(u), \mathcal{E}'(u') \rangle$. If an edge (u, v) of F is mapped to a path $(\mathcal{E}(u), w_1, \ldots, w_k, \mathcal{E}(v))$ in G, then the F edge $(\langle u, u' \rangle, \langle v, u' \rangle)$ of $F \otimes F'$ is mapped to the path

$$(\langle \mathcal{E}(u), \mathcal{E}'(u') \rangle, \langle w_1, \mathcal{E}'(u') \rangle, \ldots, \langle w_k, \mathcal{E}'(u') \rangle, \langle \mathcal{E}(v), \mathcal{E}'(u') \rangle)$$

in $G \otimes G'$. F' edges are mapped similarly. The claimed bounds follow easily from these mappings. □

Since H_{2^n} is equivalent to H_2 taken as a product with itself n times, it follows that H_{2^n} can be written as the product of two smaller hypercubes. Hence we can write H_{2^n} as $H_{2^{n-\eta}} \otimes H_{2^\eta}$ and $H_{2^{n-k}} \otimes K_{2^k}$ as $H_{2^{n-\eta}} \otimes H_{2^{\eta-k}} \otimes K_{2^k}$ which is just $H_{2^{n-\eta}} \otimes P_{2^\eta}$. (Recall that we are assuming that $n \geq \eta$ and $k \geq 2$ so that $\eta - k > 0$.) Now we use the product embedding consisting of the identity embedding of $H_{2^{n-\eta}}$ into itself along with the previously described embedding of P_{2^η} into H_{2^η}. This finishes the proof of Lemma 1. □

Lemma 1 along with a basis B for the nullspace of C_k with constant height and width for all k would yield an embedding P_{2^n} into H_{2^n} with load 1 and constant dilation and congestion for all n. We describe an algorithm for constructing such a basis in Appendix A where we prove the following.

LEMMA 3. *A basis of the nullspace of C_k with height* 6 *and width* 9 *can be found in time polynomial in* 2^k.

Our original bounds in [1] were height 6 and width 28. The width was improved in [17] to 11 before being further improved to 9 as presented here. Subsequently, Pritikin [21] improved both the height and the width to 3 and 5, respectively.

For many values of k one can do even better. For example, it is well known that cyclic codes based on primitive polynomials for $GF(2)[x]$ with degree k and t nonzero coefficients give a basis for the nullspace of C_k with height t. In Appendix A we give an easy proof that the width of the basis is also bounded by t. Empirical work has shown that for most degrees up to 137, the primitive polynomials have three nonzero coefficients [23]. This means that for most practical values of k, the height and width are both 3.

Finally, we can state our main theorem.

THEOREM 1. *There is an embedding of $H_{2^{n-k}} \otimes K_{2^k}$ into H_{2^n} with load* 1*, dilation* 3*, and congestion* 12 *for all $k \geq 2$ and $n \geq 2^k - 1$. For most $n \in [3, 2^{137} - 1]$ the congestion can be reduced to* 8.

As an obvious corollary, P_{2^n} can be embedded into H_{2^n} with the stated bounds for $n \geq 3$.

The above theorem can also be used as a base case to yield an embedding for the r-fold hypercube of cliques.

Theorem 2. *There is an embedding of $H_{2^{n-rk}} \otimes K_{2^k}^{(r)}$ into H_{2^n} with load* 1*, dilation* 3^r*, and congestion* 12^r *for all $k \geq 2$ and $n \geq 2^k - 1 + (r-1)k$.*

Proof. The proof will proceed by induction on r. The base case $r = 1$ is simply Theorem 1 above. Assume Theorem 2 is true for $r \leq t - 1$. Now given an $n \geq 2^k - 1 + (t-1)k$ we will embed $H_{2^{n-tk}} \otimes K_{2^k}^{(t)}$ into H_{2^n}. Note that $H_{2^{n-tk}} \otimes K_{2^k}^{(t)}$ can be written as $H_{2^{(n-k)-(t-1)k}} \otimes K_{2^k}^{(t-1)} \otimes K_{2^k}$. The first two terms in this product can be embedded into $H_{2^{n-k}}$ by induction since $n - k \geq 2^k - 1 + ((t-1)-1)k$ follows immediately from the assumption that $n \geq 2^k - 1 + (t-1)k$. The embedding has load 1, dilation 3^{t-1}, and congestion 12^{t-1}. Hence, $H_{2^{n-tk}} \otimes K_{2^k}^{(t-1)} \otimes K_{2^k}$ can be embedded into $H_{2^{n-k}} \otimes K_{2^k}$ with load 1, dilation 3^{t-1}, and congestion 12^{t-1} using the product embedding. The latter can now be embedded into H_{2^n} using the base case. This multiplies the dilation and congestion by 3 and 12, respectively. □

It follows immediately that $P_{N,r}$ can be embedded into H_N with load 1 and constant load and congestion for constant r.

3. Embedding trees into hypercubes.

3.1. The tree compression lemma. This section describes several results on embedding binary trees into the hypercube. A main tool for these embeddings will be the following lemma which is proved in Appendix B. A similar lemma using different techniques appears in [10].

Lemma 4 (the tree compression lemma). *For any M-node binary tree T and any positive integer $N < M$, it is possible to partition T into N subtrees by removing $N - 1$ edges so that every subtree has at most $\frac{12M}{N} + 1$ nodes. Furthermore, if each subtree is considered a single supernode, then the $N - 1$ removed edges and the N supernodes form a binary tree.*

As our first application of the tree compression lemma, we will generalize the result of Bhatt et al. [6] who show that any binary tree of size N can be embedded into H_N with load 1, congestion $O(1)$, dilation $O(1)$, and node congestion $O(1)$. By *node congestion* we mean the maximum number of paths of the embedding that pass through a hypercube node.[2]

Theorem 3. *Any M-node binary tree can be embedded into H_N with load $O(\frac{M}{N} + 1)$, congestion $O(1)$, dilation $O(1)$, and node congestion $O(1)$.*

Proof. Given an M-node tree T, $M > N$, use the tree compression lemma to get a tree T' with N supernodes, each consisting of subtrees of size $O(M/N)$. Now use the theorem of [6] to embed T' into H_N. This clearly gives $O(1)$ dilation, congestion, and node congestion. It also maps one supernode onto each hypercube node which gives the desired bounds on the load. □

3.2. Embedding dilated trees in a hypercube. As our second application of the tree compression lemma, we will show how to embed $O(\log N)$-dilated binary trees in a hypercube with constant dilation and congestion. A tree is said to be k-dilated if every edge is replaced with k edges.

Greenberg and Bhatt [12] studied the problem of embedding N-node, $O(\log N)$-dilated grids, and $O(\log N)$-dilated trees into H_N. Their motivation was as follows. If

[2] We do not count paths where both endpoints are mapped to the same node. Using such a definition, one cannot do better than $O(M/N)$ node congestion. Counting paths where one endpoint is mapped to a node does not change the stated results.

one is simulating a tree or grid algorithm on a hypercube in which the interprocessor messages are large, say of $\Omega(\log^2 N)$ length, then one might be able to achieve an $O(\log N)$ speedup by breaking up the messages into $O(\log N)$ pieces and sending each piece on a path which is short and has low congestion. They were able to embed the $O(\log N)$-dilated grid into H_N with $O(1)$ load, congestion, and dilation, but they were able to achieve only $O(\log\log N)$ dilation and congestion for $O(\log N)$-dilated binary trees. In what follows, we show how to achieve $O(1)$ dilation and congestion for $O(\log N)$-dilated binary trees.

Theorem 4. *Any N-node $O(\log N)$-dilated binary tree can be embedded into H_N with $O(1)$ congestion, dilation, and load.*

Proof. Let T be an $O(\log N)$-dilated tree with $N = 2^n$ nodes. Apply the tree compression lemma to get an $O(\log N)$-dilated binary tree T' with $N/2^s$ supernodes, where as before $s = \lfloor \log(n+1) \rfloor$. Each supernode is an $O(\log N)$-dilated subtree of size $O(2^s)$. Let T'' be the binary tree derived from T' by converting supernodes to nodes and dilated edges to edges. Below we will embed T' into $T'' \otimes K_{2^s}$ with dilation 3 and constant load and congestion. This will imply the theorem as follows. T'' can be embedded into $H_{N/2^s}$ with $O(1)$ load, dilation, and congestion by [6]. Hence $T'' \otimes K_{2^s}$ can be embedded into $H_{N/2^s} \otimes K_{2^s} = P_N$ with constant load, dilation, and congestion by Lemma 2. P_N is then embedded into H_N by Theorem 1.

Our embedding of T' into $T'' \otimes K_{2^s}$ is as follows. Each supernode v of T' is embedded into the clique labeled v in $T'' \otimes K_{2^s}$. The $O(2^s)$ nodes of a supernode v are assigned evenly to the 2^s nodes of clique v to achieve constant load. To analyze the dilation and congestion we'll analyze the dilated edges within a supernode and between supernodes separately.

If two neighbors within supernode v of T' are mapped to different nodes of clique v, then the $O(\log N)$ edges in the dilated edge between them are mapped evenly to the $2^s - 1$ paths of length 1 or 2 between the endpoints in clique v. Note that a pair of neighbors within supernode v that are mapped to the pair of nodes $\{x, y\}$ or the pair $\{x, z\}$ or the pair $\{w, y\}$ in clique v contribute $O(1)$ congestion to the edge (x, y) of clique v. Since $O(1)$ nodes of supernode v are mapped to any one node of clique v and since each node of supernode v has at most three neighbors, it follows that the congestion due to neighbors within supernodes of T' is $O(1)$.

Suppose supernodes u and v are neighbors in T' by virtue of the fact that nodes $U \in u$ and $V \in v$ are neighbors in T. U and V are mapped to nodes in neighboring cliques u and v in $T'' \otimes K_{2^s}$. Between any two nodes, one each in neighboring cliques in $T'' \otimes K_{2^s}$, there are 2^s paths of length at most 3. Hence, the $O(\log N)$ edges in the dilated edge between U and V can be evenly distributed among 2^s paths of length at most 3. These paths contribute at most $O(1)$ congestion to the clique edges of $T'' \otimes K_{2^s}$ and $O(1)$ congestion to the T'' edges of $T'' \otimes K_{2^s}$. □

Corollary 1. *Any M-node, $O(\log N)$-dilated binary tree can be embedded into H_N with load $O(\lceil M/N \rceil)$, congestion $O(1)$, and dilation $O(1)$.*

Proof. Apply the tree compression lemma to get a dilated tree with supernodes of size $O(\lceil M/N \rceil)$. Now apply Theorem 4 treating each supernode as a node. □

3.3. Embedding dynamic trees. The previous section dealt with off-line embeddings of arbitrary trees. However, for many parallel algorithms with a tree structure, the growth pattern and eventual shape of the tree are dependent on the input. Bhatt and Cai [5] were the first to examine whether dynamic trees could be efficiently simulated on the hypercube. Their results were improved by Leighton et al. [18] who gave a probabilistic embedding of a dynamic tree of size M into H_N which achieved, with high probability, $O(\frac{M}{N}+1)$ load, $O(1)$ dilation, and $O(\frac{M}{N}+1)$ congestion. Their bounds for load and dilation are clearly optimal (and, in fact, they show that a deterministic algorithm cannot simultaneously achieve optimal load and dilation). Using an algorithm similar to theirs we improve the congestion to $O(\frac{M}{N\log N}+1)$, which is a $\log N$ factor improvement. This is within a $\log(M/N)$ factor of the lower bound $\Omega(M/N\log N\log(M/N))$ on congestion for on-line algorithms implied in the work of Bhatt et al. [7].

THEOREM 5. *An arbitrary binary tree T with M vertices can be dynamically grown on an N-processor hypercube with constant dilation such that with high probability the maximum load per processor is $O(\frac{M}{N}+1)$ and the maximum congestion on any edge is $O(\frac{M}{N\log N}+1)$.*

Proof. The starting point of our proof is an algorithm by Leighton et al. [18] which dynamically embeds an M-node binary tree into an N-input, $\log N$-depth butterfly B_N with dilation 2 such that with high probability the number of tree nodes per column is $O(\frac{M}{N}+\log N)$ and the load is $O(\frac{M}{N\log N}+\log N)$.

The essential idea will be to use the algorithm of [18] on a butterfly $B_{N/2^s}$ of size $N/2^s$ by $\log N-s$ and then adapt this algorithm to run on $P_N=H_{N/2^s}\otimes K_{2^s}$ (where as before, $N=2^n$ and $s=\lfloor\log(n+1)\rfloor$). Observe that the algorithm of [18] will dynamically embed an M-node tree on $B_{N/2^s}$ with dilation 2 such that with high probability there are $O(\frac{M}{N/\log N}+\log N)$ nodes per column and $O(\frac{M}{N}+\log N)$ load. Call this algorithm A. Algorithm A' will be the adapted version of this algorithm which runs on P_N.

We will rely on the following simple combinatorial lemma to help distribute load and congestion.

LEMMA 5. *Assume a total of m balls are placed into n boxes one at a time. If at each step a ball is placed in any one of the $\lceil\alpha n\rceil$ least occupied boxes, $0<\alpha<1$, then after all balls have been assigned, at most $\frac{m}{n(1-\alpha)}+1$ balls are in any box.*

Proof. Let $\lceil\alpha n\rceil=k$. Let b_i be the number of balls in box i, where the boxes are sorted by increasing size after each ball is added. We first argue by induction that $b_n-1\le b_k\le b_n$. After the first ball has been added, $b_n=1$ and $b_k=0$. Now assume the statement is true after many balls have been added. When the next ball is added, the algorithm requires that it be added to a box i with $b_i<b_k$ or a box j with $b_j=b_k$. In the former case the inductive hypothesis clearly remains true. In the latter case, there are two cases. If $b_k=b_n$, then adding a ball to any box j with $b_j=b_k$ and resorting will result in $b_n=b_k+1$. If $b_k=b_n-1$, then adding a ball to any box j with $b_j=b_k$ and resorting will result in $b_k=b_n$ or $b_k=b_n-1$ depending on the configuration of boxes before the ball is added.

Now suppose to the contrary of the lemma that more than $\frac{m}{n(1-\alpha)}+1$ balls are in one box. The fact that $b_n-1\le b_k\le b_n$ is an invariant implies that there are at least $n-\lceil\alpha n\rceil+1$ boxes with more than $\frac{m}{n(1-\alpha)}$ balls. Since $n-\lceil\alpha n\rceil+1$ is strictly greater than $n-\alpha n$, this implies that there are more than m balls. □

If A embeds a tree node in column v in $B_{N/2^s}$, then A' will embed the same tree node in clique v in P_N. If A embeds a root u into column v, then A' will map u to one of the $\lceil\frac{4}{5}2^s\rceil$ least loaded nodes of clique v. We will now describe how the children of an arbitrary node u are embedded. To begin, assume that A has embedded u to some node $\langle v,i\rangle$ of $B_{N/2^s}$ and that A' has embedded u into $\langle v,j\rangle$ in P_N for some j. Algorithm A will only embed the children of u in column v, v^{i+1}, or v^{i+2} of $B_{N/2^s}$ (i.e., in $\langle v,i+1\rangle$, $\langle v^{i+1},i+1\rangle$, $\langle v,i+2\rangle$, or $\langle v^{i+2},i+2\rangle$). A' will handle these three cases as follows. In the first case, A' will map the child to $\langle v, k\rangle$ such that $\langle v, k\rangle$ is among the $\lceil \frac{4}{5} 2^s \rceil$ least loaded nodes of clique v and $\langle v, j\rangle \rightarrow \langle v, k\rangle$ is among the $\lceil \frac{4}{5} 2^s \rceil$ least congested edges with endpoint $\langle v, j\rangle$. Note that such an embedding can always be found.

Now suppose that A embeds the child in column v^{i+1}. A' embeds the node across a path of length 2:

$$\langle v, j\rangle \rightarrow \langle v, k\rangle \rightarrow \langle v^{i+1}, k\rangle$$

such that the first edge is among the $\lceil \frac{4}{5} 2^s \rceil$ least congested edges with endpoint $\langle v, j\rangle$, the second edge is among the $\lceil \frac{4}{5} 2^s \rceil$ least congested edges between clique v and clique v^{i+1}, and $\langle v^{i+1}, k\rangle$ is among the $\lceil \frac{4}{5} 2^s \rceil$ least loaded nodes in clique v^{i+1}. Note that such a path can always be found.

When A embeds a child in column v^{i+2}, A' embeds the node across a path of length 3:

$$\langle v, j\rangle \rightarrow \langle v, k\rangle \rightarrow \langle v^{i+1}, k\rangle \rightarrow \langle v^{i+2}, k\rangle$$

such that the first edge is among the $\lceil \frac{4}{5} 2^s \rceil$ least congested edges with endpoint $\langle v, j\rangle$, the second edge is among the $\lceil \frac{4}{5} 2^s \rceil$ least congested edges across dimension $i+1$, the third edge is among the $\lceil \frac{4}{5} 2^s \rceil$ least congested edges across dimension $i+2$, and $\langle v^{i+2}, k\rangle$ is among the $\lceil \frac{4}{5} 2^s \rceil$ least loaded nodes in clique v^{i+2}. As before, such a path can always be found.

That the load is $O(\lceil M/N \rceil)$ with high probability easily follows from Lemma 5 and the fact that the number of tree nodes per column in A is $O(\frac{M}{N} \log N + \log N)$ with high probability. To bound the congestion on clique edges of P_N use the fact that the load of A' is $O(\lceil M/N \rceil)$ with high probability. By the fact that each tree node has at most two children and by Lemma 5, the congestion is $O(\lceil \frac{M}{N \log N} \rceil)$. To bound the congestion on the hypercube edges of P_N, first note that the congestion on the butterfly edges due to A is at most $O(\frac{M}{N} + \log N)$ with high probability. This follows from the fact that the load due to A is $O(\frac{M}{N} + \log N)$ with high probability and the fact that each tree node has at most two children. That the congestion on hypercube edges of P_N is $O(\lceil \frac{M}{N \log N} \rceil)$ with high probability follows from Lemma 5 and the fact that each embedding by A' uses at most two hypercube edges when A uses a butterfly crossedge. □

In the model just analyzed only one node is expanded at a time. It is more natural to allow many tree nodes to expand in one time step. Because the algorithm embeds nodes locally, it is not difficult to adapt the algorithm to run in an on-line fashion with a polylogarithmic slowdown on the hypercube, even when many nodes are expanded simultaneously.

4. Reconfiguring hypercubes around faults.

In this section, we show how to apply the techniques developed in sections 2 and 3 to devise improved algorithms for using faulty hypercubes to simulate fault-free hypercubes. We will consider the case of random faults in section 4.1 and the case of worst-case faults in section 4.2.

4.1. Random faults.

Consider an N-node hypercube H_N in which every edge and node is faulty with some constant probability $p < 1$. We will assume that each fault is independent of the location of other faults and that if a node is faulty, then it cannot be used for any purpose (i.e., no communication can pass through it). In what follows, we will prove that if p is a sufficiently small constant, then with high probability we can embed P_N into H_N with constant load, dilation, and congestion so that every node of P_N is mapped to a live node of H_N and so that every edge of P_N is mapped to a path of live nodes and edges of H_N. Since H_N is a subgraph of P_N, we will have thus shown how a faulty hypercube can simulate a functioning hypercube of the same size with constant slowdown.

The result is similar to that proved by Hastad, Leighton, and Newman [14], except that the reconfiguration algorithm in [14] is probabilistic and uses a polylogarithmic number of communication steps to achieve the reconfiguration. The only previously known deterministic reconfiguration algorithm requires $\Omega(N)$ steps [3]. The algorithm described in this section is simple and deterministic and uses only $O(\log N)$ local communication steps. The algorithm is limited by the fact that the probability of component failure p must be a small constant, but it should be possible to extend the result to work for all $p < 1$ using the methods of [14] and section 4.2 of this paper.

THEOREM 6. *There is a deterministic embedding algorithm using $O(\log N)$ local communication rounds such that if the components of H_N fail independently with probability $p \leq p_0$ (for some fixed constant p_0), then P_N can be embedded in the faulty H_N with $O(1)$ load, dilation, and congestion with probability $1 - 1/N^{O(1)}$.*

Proof. The algorithm for embedding P_N into a faulty hypercube is very similar to the fault-free embedding described in section 2. To economize on space, we will only sketch the additional steps needed to tolerate faults in what follows. We will assume that $N = 2^n$, where $n = 2^k - 1$ for some integer k, so that $P_N = H_{2^{n-k}} \otimes K_{2^k}$. As in section 2 the extension of the result to general n is straightforward.

The first step of the embedding is to map the nodes of each clique of P_N to the nodes of the corresponding star in H_N. For ease of exposition we will assume that every star center is faulty. It is easy to show that, for any $\beta > 0$ and any $\epsilon_1 > 0$, there exists a sufficiently small p (e.g., $p \leq 2^{-1-\beta/\epsilon_1}$) such that with probability at least $1 - 1/N^\beta$, at least $(1-\epsilon_1)n$ nodes of each star are alive. Hence, we will map the ith node of a clique to the ith live node of the corresponding star, using wraparound as necessary. In particular, if $\epsilon_1 \leq 1/2$, then at most two nodes of any clique will be mapped to any star node by this process.

We next show how to interconnect the live nodes of each star into a clique. Given any two live nodes v^i, v^j in a star centered at v, we will connect them with the path $v^i \to v^{ij} \to v^j$ provided that the path is fault-free. Note that the probability that such a path is faulty is $1 - (1-p)^3 < 3p$ and that the path for $v^i \to v^j$ is disjoint from the path for $v^{i'} \to v^{j'}$ whenever $\{i, j\} \neq \{i', j'\}$. Hence, we can use a simple probabilistic argument to show that, with high probability, every live node of every star is connected to at least $(1 - \epsilon_2)n$ live nodes in the same star with live disjoint paths of length 2. These connections can be made in constant time on the hypercube.

We next show how to connect a pair of live nodes in a star v^i, v^j for which the path $v^i \to v^{ij} \to v^j$ contains a fault. For each such pair we will find a third live node v^s for which the paths $v^i \to v^{is} \to v^s$ and $v^s \to v^{sj} \to v^j$ are functioning. There are at least $(1-2\epsilon_2)n$ choices for s since both v^i and v^j are connected to at least $(1-\epsilon_2)n$ other live nodes with paths of length 2. In order to avoid congestion problems, we will choose intermediate nodes in rounds as follows. In the first round, we will find intermediate nodes for disconnected live pairs v^i, v^j, where $j \equiv i + 1 \pmod{n}$. In particular, the processors at node v^i will select a value of s such that both $v^i \to v^{is} \to v^s$ and $v^s \to v^{sj} \to v^j$ are live where $j = i + 1$. This step can be accomplished with $O(1)$ communication steps (where we allow $\log N$ bits to traverse each edge during a step) and can increase the congestion of every edge by at most 2 (due to reconfiguring this star). (Note that every edge in a path of length 4 is involved in reconfiguring at most two stars.) In the rth round $(1 \leq r \leq n)$, we will find intermediate nodes for live pairs

v^i, v^j, where $j \equiv i + r \pmod{n}$. In particular, the processor at node v^i will select a value of s such that every component along the path $v^i \to v^{is} \to v^s \to v^{sj} \to v^j$ is alive and so that every edge along the path has congestion (due to reconfiguring this star) at most C, where C is a predetermined, sufficiently large constant. We can always find such an s since at most $2\epsilon_2 n$ values of s are eliminated due to faults and since at most $2n/C$ values are eliminated due to edge congestion. (Hence, it suffices to choose $C > 2/(1 - 2\epsilon_2)$.) Information about whether or not edges are congested can be transmitted in $O(1)$ communication steps, where we allow $O(\log N)$ bits of data to be transmitted at each step. Note that we don't need to coordinate among selections being made in a given round since the congestion of any edge can go up by at most 2 in a single round.

After n rounds, the previous algorithm will have connected every pair of live nodes in a star with a live path of length at most 4. Moreover, the congestion is bounded by $O(1)$. This completes the embedding of the cliques of P_N into the faulty H_N. We next show how to embed the hypercube edges of P_N into the faulty H_N.

By Theorem 1, the hypercube edges of P_N are mapped to paths of constant length in H_N. For every pair of neighboring cliques in P_N there are n paths in H_N between the stars corresponding to the cliques (excluding the path between the star centers). For example, clique u and clique u^t are neighbors across hypercube dimension t in P_N. Let $\mathcal{P}(u, u^t)$ be the paths (nodes and edges, including endpoints) in H_N between $g(u)^i$ and $g(u^t)^i$, $1 \leq i \leq n$, given by the embedding. It can be shown that at most three paths in $\mathcal{P}(u, u^t)$ can intersect in a single node of H_N. Hence, these paths collectively use $\Theta(n)$ nodes and edges of H_N. If p is a sufficiently small constant, then with high probability, at most $\epsilon_3 n$ of the nodes or edges in $\mathcal{P}(u, u^t)$ will be faulty for any pair u, u^t and $\epsilon_3 > 0$. Since each faulty node or edge in $\mathcal{P}(u, u^t)$ can destroy at most three of the paths in $\mathcal{P}(u, u^t)$, this means that with high probability, at least $(1 - 3\epsilon_3)n$ of the paths in $\mathcal{P}(u, u^t)$ will be live. Hence, we embed the ith path along the ith live path (counted with wraparound). Provided that $\epsilon_3 \leq 1/6$, this results in at most a factor of 2 increase in congestion.

Of course, we still have to hook up the endpoints of the live paths to the appropriate endpoints in each star. This is easily accomplished in the following manner. If e is a dimension t edge of $H_{2^{n-k}}$ whose path ends at v^j in the star centered at v but whose clique endpoint has been embedded at v^i, we make the connection by routing the path through the path of length at most 4 from v^j to the tth live node of the star and then through the path of length at most 4 from this node to v^i. We use a similar path at the other endpoint. Because each v^i contains at most $O(1)$ clique nodes and each v^j contains at most $O(1)$ path endpoints for e in dimension t, this embedding results in only constant congestion. This completes the proof of Theorem 6. □

4.2. Worst-case faults. In this section we prove the following theorem.

THEOREM 7. *An N-node hypercube with $\log^{O(1)} N$ worst-case faults contains an embedding of a fully functioning N-node hypercube with constant load, dilation, and congestion.*

Proof. Below we will show that, for every constant r, the r-fold hypercube of cliques $P_{N,r}$ with $O(n^r)$ faults contains an embedding of a fault-free $P_{N,r}$ with constant load, dilation, and congestion. This implies Theorem 7 as follows. Let the number of faults that an r-fold hypercube of cliques can tolerate (i.e., still reconfigure to run with constant slowdown) be $a_r n^r$ for some constant $a_r > 0$. Suppose we embed $P_{N,r}$ into an N-node hypercube with F faults. Consider nodes or edges of $P_{N,r}$ that are embedded in a fault of H_N to be faulty. Since one node fault in H_N induces n edge faults in H_N, by Theorem 2 at most bnF nodes or edges of $P_{N,r}$ are faulty for some constant b that is independent of N. If $F \le \frac{a_r n^{r-1}}{b}$, then we can reconfigure $P_{N,r}$ within itself so as to avoid these faults, thereby obtaining an embedding of $P_{N,r}$ into H_N that avoids the faulty components of H_N. Since $P_{N,r}$ contains H_N as a subgraph, this means that an N-node hypercube can be reconfigured around $\frac{a_r n^{r-1}}{b}$ faults with $O(1)$ load, dilation, and congestion for any constant r. Hence, the N-node hypercube can tolerate $\log^{O(1)} N$ faults, as claimed.

For simplicity we will assume that all the faults in $P_{N,r}$ are node faults. This affects the bounds by only constant factors since for any set E of edge faults in $P_{N,r}$ there is a set F of node faults of size at most $2|E|$ such that the set of edge faults induced by the node faults contains E. We will show that a fault-free $P_{N,r}$ can be embedded into a $P_{N,r}$ with $c_r 2^{sr} = \Theta(n^r)$ faults, for some constant $c_r > 0$ (and s as defined previously), with constant load, dilation, and congestion for a fixed r. Recall that

$$P_{N,r} = H_{N/2^{sr}} \otimes \overbrace{K_{2^s} \otimes \cdots \otimes K_{2^s}}^{r} = H_{N/2^{sr}} \otimes K_{2^s}^{(r)},$$

where for convenience we will denote $K_{2^s}^{(r)}$ as the r-dimensional mesh of cliques. The key to the reconfiguration of $P_{N,r}$ is the reconfiguration of the r-dimensional mesh of cliques $K_{2^s}^{(r)}$ around $c_r 2^{sr}$ faults. This is established in the following lemma. To state the lemma let us define an M-relation on a set of nodes A to be a set of origin-destination pairs such that any node in A is the origin in at most M pairs and the destination in at most M pairs.

LEMMA 6. *Define $c_r = 2^{-(r+1)^2}$ for $r \ge 1$, and set $d_1 = \frac{15}{16}$ and $d_r = \left(1 - \frac{c_r}{c_{r-1}}\right) d_{r-1}$ for $r > 1$. Then if $K_M^{(r)}$ has at most $c_r M^r$ worst-case node faults, there exists a set Γ of $d_r M^r$ nonfaulty nodes in $K_M^{(r)}$ for which the paths for any M-relation on Γ can be routed in a fault-avoiding fashion through $K_M^{(r)}$ using constant dilation and congestion (where the constant depends on r but not on M). Moreover, there exists a fault-free embedding of $K_M^{(r)}$ in the faulty graph with constant load, dilation, and congestion (where the constant depends on r but not on M) for which every node of the fault-free $K_M^{(r)}$ is mapped to a node in Γ.*

Proof. The proof is by induction on r. The base case when $r = 1$ is easy since $K_M^{(1)} = K_M$ and $c_1 = \frac{1}{16}$. Then the set Γ will consist of the $\frac{15}{16}M$ fault-free nodes. To route any M-relation on $K_{15M/16}$, we use the well-known fact that any M-relation can be partitioned into M 1-relations [17]. A pair (i, j) in the tth 1-relation is then routed along the path $i \to t \to j$ in $K_{15M/16}$, where t is computed modulo $15M/16$. It is easy to check that these paths have dilation 2 and congestion $O(1)$. We can also use any $M/2$ of the $15M/16$ nodes in Γ to simulate the fault-free K_M with load 2, dilation 1, and congestion 4.

We now assume that the lemma is true for $r - 1$ dimensions, and we consider the case for r dimensions. We start by showing how to find the set Γ.

By definition, $K_M^{(r)} = K_M \otimes K_M^{(r-1)}$. Hence, we can partition $K_M^{(r)}$ into M copies of $K_M^{(r-1)}$. At least $(1 - \frac{c_r}{c_{r-1}})M$ of these copies contain at most $c_{r-1}M^{r-1}$ faults each. (Otherwise, we would have more than $c_r M^r$ faults overall, which is not possible.) Define these copies as the *useful* copies. The useful copies will be used for Γ and for reconfiguring $K_M^{(r)}$.

The set of nodes Γ will be the union of $\Gamma_1, \Gamma_2, \ldots, \Gamma_{(1-c_r/(c_{r-1}))M}$, where Γ_i is the corresponding set of $d_{r-1}M^{r-1}$ nodes in the ith useful copy of $K_M^{(r-1)}$ (which exist by induction). Hence

$$|\Gamma| = d_{r-1}M^{r-1}\left(1 - \frac{c_r}{c_{r-1}}\right)M = d_r M^r$$

as desired.

We next show how to route any M-relation on Γ using constant dilation and congestion. Given any M-relation on Γ, we will first consider the congestion and dilation necessary to route paths to the correct copy of $K_M^{(r-1)}$. If G is the subgraph of $K_M^{(r)}$ restricted to the useful copies of $K_M^{(r-1)}$ and G' is derived from G by viewing each $K_M^{(r-1)}$ as a supernode, then this part of the routing corresponds to an M^r-relation on G'. The M^r-relation can be factored into M^r 1-relations which can then be coalesced into M M^{r-1}-relations. A pair (i, j), i and j in G', in the tth M^{r-1}-relation will be routed in two parts: first from the ith useful copy of $K_M^{(r-1)}$ to the tth useful copy, and then from the tth useful copy to the jth useful copy. Overall, at most $O(M^{r-1})$ paths will have to be routed between any pair of copies of $K_M^{(r-1)}$.

By definition, each useful copy of $K_M^{(r-1)}$ contains $d_{r-1}M^{r-1}$ nodes which can be used for routing any M-relation. By definition, $d_{r-1} \geq 2/3$. Hence, for any pair of useful copies there must be at least $(2d_{r-1} - 1)M^{r-1} > M^{r-1}/3$ of these nodes which are common to both. These nodes will be used to interconnect the two copies. A total of $O(M^{r-1})$ paths need to be routed between the two copies, and we will use the $M^{r-1}/3$ available edges in an arbitrary fashion. Hence, we can route the paths between each pair of $K_M^{(r-1)}$ with constant dilation and congestion. Thus, all the paths for any M-relation on Γ can be routed to the correct copy of $K_M^{(r-1)}$ with constant dilation and congestion. The remainder of the routing is handled within each useful copy of $K_M^{(r-1)}$ by induction.

We next show how to reconfigure $K_M^{(r)}$ within Γ. Since $c_r < \frac{c_{r-1}}{2}$, there will be at least $\frac{M}{2}$ useful copies of $K_M^{(r-1)}$, each of which (by induction) can be reconfigured to embed two fault-free copies of $K_M^{(r-1)}$ (within the appropriate subsets of Γ) with constant load, dilation, and congestion. It remains to hook up these M copies of $K_M^{(r-1)}$ to each other to produce a single copy of $K_M^{(r)} = K_M \otimes K_M^{(r-1)}$.

To hook up any pair of copies of $K_M^{(r-1)}$, we need to route an edge connecting the ith node of one copy to the ith node of the other copy for each i, $1 \leq i \leq M^{r-1}$. These edges are routed using the same method as before. In particular, we know that there are at least $M^{r-1}/3$ edges connecting each pair of copies with endpoints in Γ. These edges will be used to interconnect a pair of copies with congestion 3. The connection is completed by routing a 3-relation in each copy.

To make connections between all pairs, we will load the edges between the useful copies of $K_M^{(r-1)}$ by a factor of at most 12 (4 pairs, each with load 3), and we will need to embed a $6M$-relation in each copy (4 pairs for each of $\frac{M}{2}$ faulty copies, each needing a 3-relation). By the inductive hypothesis, this can be accomplished with constant dilation and congestion. This concludes the proof that $K_M^{(r)}$ can be reconfigured around $c_r M^r$ faults. □

To complete the reconfiguration of $P_{N,r}$ we need to embed the hypercube edges of $P_{N,r}$ into the faulty $P_{N,r}$. This is shown with an argument similar to that used in Lemma 6 for routing edges between two copies of $K_{2^s}^{(r-1)}$ in $K_{2^s}^{(r)}$. By the lemma, each mesh of cliques of the faulty $P_{N,r}$ has a set Γ of $d_r 2^{sr}$ nodes through which to route a 2^s-relation. Since $d_r > 2/3$, any pair of meshes of cliques, $\mathcal{M}$ and $\mathcal{N}$, which are neighbors across a hypercube dimension must share at least 1/3 of these nodes. A total of 2^{sr} edges will be routed between $\mathcal{M}$ and $\mathcal{N}$ using the $2^{sr}/3$ available edges. The connection between the ith node of $\mathcal{M}$ and the ith node of $\mathcal{N}$, for each i, is completed by routing a 3-relation in each mesh of cliques. This embedding yields only constant congestion on the hypercube edges of the faulty $P_{N,r}$. Since each hypercube dimension requires only the set Γ of each mesh of cliques to route an $O(1)$-relation and there are $n - rs = \Theta(2^s)$ dimensions, this routing adds another $O(2^s)$-relation for each Γ of each mesh of cliques to route. Hence, the additional load and congestion on the nodes and edges, respectively, of the meshes of cliques in the faulty $P_{N,r}$ is increased by at most a constant factor. □

4.3. Remarks. We have shown how to embed $P_{N,r}$ into H_N with load 1, dilation $O(1)$, and congestion $O(1)$ for any constant $r \geq 1$, and we have demonstrated several applications of this embedding. Subsequent to [1], Kaklamanis, Krizanc and Rao have used the result to show that the hypercube can emulate any bounded degree planar graph with constant slowdown [15]. In addition the constants in the embedding have been improved by finding Hamming codes with smaller height and width: first in [17], then in the present paper, then in [21] in which a height 3, width 5 code is claimed. Determining whether a height 3 Hamming code with width less than 5 can be constructed is an interesting open problem.

Characterizing the height and width of other linear codes may yield other applications for hypercube routing, embedding, and emulation algorithms. Such characterizations may also be of interest in their own right. Note that the height and width are not parameters of the codewords themselves but rather of the particular mapping of input strings to codewords. The height is the maximum distance between codewords whose inverses differ by one bit. The width is equivalent to the number of input bits needed to compute one bit of the codeword. Hence, linear codes with n-bit codewords and width w can be computed by bounded degree circuits of depth $O(\log w)$ and size $O(nw)$.

Appendix A. Constructing a good basis for the codewords.

Proof of Lemma 3. Finding a basis with the above bounds for C_1 and C_2 is trivial. Henceforth, assume that $k \geq 3$. The basic approach is to take a well-defined basis of height 3 but large width and manipulate it to get a basis with height 6 and width 9. We will first need some definitions.

Let c_i denote the ith column of C_k. For concreteness assume that the columns of C_k are fixed as follows: the first k columns, $c_1, \ldots, c_k$, are all the strings of length k with weight 1; the next $\binom{k}{2}$ columns are all the strings of weight 2; and so on. So, e.g.,

$$C_3 = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

Suppose a vector in the nullspace is nonzero in bit positions $i_1, \ldots, i_h$, then columns $c_{i_1}, \ldots, c_{i_h}$ when added up in $GF(2)^k$ yield the all zero vector of length k. Conversely, any subset of columns $c_{i_1}, \ldots, c_{i_h}$ which add up to zero correspond to a vector in the nullspace of C_k. Henceforth, we will assume that vectors in the nullspace of C_k and subsets of columns of C_k which add up to zero are synonymous.

Let S be the set of all vectors of weight 3 which are in the nullspace of C_k. An easy counting argument shows that there are $\frac{1}{3}\binom{n}{2}$ such vectors, where $n \equiv 2^k - 1$. Let $S(i)$ be all the vectors in S which have a 1 in the ith dimension. Observe that the size of $S(i)$ is $(n-1)/2$ for all i.

Our starting basis B will be composed of $n-k$ vectors from S. Obviously this basis will have height 3. However, the width may be unbounded. The basis is constructed as follows. For each column c_i in C_k of weight 2 or more we will find two more columns which when added to c_i yield zero. This will obviously give us $n-k$ vectors provided that no triple of columns is chosen more than once. To do this, we need the following definition. Recall that for a string s, $s(i)$ denotes the ith bit of s. A k bit string s *contains* another k bit string s' if $s(i) \geq s'(i)$ for all $i \in [1, k]$. For each column c of weight $l \geq 2$, we will choose one of the l columns of weight $l-1$ which are contained by c. Call this c'. Obviously, there will be a unique vector of weight 1 which when added to c and c' will yield zero. We will pick c' carefully among the l choices as follows. It is well known that there is a surjection, f, from strings of weight l to strings of weight $l-1$ such that s contains $f(s)$ for $1 \leq l \leq \lceil k/2 \rceil$ [8]. There is also an injection, g, from strings of length l to strings of length $l-1$ such that s contains $g(s)$ for $\lfloor k/2 \rfloor + 1 \leq l \leq k$. For $2 \leq l \leq \lceil k/2 \rceil$, let $c' = f(c)$. For $\lceil k/2 \rceil + 1 \leq l \leq k$, let $c' = g(c)$.

B as defined above contains $n-k$ vectors. Since this is the dimension of the nullspace, to show that B is a basis, we need show only that it spans.

LEMMA 7. *B spans the nullspace of C_k.*

Proof. Consider a nonzero vector y, and let $\sigma(y)$ be the largest nonzero dimension of y, i.e., $\sigma(y) = i$ when $y(i) = 1$ and $y(j) = 0$ for $j > i$. We will prove by induction on l that all vectors in the nullspace with $k \leq \sigma \leq l$ are spanned by B. The base case is trivial since there are no vectors with $\sigma = k$ in the nullspace of C_k. This is because the columns 1 through k of C_k are a permutation of the k by k identity matrix and so no subset of them can add up to zero.

Suppose by induction that all vectors in the nullspace with $k \leq \sigma \leq l$ are spanned by B. Let y be a vector in the nullspace with $\sigma(y) = l+1$. Let v be the unique vector in B with $\sigma(v) = l+1$. The vector $y' = y \oplus v$ is in the nullspace since y and v are. Furthermore, $\sigma(y') \leq l$. Hence, by the induction hypothesis, y' is spanned by B. Since $y = y' \oplus v$, y is spanned by B as well. □

Before we begin manipulating B, let us first show that B is not too bad to begin with. Define $W(i)$ to be the set of basis vectors with a 1 in dimension i, and define $w(i) = |W(i)|$ to be the width of dimension i. Define D_q as the set of dimensions i with $w(i) = q$ and $D_{>q} \equiv D_{\geq q+1}$ as the set of dimensions i with $w(i) > q$. Define $M_{>q} \equiv M_{\geq q+1}$ as

$$\sum_{i \in D_{>q}} w(i) - q.$$

This is the mass of 1's causing dimensions to have width more than q.

LEMMA 8. *The basis B given above has $M_{\geq 3} \leq (3/2)n - 3k$ for $k \geq 3$.*

Proof. First observe that the total number of 1's in B is $3(n-k)$ since it is composed of $n-k$ vectors each of weight 3. It follows that $|D_1| + 2|D_{\geq 2}| + M_{\geq 3} = 3(n-k)$. Every dimension of B has width at least 1 so that $|D_1| + |D_{\geq 2}| = n$. Hence, $M_{\geq 3} = n - 3k + |D_1|$. To bound $|D_1|$ recall that f is a surjection and g is an injection. This implies that the highest $\sum_{i=\lceil k/2 \rceil}^{k} \binom{k}{i}$ dimensions have width 1 or 2 (and only these dimensions can have width 1). It further implies that, of these, $\sum_{i=\lceil k/2 \rceil+1}^{k} \binom{k}{i}$ have width 2. So, $|D_1| = \binom{k}{\lceil k/2 \rceil}$ which is at most $n/2$ for $k \geq 3$. □

Now we will start the process of manipulating B to achieve bounded width. Our main tool for doing this is the lemma below, for which we will need the following definition. Given a basis $B = \{v_1, \ldots, v_{n-k}\}$ and vectors z and z_B such that $z = B \cdot z_B$, we say that v_j *contributes* to z if $z_B(j) = 1$; otherwise, we say that it does not contribute.

LEMMA 9. *Given any z, any v in B which contributes to z, and any other basis vector x, the following holds. If x does not contribute to z, then $B' = (B - \{v, x\}) \cup \{z, z \oplus x\}$ is a new basis. If x contributes to z, then $B' = (B - \{v, x\}) \cup \{z \oplus v, z\}$ is a new basis.*

Proof. Let us examine the first case. Since B' has $n - k$ vectors, we need argue only that B' spans. To do this we need show only how to produce v and x from B'. We can produce x from B' simply by adding z and $z \oplus x$. To produce v, note that by assumption $z = v \oplus a$, where a is in the span of $B - \{v, x\}$. Hence, v is produced from B' simply by adding the basis vector z and the vector a.

The proof for the second case follows identical reasoning. □

We will use Lemma 9 in the following way. Suppose i is a dimension with large width, i.e., $W(i)$ is a large set. Let z be any vector in $S(i)$. Since B spans, there must be at least one vector in B which contributes to z and is also in $W(i)$ (otherwise z could not have a 1 in dimension i). Call this vector v. Let x be any other vector in $W(i)$. If we use the above lemma, then the width of dimension i of B' is $w(i) - 1$. Of course, in the process we have increased the width of two other dimensions (since the weight of z was 3). These must be dimensions which had small width. We will ensure that this is the case as follows.

Let $D_{\geq g+2}$ be the *bad* dimensions where g will be determined below. Let the dimensions with width less than g be the *good* dimensions. Below we will show that for sufficiently large g, if i is a bad dimension, there is a $z \in S(i)$ such that z's other two 1's are in good dimensions (say j and k). Now when we apply Lemma 9, since dimensions j and k increase their width by at most 2, they cannot become bad dimensions.

Suppose we have just one more application of Lemma 9 left to make. That is, there is one dimension, i, with width $g+2$, and the remainder of the dimensions have width at most $g+1$. Recall that D_{g+1} is the set of dimensions which have width exactly $g+1$ and that the size of $S(i)$ is $(n-1)/2$. Moreover, for any dimension other than i there is a unique vector in $S(i)$ with a 1 in that dimension. In order to be able to guarantee that we can pick a z in $S(i)$ with its other two 1's in good dimensions, we need

$$\frac{n-1}{2} - |D_{g+1}| - |D_g| \geq 1$$

(every dimension with width g or $g+1$ may eliminate one vector from $S(i)$).

To check that we can satisfy the above inequality we need to calculate an upper bound on the size of $D_g \cup D_{g+1}$. This is maximized when the dimensions not in $D_g \cup D_{g+1}$ (other than dimension i which has width $g+2$) have minimum width, i.e., width 1. That is,

$$(g+2) + |D_{g+1}|(g+1) + |D_g|g + n - (|D_{g+1}| + |D_g| + 1) \leq T,$$

where T is the total number of 1's in the basis before the last step.

T is bounded by the number of 1's at the start $(3(n-k))$ plus the number of 1's added to the basis before the last step. To bound this latter quantity first note that we add at most 1 to our basis with each application of Lemma 9. This follows from the fact that z has weight 3, $z \oplus x$ has weight at most the weight of x plus 1, $z \oplus v$ has weight at most the weight of v plus 1, and the weights of x and v are at least 3 since they are basis vectors. The number of applications of Lemma 9 is bounded by $M_{\geq g+2}-1$, where $M_{\geq g+2}$ is the mass of 1's contributing to width $\geq g+2$ in our original basis. But $M_{\geq g+2} \leq M_{\geq 3} \leq (3/2)n-3k$. Putting this all together we find

$$|D_g|+|D_{g+1}| \leq \frac{\frac{7}{2}n-6k-g-2}{g-1}.$$

Substituting this into $\frac{n-1}{2}-|D_{g+1}|-|D_g| \geq 1$ and doing some algebra, we find that we require

$$(g-8)n+12k+7-g \geq 0.$$

Hence, $g=8$ suffices which bounds the width of our final basis by 9.

We have achieved a basis with the desired width, but let us check that we have not unduly increased the height. When we eliminated a 1 in a vector in a bad dimension, we added at most two 1's in good dimensions. Since there were at most three 1's in bad dimensions at the start, the resulting vector has weight at most 6.

The algorithm for constructing the original B is polynomial in n (f and g are efficiently constructable), and the manipulations to reduce the width of B require only polynomial in n time. This completes the proof of Lemma 3. □

An alternate construction. In what follows, we will describe a relationship between the well-known cyclic codes and good bases. This will lead to a good basis for C_k for k up to 137. This in turn gives us an alternative method for obtaining good embeddings for hypercubes up to size $2^{2^{137}}$.

Let p be a primitive polynomial of degree k in $GF(2)[x]$. By definition, x^i mod p, $0 \leq i \leq 2^k-1$, generates all polynomials of degree $k-1$ or less except for the zero polynomial. For convenience, in this section we will assume that the dimensions (bit positions) of a vector (string) of length n are labeled from 0 to $n-1$. There is a one-to-one correspondence between all polynomials of degree $k-1$ or less and strings of length k when the strings are regarded as a sequence of coefficients. Let the weight of a polynomial be the weight of its associated string.

LEMMA 10. *If there exists a primitive polynomial of degree k in $GF(2)[x]$ of weight s, then there is a basis for the nullspace of C_k of height and width s.*

Proof. Let column i, c_i, of C_k be x^i reduced modulo p for $0 \leq i \leq 2^k-1$. Now suppose for simplicity that p has weight 3: $p = x^k + x^j + 1$, where $k > j$. It will be clear how the proof generalizes to primitive polynomials of arbitrary weight. Define v_l to be a vector of length 2^k which is zero everywhere except for a 1 in the lth position, a 1 in the $(l+j)$th position, and a 1 in the $l+k$th position. Let $\mathcal{V}$ be the set of all v_l with $0 \leq l \leq n-k-1$. This is the well-known cyclic code.

CLAIM 1. *$\mathcal{V}$ is in the nullspace of C_k.*

Proof. $C_k \cdot v_l$ is just the sum $c_l \oplus c_{l+j} \oplus c_{l+k}$, where addition is in $GF(2)^k$. By definition of c_i this is equivalent to

$$x^l(\text{mod} p) + x^{l+j}(\text{mod} p) + x^{l+k}(\text{mod} p),$$

where addition is in $GF(2)[x]$. But this is just $x^l(1+x^j+x^k) \pmod p$ which is 0. □

CLAIM 2. *$\mathcal{V}$ is a basis for the nullspace of C_k.*

Proof. Since $\mathcal{V}$ has the correct cardinality, it is enough to show that it spans. The proof that $\mathcal{V}$ spans is nearly identical to the proof of Lemma 7 that B spans and so we omit it. □

To finish the proof of the lemma observe that the number of 1's in each $v_l \in \mathcal{V}$ is at most s, the weight of the primitive polynomial, and that the number of v_l's with a 1 in a given dimension is at most s as well. □

It is not known whether there is a c such that for all degrees there is a primitive polynomial of weight at most c. Nonetheless, small primitive polynomials of degree k in $GF(2)[x]$ have been tabulated for k up to 137 [23]. For each of these degrees there is a polynomial of weight 3 or 5.

Appendix B. Proof of the tree compression lemma.

Proof of Lemma 4. The proof makes use of the following two simple lemmas. The first is a well-known result which first appeared in [19].

Lemma 11. *Given any M-node binary tree T, it is possible to partition T into two subtrees, each with at least $\frac{M-1}{3}$ nodes, by removing a single edge from T.*

Lemma 12. *Let T be a weighted M-node binary tree for which $d_i + w_i \leq 3$ and $w_i \in \{0, 1, 2\}$ for each $i(1 \leq i \leq M)$, where d_i and w_i are the degree and weight, respectively, of the ith node in T. Then if $\sum_{i=1}^{M} w_i = 4$, it is possible to partition T into two subtrees each having total weight 2 by removing a single edge from T.*

Proof. If T contains a node with weight 2, then this node has degree 1, and we can form the desired partition by removing the edge incident to this node.

If T does not contain a node with weight 2, then it contains precisely four nodes with weight 1. Define T' to be the subtree of T consisting of the simple paths that interconnect the four weight-1 nodes. Since the weight-1 nodes can have degree at most 2 in T, there are only three possibilities for the homeomorphic structure of T', and it is easy to find the desired partition of T for each possibility. □

We are now ready to prove Lemma 4. The partition is constructed by repeatedly applying Lemma 11 to split T into smaller and smaller subtrees. However, we must be careful not to remove too many edges from the same subtree. In order to keep everything balanced, we will apply the following splitting process to subtrees in the partially formed partition.

Let T_1 be an M_1-node subtree in the partially formed partition of T. We will assume for now that T_1 is incident to at most three previously removed edges of T. We start by using Lemma 11 to split T_1 into two subtrees, each having at least $\frac{M_1-1}{3}$ nodes. Since T_1 is incident to at most three removed edges, the two newly formed subtrees are incident to two and three removed edges, respectively, or to one and four removed edges, respectively. In the former case, we quit and repeat the entire process on another subtree in the partially formed partition. In the latter case, we apply Lemma 12 to further split the subtree that is incident to four removed edges. As a consequence, we will have split T_1 into three subtrees, each incident to at most three removed edges and at least two of which have at least $\frac{M_1-1}{6}$ nodes.

In order to split T into N subtrees, we repeatedly apply the process just described to the largest remaining subtree until a total of $N-2$ or $N-1$ edges of T have been removed. If $N-2$ edges have been removed, then we apply Lemma 12 once more to a subtree to produce a partition with precisely N subtrees.

By construction, we know that each of the subtrees in the partition is incident to at most three removed edges. In what follows, we show that every subtree has at most $\frac{12M}{N} + 1$ nodes. The proof is not difficult. Each time that we apply the splitting process, we remove at most two edges from T, and we replace the largest current subtree with two or three subtrees, at least two of which have size at least $\frac{M_1-1}{6}$, where M_1 is the size of the subtree being split. Thus, after the splitting process is applied s times, at least $s+1$ of the subtrees will have size at least $\frac{x-1}{6}$, where x is the size of the largest subtree at that point in the algorithm. Since the splitting process is applied at least $\frac{N}{2} - 1$ times during the formation of the partition, this means that the largest subtree in the partition can have size at most x where

$$\frac{N}{2}\frac{(x-1)}{6} \leq M.$$

Hence the largest subtree can have size at most $\frac{12M}{N} + 1$, as claimed. This concludes the proof of Lemma 4. □

REFERENCES

[1] W. Aiello and F. T. Leighton, *Coding theory, hypercube embeddings and fault tolerance*, in Proceedings of the 1991 ACM Symposium on Parallel Algorithms and Architectures, 1991, pp. 125–136.

[2] W. Aiello, F. T. Leighton, B. Maggs, and M. Newman, *Fast algorithms for bit-serial routing on a hypercube*, Math. Syst. Theory, 24 (1991), pp. 253–271.

[3] M. Baumslag, *private communication*, 1990.

[4] B. Becker and H. U. Simon, *How robust is the n-cube?*, Inform. Comput., 77 (1988), pp. 283–291.

[5] S. N. Bhatt and J. Cai, *Taking random walks to grow trees in hypercubes*, J. ACM, 40 (1993), pp. 741–764.

[6] S. N. Bhatt, F. R. K. Chung, F. T. Leighton, and A. L. Rosenberg, *Efficient embeddings of trees in hypercubes*, SIAM J. Comput., 21 (1992), pp. 151–162.

[7] S. N. Bhatt, D. Greenberg, F. T. Leighton, and P. Liu, *Tight bounds for on-line tree embeddings*, SIAM J. Comput., 29 (1999), pp. 474–491.

[8] B. Bollobás, *Combinatorics*, Cambridge University Press, Cambridge, 1986.

[9] J. Bruck, R. Cypher, and D. Soroker, *Tolerating faults in hypercubes using subcube partitioning*, IEEE Trans. Comput., 41 (1992), pp. 599–605.

[10] H. Gazit, G. L. Miller, and S. H. Teng, *Optimal tree contraction in the erew model*, in Concurrent Computations, S. Tewsburg, B. Dickson, and S. Schwartz, eds., Plenum Press, New York, 1988, pp. 139–156.

[11] N. Graham, F. Harary, M. Livingston, and Q. F. Stout, *Subcube fault-tolerance in hypercubes*, Inf. Comput., 102 (1993), pp. 280–314.

[12] D. S. Greenberg and S. N. Bhatt, *Routing multiple paths in hypercubes*, Math. Syst. Theory, 24 (1991), pp. 295–321.

[13] R. W. Hamming, *Error detecting and error correcting codes*, Bell Syst. Tech. J., 29 (1950), pp. 147–160.

[14] J. Hastad, F. T. Leighton, and M. Newman, *Fast computation using faulty hypercubes*, in Proceedings of the 21st Annual ACM Symposium on Theory of Computing, 1989, pp. 251–263.

[15] C. Kaklamanis, D. Krizanc, and S. Rao, *New graph decompositions and fast emulations in hypercubes and butterflies*, in Proceedings of the 1993 ACM Symposium on Parallel Algorithms and Architectures, 1993, pp. 325–334.

[16] R. R. Koch, *Increasing the size of a network by a constant factor can increase performance by more than a constant factor*, SIAM J. Comput., 21 (1992), pp. 801–823.

[17] F. T. Leighton, *An Introduction to Parallel Algorithms and Architectures*, Morgan Kaufmann, San Mateo, CA, 1992.

[18] F. T. Leighton, M. Newman, A. Ranade, and E. Schwabe, *Dynamic tree embeddings in butterflies and hypercubes*, SIAM J. Comput., 21 (1992), pp. 639–654.

[19] P. M. Lewis, R. E. Stearns, and J. Hartmanis, *Memory bounds for recognition of context-free and context-sensitive languages*, in Proceedings of the IEEE Conference on Switching Theory and Logical Design, Washington, D.C., IEEE Computer Society, 1965, pp. 191–202.

[20] D. Peleg and J. Ullman, *An optimal synchronizer for the hypercube*, SIAM J. Comput., 18 (1989), pp. 740–747.

[21] D. Pritikin, *Graph Embeddings from Hamming Bases*, manuscript, 1992.

[22] M. O. Rabin, *Efficient dispersal of information for security, load balancing, and fault tolerance*, J. ACM, 36 (1989), pp. 335–348.

[23] W. Stahnke, *Primitive binary polynomials*, Math. Comput., 27 (1973), pp. 977–980.

SIAM J. COMPUT.
Vol. 37, No. 3, pp. 804–826

PLANAR EARTHMOVER IS NOT IN L_1*

ASSAF NAOR† AND GIDEON SCHECHTMAN‡

Abstract. We show that any L_1 embedding of the transportation cost (a.k.a. Earthmover) metric on probability measures supported on the grid $\{0,1,\dots,n\}^2 \subseteq \mathbb{R}^2$ incurs distortion $\Omega\left(\sqrt{\log n}\right)$. We also use Fourier analytic techniques to construct a simple L_1 embedding of this space which has distortion $O(\log n)$.

Key words. Earthmover metric, nearest neighbor search, metric embeddings

AMS subject classifications. 51F99, 65D18, 46B99

DOI. 10.1137/05064206X

1. Introduction.

For a finite metric space (X, d_X) we denote by $\mathscr{P}_X$ the space of all probability measures on X. The transportation cost distance (also known as the Earthmover distance in the computer vision/graphics literature) between two probability measures $\mu, \nu \in \mathscr{P}_X$ is defined by

$$\tau(\mu,\nu) := \min\left\{ \sum_{x,y\in X} d_X(x,y)\pi(x,y) :\ \forall x,y \in X,\ \pi(x,y) \ge 0, \right.$$
$$\left. \sum_{z\in X} \pi(x,z) = \mu(x),\ \sum_{z\in X}\pi(z,y) = \nu(y) \right\}.$$

Observe that if μ and ν are the uniform probability distribution over k-point subsets $A \subseteq X$ and $B \subseteq X$, respectively, then by the fact that all the extreme points of the $k \times k$ doubly stochastic matrices are permutation matrices,

$$\tau(\mu,\nu) = \min\left\{ \frac{1}{k}\sum_{a\in A} d_X(a, f(a)) :\ f : A \to B \text{ is a bijection} \right\}. \tag{1.1}$$

This quantity is also known as the *minimum weight matching* between A and B, corresponding to the weight function $d_X(\cdot,\cdot)$ (see [42]). Thus, the Earthmover distance is a natural measure of similarity between images [42, 15, 14]—the distance is the optimal way to match various features, where the cost of such a matching corresponds to the sum of the distances between the features that were matched. Indeed, such metrics occur in various contexts in computer science. Apart from being a popular distance measure in graphics and vision [42, 15, 14, 26], they are used as LP relaxations for classification problems such as 0-extension and metric labeling [9, 8, 2]. Transportation cost metrics are also prevalent in several areas of analysis and PDEs (see the book [53] and the references therein).

[*] Received by the editors October 6, 2005; accepted for publication (in revised form) February 13, 2007; published electronically July 18, 2007.
http://www.siam.org/journals/sicomp/37-3/64206.html

[†] Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012 (naor@cims.nyu.edu). This work was carried out while this author was a member of the Theory Group of Microsoft Research.

[‡] Department of Mathematics, Weizmann Institute, Rehovot 76100, Israel (gideon@weizmann.ac.il). This author's research was supported in part by the Israel Science Foundation. This work was carried out while this author was a long-term visitor at the Theory Group of Microsoft Research.

Following extensive work on nearest neighbor search and data stream computations for L_1 metrics (see [24, 20, 19, 10, 22]), it became of great interest to obtain low distortion embeddings of useful metrics into L_1 (here, and in what follows, L_1 denotes the space of all Lebesgue measurable functions $f : [0, 1] \to \mathbb{R}$ such that $\|f\|_1 := \int_0^1 |f(t)|dt < \infty$). Indeed, such embeddings can be used to construct approximate nearest neighbor databases, with an approximation guarantee depending on the *distortion* of the embedding (we are emphasizing here only one aspect of the algorithmic applications of low distortion embeddings into L_1—they are also crucial for the study of various cut problems in graphs, and we refer the reader to [36, 23, 21] for a discussion of these issues).

In the context of the Earthmover distance, nearest neighbor search (a.k.a. similarity search in the vision literature) is of particular importance. It was therefore asked (see, e.g., [35]) whether the Earthmover distance embeds into L_1 with constant distortion (the best known upper bounds on the L_1 distortion were obtained in [8, 26] and will be discussed further below). In [30] the case of the Hamming cube was settled negatively: It is shown there that any embedding of the Earthmover distance on $\{0, 1\}^d$ (equipped with the L_1 metric) incurs distortion $\Omega(d)$. However, the most interesting case is that of the Earthmover distance on the Euclidean plane, as this corresponds to a natural similarity measure between images [14] (indeed, the case of the L_1 embeddability of planar Earthmover distance was explicitly asked about in [35]). Here we settle this problem negatively by obtaining the first superconstant lower bound on the L_1 distortion of the planar Earthmover distance. To state it we first recall some definitions.

Given two metric spaces (X, d_X) and (Y, d_Y), and a mapping $f : X \to Y$, we denote its Lipschitz constant by

$$\|f\|_{\mathrm{Lip}} := \mathrm{supp}_{\substack{x,y\in X\\ x\neq y}} \frac{d_Y(f(x), f(y))}{d_X(x, y)}.$$

If f is one to one, then its distortion is defined as

$$\mathrm{dist}(f) := \|f\|_{\mathrm{Lip}} \cdot \|f^{-1}\|_{\mathrm{Lip}} = \mathrm{supp}_{\substack{x,y\in X\\ x\neq y}} \frac{d_Y(f(x), f(y))}{d_X(x, y)} \cdot \mathrm{supp}_{\substack{x,y\in X\\ x\neq y}} \frac{d_X(x, y)}{d_Y(f(x), f(y))}.$$

The smallest distortion with which X can be embedded into Y is denoted $c_Y(X)$, i.e.,

$$c_Y(X) := \inf \{\mathrm{dist}(f) : \ f : X \hookrightarrow Y \text{ is one to one}\}.$$

When $Y = L_p$ we use the shorter notation $c_Y(X) = c_p(X)$. Thus, the parameter $c_2(X)$ is the Euclidean distortion of X, and $c_1(X)$ is the L_1 distortion of X.

Our main result bounds from below the L_1 distortion of the space of probability measures on the n by n grid, equipped with the transportation cost distance.

THEOREM 1.1. $c_1\left(\mathscr{P}_{\{0,1,\ldots,n\}^2}, \tau\right) = \Omega\left(\sqrt{\log n}\right)$.

We note that the best known upper bound for $c_1\left(\mathscr{P}_{\{0,1,\ldots,n\}^2}, \tau\right)$ is $O(\log n)$, as proved in [8, 26]. Later on (see Theorem 1.4) we will show a new embedding which achieves this bound.

After reducing the problem to a functional analytic question, our proof of Theorem 1.1 is a discretization of a theorem of Kislyakov from 1975 [32]. We attempted to make the presentation self-contained by presenting here appropriate versions of the various functional analytic lemmas that are used in the proof.

For readers who are more interested in the minimum cost matching metric (1.1), we also prove the following lower bound.

Theorem 1.2 (discretization). *For arbitrarily large integers n there is a family $\mathscr{Y}$ of n-point subsets of $\left\{0,1\ldots,O\left(\sqrt{n\log\log n}\right)\right\}^2$, with $|\mathscr{Y}|\le n^{O(\log\log n)}$, such that any L_1 embedding of $\mathscr{Y}$, equipped with the minimum weight matching metric τ, incurs distortion:*

$$\Omega\left(\sqrt{\log\log\log n}\right)=\Omega\left(\sqrt{\log\log\log|\mathscr{Y}|}\right).$$

A metric space (X,d_X) is said to embed into squared L_2, or to be of negative type, if the metric space $\left(X,\sqrt{d_X}\right)$ is isometric to a subset of L_2. Squared L_2 metrics are important in various algorithmic applications since it is possible to efficiently solve certain optimization problems on them using semidefinite programming (see the discussion in [3, 31]). It turns out that planar Earthmover does not embed into any squared L_2 metric (see Remark 3.3 for a more general result).

Theorem 1.3 (nonembeddability into squared L_2). $\lim_{n\to\infty}c_2\left(\mathscr{P}_{\{0,\ldots,n\}^2},\sqrt{\tau}\right)=\infty$.

Motivated by the proof of Theorem 1.1, we also construct simple low-distortion embeddings of the space $\left(\mathscr{P}_{\{0,1,\ldots,n\}^2},\tau\right)$ into L_1. It is convenient to work with probability measures on the torus $\mathbb{Z}_n^2$ instead of the grid $\{0,1,\ldots,n\}^2$. One easily checks that $\{0,\ldots,n\}^2$ embeds with constant distortion into $\mathbb{Z}_{2n}^2$ (see, e.g., Lemma 6.12 in [37]). Every $\mu\in\mathscr{P}_{\mathbb{Z}_n^2}$ can be written in the Fourier basis as

$$\mu=\sum_{(u,v)\in\mathbb{Z}_n^2}\widehat{\mu}(u,v)e_{uv},\tag{1.2}$$

where

$$\forall(a,b),(u,v)\in\mathbb{Z}_n^2,\ e_{uv}(a,b):=e^{\frac{2\pi i(au+bv)}{n}}$$

and

$$\forall(u,v)\in\mathbb{Z}_n^2,\quad\widehat{\mu}(u,v):=\frac{1}{n^2}\sum_{(a,b)\in\mathbb{Z}_n^2}\mu(a,b)e_{uv}(-a,-b).$$

Observe that for $n=2^k+1$, $k\in\mathbb{N}$, the decomposition (1.2) can be computed in time $O\left(n^2\log n\right)$ using the fast Fourier transform [45]. Motivated in part by the results of [40] (see also [5, 41]), we define

$$A\mu=\sum_{(u,v)\in\mathbb{Z}_n^2\setminus\{(0,0)\}}\frac{e^{\frac{2\pi iu}{n}}-1}{\left|e^{\frac{2\pi iu}{n}}-1\right|^2+\left|e^{\frac{2\pi iv}{n}}-1\right|^2}\cdot\widehat{\mu}(u,v)\cdot e_{uv}\tag{1.3}$$

and

$$B\mu=\sum_{(u,v)\in\mathbb{Z}_n^2\setminus\{(0,0)\}}\frac{e^{\frac{2\pi iv}{n}}-1}{\left|e^{\frac{2\pi iu}{n}}-1\right|^2+\left|e^{\frac{2\pi iv}{n}}-1\right|^2}\cdot\widehat{\mu}(u,v)\cdot e_{uv}.\tag{1.4}$$

Theorem 1.4. *The mapping $\mu\mapsto(A\mu,B\mu)$ from $\left(\mathscr{P}_{\mathbb{Z}_n^2},\tau\right)$ to $L_1\left(\mathbb{Z}_n^2\right)\oplus L_1\left(\mathbb{Z}_n^2\right)$ has distortion $O(\log n)$.*

The $O(\log n)$ distortion in Theorem 1.4 matches the best known distortion guarantee proved in [26, 8]. Our embedding has various nice features. First of all, it is a simple closed-form linear mapping into a low dimensional L_1 space, which is based on the computation of the Fourier transform. It is thus very fast to compute, and it is versatile in the sense that it might behave better on images whose Fourier transform is sparse (we do not study this issue here). Thus there is scope to apply the embedding on certain subsets of the frequencies, and this might improve the performance in practice. This is an interesting "applied" question which should be investigated further (see section 5).

2. Preliminaries and notation. For the necessary background on measure theory, we refer the reader to the book [46]; however, in the setting of the present paper, our main results will deal with finitely supported measures, in which case no background and measurabilty assumptions are necessary. We also refer the reader to the book [53] for background on the theory of optimal transportation of measures. Let (X, d_X) be a metric space. We denote by $\mathscr{M}_X$ the space of all Borel measures on X with bounded total variation and by $\mathscr{P}_X \subseteq \mathscr{M}_X$ the set of all Borel *probability* measures on X. We also let $\mathscr{M}_X^+ \subseteq \mathscr{M}_X$ be the space of *nonnegative* measures on X with finite total mass, and we denote by $\mathscr{M}_X^0 \subseteq \mathscr{M}_X$ the space of all measures $\mu \in \mathscr{M}_X$ with $\mu(X) = 0$. Given a measure $\mu \in \mathscr{M}_X$, we can decompose it in a unique way as $\mu = \mu^+ - \mu^-$, where $\mu^+, \mu^- \in \mathscr{M}_X^+$ are disjointly supported. If $\mu, \nu \in \mathscr{M}_X^+$ have the same total mass, i.e., $\mu(X) = \nu(X) < \infty$, then we let $\Pi(\mu, \nu)$ be the space of all *couplings* of μ and ν, i.e., all nonnegative Borel measures π on $X \times X$, such that for every measurable bounded $f : X \to \mathbb{R}$,

$$\int_{X\times X} f(x)d\pi(x,y) = \int_X f(x)d\mu(x) \quad \text{and} \quad \int_{X\times X} f(y)d\pi(x,y) = \int_X f(y)d\nu(y).$$

Observe that in the case of finitely supported measures, this condition translates to the standard formulation, in which we require that the marginals of π are μ and ν, i.e.,

$$\forall x, y \in X,\ \sum_{z\in X} \pi(x,z) = \mu(x) \quad \text{and} \quad \sum_{z\in X} \pi(z,y) = \nu(y).$$

The *transportation cost distance* between μ and ν, denoted here by $\tau(\mu,\nu) = \tau_{(X,d_X)}(\mu,\nu)$ (and also referred to in the literature as the Wasserstein 1 distance, Monge–Kantorovich distance, or the Earthmover distance), is

$$\tau(\mu,\nu) := \inf\left\{\int_{X\times X} d_X(x,y)\, d\pi(x,y) :\ \pi \in \Pi(\mu,\nu)\right\}. \tag{2.1}$$

For $\mu \in \mathscr{M}_X^0$, $\mu^+(X) = \mu^-(X)$, and so we may write $\|\mu\|_\tau := \tau(\mu^+, \mu^-)$. This is easily seen to be a norm on the vector space $\mathscr{M}_{X,\tau}^0 := \left\{\mu \in \mathscr{M}_X^0 :\ \|\mu\|_\tau < \infty\right\}$. It is easy to check that for any two nonnegative measures μ, ν with the same total mass, and for every nonnegative measure σ, we have $\tau(\mu + \sigma,\ \nu + \sigma) = \tau(\mu, \nu)$. It follows that

$$\begin{aligned} \|\mu - \nu\|_\tau &= \tau\left((\mu-\nu)^+, (\mu-\nu)^-\right) \\ &= \tau\left((\mu-\nu)^+ + \min\{\mu,\nu\}, (\mu-\nu)^- + \min\{\mu,\nu\}\right) \\ &= \tau(\mu,\nu). \end{aligned} \tag{2.2}$$

Fix some $x_0 \in X$, and let $\mathrm{Lip}_0(X) = \mathrm{Lip}_{x_0}(X)$ be the linear space of all Lipschitz mappings $f : X \to \mathbb{R}$ with $f(x_0) = 0$, equipped with the norm $\|\cdot\|_{\mathrm{Lip}}$ (i.e., the norm of a function equals its Lipschitz constant). Any $\mu \in \mathscr{M}^0_{X,\tau}$ can be thought of as a bounded linear functional on $\mathrm{Lip}_0(X)$, given by $f \mapsto \int_X f d\mu$. The famous *Kantorovich duality theorem* (see Theorem 1.14 in [53]) implies that $\mathrm{Lip}_0(X)^* = \mathscr{M}^0_{X,\tau}$, in the sense that every bounded linear functional on $\mathrm{Lip}_0(X)$ is obtained in this way, and for every $\mu \in \mathscr{M}^0_{X,\tau}$,

$$\|\mu\|_\tau = \|\mu\|_{\mathrm{Lip}_0(X)^*} := \mathrm{supp}\left\{\int_X f d\mu : \ f \in \mathrm{Lip}_0(X), \ \|f\|_{\mathrm{Lip}} \le 1\right\}.$$

(We note that this identity amounts to duality of linear programming.)

3. Proof of Theorem 1.1. Fix an integer $n \ge 2$, and denote $X = \{0, 1, \dots, n-1\}^2$, equipped with the standard Euclidean metric. In what follows, for concreteness, $\mathrm{Lip}_0 := \mathrm{Lip}_0(X)$ is defined using the base point $x_0 = (0, 0)$. Also, for ease of notation we denote $\mathscr{M} = \mathscr{M}^0_{X,\tau}$. Observe that Lip_0 and $\mathscr{M}$ are vector spaces of dimension $n^2 - 1$, and by Kantorovich duality, $\mathrm{Lip}_0^* = \mathscr{M}$ and $\mathscr{M}^* = \mathrm{Lip}_0$.

Assume that $F : \mathscr{P}_X \to L_1$ is a bi-Lipschitz embedding satisfying, for any two probability measures $\mu, \nu \in \mathscr{P}_X$,

$$\tau(\mu, \nu) \le \|F(\mu) - F(\nu)\|_1 \le L \cdot \tau(\mu, \nu). \tag{3.1}$$

Our goal is to bound L from below. We begin by reducing the problem to the case of *linear mappings*. Recall that given two normed spaces $(Z, \|\cdot\|_Z)$ and $(W, \|\cdot\|_W)$, the norm of a linear mapping $T : Z \to W$ is defined as $\|T\| = \mathrm{supp}_{z \in Z\setminus\{0\}} \frac{\|Tz\|_W}{\|z\|_Z}$ (observe that in this case $\|T\| = \|T\|_{\mathrm{Lip}}$).

Lemma 3.1 (reduction to a linear embedding of $\mathscr{M}$ into ℓ_1^N). *Under the assumption of an existence of an embedding $F : \mathscr{P}_X \to L_1$ satisfying* (3.1)*, there exists an integer N, and an invertible linear operator $T : \mathscr{M} \to \ell_1^N$, with $\|T\| \le 2L$ and $\|T(\mu)\|_1 \ge \|\mu\|_\tau$ for all $\mu \in \mathscr{M}$ (the factor 2 can be replaced by $1 + \epsilon$ for every $\epsilon > 0$, but this is irrelevant for us here).*

Proof. By translation we may assume that F maps the uniform measure on X to 0. For $\mu \in \mathscr{M}$ denote $\|\mu\|_\infty := \max_{x\in X} |\mu(x)|$. Observe that it is always the case that $\|\mu\|_\infty \le \|\mu\|_\tau$. Indeed, if $\pi \in \Pi(\mu^+, \mu^-)$, then

$$\int_{X\times X} \|x - y\|_2 d\pi(x, y) \ge \int_{X\times X} d\pi(x, y) = \mu^+(X) = \mu^-(X) \ge \|\mu\|_\infty.$$

Let B_{M} denote the unit ball of $\mathscr{M}$. Define for $\mu \in B_{\mathrm{M}}$ a probability measure $\psi(\mu) \in \mathscr{P}_X$ by $\psi(\mu)(x) := \frac{\mu(x)+1}{n^2}$. It is clear that for every $\mu, \nu \in \mathscr{M}$, $\|\mu - \nu\|_\tau = n^2\|\psi(\mu) - \psi(\nu)\|_\tau$. The mapping $h := \frac{1}{n^2} \cdot F \circ \psi : B_{\mathrm{M}} \to L_1$ satisfies $h(0) = 0$, $\|h\|_{\mathrm{Lip}} \le L$, and $\|h(\mu) - h(\nu)\|_1 \ge \|\mu - \nu\|_\tau$, where we have used (3.1) and (2.2). This implies that there exists a map $\tilde{h} : \mathscr{M} \to L_1$ satisfying the same inequalities. We shall present two arguments establishing this fact: The first is a soft nonconstructive proof, using the notion of ultraproducts, and the second argument is more elementary but does not preserve the Lipschitz constant.

Let $\mathscr{U}$ be a free ultrafilter on $\mathbb{N}$, and denote by $(L_1)_{\mathrm{U}}$ the corresponding ultrapower of L_1. (See [16] for the necessary background on ultrapowers of Banach spaces. In particular, it is shown there that $(L_1)_{\mathrm{U}}$ is isometric to an $L_1(\sigma)$ space for some measure σ.) Define for $\mu \in \mathscr{M}$, $\tilde{h}(\mu) = (j \cdot h(\mu/j))_{j=1}^\infty / \mathscr{U}$, where we set, say, $h(\nu) = 0$ for $\nu \in \mathscr{M} \setminus B_{\mathrm{M}}$. Then, by standard arguments, $\|\tilde{h}\|_{\mathrm{Lip}} \le L$ and $\|\tilde{h}^{-1}\|_{\mathrm{Lip}} \le 1$. Moreover, $\tilde{h}(\mathscr{M})$ spans a separable subspace of $(L_1)_{\mathrm{U}}$, and thus we may assume without loss of generality that $\tilde{h}$ takes values in L_1.

An alternative proof (for those of us who do not mind losing a constant factor) proceeds as follows. For every $f \in L_1$ let $\chi(f) : [0,1]\times\mathbb{R} \to \{-1,0,1\}$ be the function given by

$$\chi(f)(s,t) = \operatorname{sign}(f(s))\cdot \mathbf{1}_{\left[0,|f(s)|\right]}(t) = \begin{cases} 1, & f(s)>0,\ 0\le t\le f(s),\\ -1, & f(s)<0,\ 0\le t\le -f(s),\\ 0 & \text{otherwise.}\end{cases}$$

It is straightforward to check that $\|\chi(f)-\chi(g)\|_{L_1([0,1]\times\mathbb{R})} = \|f-g\|_1$ for every $f,g\in L_1$. (We note here that the space $L_1([0,1]\times\mathbb{R})$ is isometric to L_1.) Define $\tilde{h} : \mathscr{M} \to L_1([0,1]\times\mathbb{R})$ by setting $\tilde{h}(\mu) = \|\mu\|_\tau \cdot \chi\circ h(\mu/\|\mu\|_\tau)$ for $\mu\in\mathscr{M}\setminus\{0\}$, and $\tilde{h}(0)=0$. Since for every $f\in L_1$, $\chi(f)$ takes values in $\{-1,0,1\}$, we have the following pointwise identity for every $\mu,\nu\in\mathscr{M}$ with $\|\mu\|_\tau\ge\|\nu\|_\tau$:

$$\begin{aligned}\left|\tilde{h}(\mu)-\tilde{h}(\nu)\right| &= \|\nu\|_\tau\cdot\left|\chi\circ h\left(\frac{\mu}{\|\mu\|_\tau}\right)-\chi\circ h\left(\frac{\nu}{\|\nu\|_\tau}\right)\right|\\ &\quad+(\|\mu\|_\tau-\|\nu\|_\tau)\cdot\left|\chi\circ h\left(\frac{\mu}{\|\mu\|_\tau}\right)\right|.\end{aligned}$$

Thus

$$\begin{aligned}\left\|\tilde{h}(\mu)-\tilde{h}(\nu)\right\|_{L_1([0,1]\times\mathbb{R})} &= \|\nu\|_\tau\cdot\left\|h\left(\frac{\mu}{\|\mu\|_\tau}\right)-h\left(\frac{\nu}{\|\nu\|_\tau}\right)\right\|_1\\ &\quad+(\|\mu\|_\tau-\|\nu\|_\tau)\cdot\left\|h\left(\frac{\mu}{\|\mu\|_\tau}\right)\right\|_1\\ &\ge \|\nu\|_\tau\cdot\left\|\frac{\mu}{\|\mu\|_\tau}-\frac{\nu}{\|\nu\|_\tau}\right\|_\tau+\|\mu\|_\tau-\|\nu\|_\tau\\ &\ge \|\nu-\mu\|_\tau-\left\|\mu-\frac{\|\nu\|_\tau}{\|\mu\|_\tau}\mu\right\|_\tau+\|\mu\|_\tau-\|\nu\|_\tau\\ &= \|\nu-\mu\|_\tau.\end{aligned}\tag{3.2}$$

It also follows from the identity (3.2) that

$$\begin{aligned}\left\|\tilde{h}(\mu)-\tilde{h}(\nu)\right\|_{L_1([0,1]\times\mathbb{R})} &\le L\|\nu\|_\tau\cdot\left\|\frac{\mu}{\|\mu\|_\tau}-\frac{\nu}{\|\nu\|_\tau}\right\|_\tau+L\|\mu-\nu\|_\tau\\ &\le L\|\mu-\nu\|_\tau+L\|\nu\|_\tau\|\mu\|_\tau\cdot\left|\frac{1}{\|\mu\|_\tau}-\frac{1}{\|\nu\|_\tau}\right|+L\|\mu-\nu\|_\tau\\ &\le 3L\|\mu-\nu\|_\tau.\end{aligned}$$

We are now in position to use a theorem of Ribe [44] (see also [17], and Corollary 7.10 in [4], for softer proofs), which implies that there is an into linear isomorphism $S:\mathscr{M}\to L_1^{**}$ satisfying $\|S\|\le L$ and $\|S^{-1}\|\le 1$. Since $\mathscr{M}$ is finite dimensional, by the principle of local reflexivity [33] (alternatively by Kakutani's representation theorem [27, 34]), and a simple approximation argument, we get that there exists an integer N and an into linear isomorphism $T:\mathscr{M}\to\ell_1^N$ satisfying $\|T\|\le 2L$ and $\|T^{-1}\|\le 1$. (The value of N is irrelevant for us here, and indeed it is possible to conclude the proof without passing to a finite dimensional L_1 space, but this slightly simplifies some of the ensuing arguments. For completeness we note here that using a theorem of Talagrand [50] we can ensure that $N=O(n^2\log n)$.) □

Remark 3.1. The first argument above is not special for L_1 and can be generalized to show that for every finite dimensional Banach space Y, if the unit ball of Y admits a bi-Lipschitz embedding into a Banach space Z, then Y embeds linearly with the same distortion into Z.

From now on let $T : \mathscr{M} \to \ell_1^N$ be the linear operator guaranteed by Lemma 3.1. Since T is an isomorphism, the adjoint operator $T^* : \ell_\infty^N \to \mathscr{M}^* = \mathrm{Lip}_0$ is a quotient mapping, i.e., $\|T^*\| \le 2L$, and the image of the unit ball of ℓ_∞^N under T^* contains the unit ball of Lip_0.

The rest of the proof follows that of Kislyakov [32] and is a discretization of his argument. The idea is to compose T^* with a map $\mathscr{F}$ which is the imaginary part of the discrete two dimensional Fourier transform (see the exact definition below), seen as a map from Lip_0 to $\ell_2(X)$, and to prove two properties of the composed map: Using the fact that $\|T^*\| \le 2L$, we shall show that $\mathscr{F} \circ T^*$ is *order bounded* with good bound, that is,

$$\mathscr{F}\left(T^*\left(B_{\ell_\infty^N}\right)\right) \subseteq \{y \in \ell_2(X) : \ |y| \le x\},$$

for some $x \in \ell_2(X)$ such that $\|x\|_2 \le 4Ln$. Then, using the quotient property of T^*, we find a family of functions $\{\phi_i \in B_{\ell_\infty^N}\}_{i\in I}$ such that if $\mathscr{F}(T^*(\phi_i))) \le x$ for all $i \in I$, then necessarily $\|x\|_2 \ge cn\sqrt{\log n}$ for some universal $c > 0$.

We now define two more auxiliary linear operators. The first is the formal identity $\mathrm{Id} : \mathrm{Lip}_0 \to W$, where W is the space of all functions $f : X \to \mathbb{R}$ with $f(0) = 0$, equipped with the (discrete Sobolev) norm

$$\begin{aligned}\|f\|_W := \sum_{i=0}^{n-1}\sum_{j=0}^{n-2} |f(i,j+1) - f(i,j)| + \sum_{j=0}^{n-1}\sum_{i=0}^{n-2} |f(i+1,j) - f(i,j)| \\ + n\sum_{i=0}^{n-2} |f(i+1,0) - f(i,0)| + n\sum_{j=0}^{n-2} |f(0,j+1) - f(0,j)|.\end{aligned}$$

The second operator is also a formal identity (discrete Sobolev embedding) $S : W \to \ell_2(X)$, where the Euclidean norm on $\ell_2(X)$ is taken with respect to the counting measure on X. The final operator that we will use is the imaginary part of the Fourier operator, already referred to above, which we denote by $\mathscr{F} : \ell_2(X) \to \ell_2(X)$. It is defined for $f : X \to \mathbb{R}$ by

$$\mathscr{F}(f)(u,v) := \Im\left(\frac{1}{n^2}\sum_{(k,\ell)\in X} f(k,\ell)e^{\frac{2\pi i(uk+v\ell)}{n}}\right) = \frac{1}{n^2}\sum_{(k,\ell)\in X} f(k,\ell)\sin\left(\frac{2\pi(uk+v\ell)}{n}\right).$$

The following lemma summarizes known estimates on the norms of these operators.

Lemma 3.2 (operator norm bounds). *The following operator norm bounds hold true:*

- $\|\mathrm{Id}\| \le 4n(n-1)$.
- $\|S\| \le \frac{1}{2}$.
- $\|\mathscr{F}\| \le \frac{1}{n}$.

Proof. The first statement means that for every $f : X \to \mathbb{R}$ with $f(0) = 0$, $\|f\|_W \le 4n(n-1)\|f\|_{\mathrm{Lip}}$, which is obvious from the definitions. The second assertion is that $\|f\|_2 \le \frac{1}{2}\|f\|_W$. This is a discrete version of Sobolev's inequality [41] (with nonoptimal constant), which can be proved as follows. First of all, since $f(0) = 0$, for every $(u, v) \in X$,

$$\begin{aligned}|f(u,v)| &= \left|\sum_{k=0}^{u-1}[f(k+1,v)-f(k,v)] + \sum_{\ell=0}^{v-1}[f(0,\ell+1)-f(0,\ell)]\right| \\ &\le \sum_{k=0}^{n-2}|f(k+1,v)-f(k,v)| + \sum_{\ell=0}^{n-2}|f(0,\ell+1)-f(0,\ell)| := A(v). \qquad (3.3)\end{aligned}$$

Analogously,

$$|f(u,v)| \le \sum_{\ell=0}^{n-2}|f(u,\ell+1)-f(u,\ell)| + \sum_{k=0}^{n-2}|f(k+1,0)-f(k,0)| := B(u). \qquad (3.4)$$

Multiplying (3.3) and (3.4), and summing over X, we see that

$$\begin{aligned}\|f\|_2^2 &\le \sum_{(u,v)\in X} A(v)B(u) = \left(\sum_{v=0}^{n-1}A(v)\right)\cdot\left(\sum_{u=0}^{n-1}B(u)\right) \\ &\le \frac{1}{4}\left(\sum_{v=0}^{n-1}A(v) + \sum_{u=0}^{n-1}B(u)\right)^2 = \frac{1}{4}\|f\|_W^2.\end{aligned}$$

The final assertion follows since the system of functions $\{(k,\ell) \mapsto e^{\frac{2\pi i(uk+v\ell)}{n}}\}_{(u,v)\in X}$ are orthogonal in $\ell_2^{\mathbb{C}}(X)$ (the space of complex-valued functions on X) and have norms bounded by n. □

We now recall some facts related to absolutely summing operators on Banach spaces (we refer the interested reader to [51, 54] for more information on this topic). Given two Banach spaces Y and Z, the π_1 norm of an operator $A : Y \to Z$, denoted $\pi_1(A)$, is defined to be the smallest constant $K > 0$ such that for every $m \in \mathbb{N}$ and every $y_1, \ldots, y_m \in Y$ there exists a norm 1 linear functional $y^* \in Y^*$ satisfying

$$\sum_{j=1}^{m}\|Ay_j\|_Z \le K\sum_{j=1}^{m}|y^*(y_j)|. \qquad (3.5)$$

This defines an *ideal norm* in the sense that it is a norm, and for every two operators $P : W \to Y$ and $Q : Z \to V$ we have $\pi_1(QAP) \le \|Q\| \cdot \pi_1(A) \cdot \|P\|$. Observe that it is always the case that $\pi_1(A) \ge \|A\|$.

LEMMA 3.3. *Using the above notation,* $\pi_1(\mathrm{Id}) \le 4n(n-1)$. *Therefore, Lemma* 3.2 *implies that*

$$\pi_1(\mathscr{F} \circ S \circ \mathrm{Id} \circ T^*) \le 4nL.$$

Proof. Fix $f_1, \ldots, f_m : X \to \mathbb{R}$ with $f_1(0) = \cdots = f_m(0) = 0$. Then

$$\begin{aligned}
\sum_{j=1}^{m} \|f_j\|_W &= \sum_{s=1}^{n-1}\sum_{t=0}^{n-2}\sum_{j=1}^{m}\big(|f_j(s,t+1) - f_j(s,t)| + |f_j(t+1,s) - f_j(t,s)|\big) \\
&\quad + (n+1)\sum_{t=0}^{n-2}\sum_{j=1}^{m}\big(|f_j(0,t+1) - f_j(0,t)| + |f_j(t+1,0) - f_j(t,0)|\big) \\
&\le 4n(n-1)\max\Bigg\{\max_{\substack{0\le s\le n-1\\ 0\le t\le n-2}}\sum_{j=1}^{m}|f_j(s,t+1) - f_j(s,t)|, \\
&\qquad\qquad \max_{\substack{0\le s\le n-1\\ 0\le t\le n-2}}\sum_{j=1}^{m}|f_j(t+1,s) - f_j(t,s)|\Bigg\}.
\end{aligned}$$

Assume without loss of generality that the maximum above equals $\sum_{j=1}^{m}|f_j(s_0, t_0 + 1) - f_j(s_0, t_0)|$ for some $0 \le s_0 \le n-1$ and $0 \le t_0 \le n-2$. Consider the measure $\mu = \delta_{(s_0,t_0+1)} - \delta_{(s_0,t_0)} \in \mathscr{M} = \mathrm{Lip}_0^*$. One checks that $\|\mu\|_\tau = 1$, and $\sum_{j=1}^{m}|f_j(s_0, t_0 + 1) - f_j(s_0, t_0)| = \sum_{j=1}^{m}|\mu(f_j)|$, implying the required result. □

The fundamental property of the π_1 norm is the Pietsch factorization theorem (see [51]), a special (particularly easy) case of which is the following lemma. We present a proof for the sake of completeness.

Lemma 3.4 (Pietsch factorization). *Let Y be a Banach space, and fix a linear operator $A : \ell_\infty^N \to Y$. Then there exists a probability measure σ on $\{1, \ldots, N\}$ and a linear operator $R : L_1(\sigma) \to Y$ such that $A = R \circ I$, where I is the formal identity from ℓ_∞^N to $L_1(\sigma)$, and $\|R\| = \pi_1(A)$.*

Proof. Recall that $A : \ell_\infty^N \to Y$ satisfies, for all $x_1, \ldots, x_m \in \ell_\infty^N$,

$$\sum_{j=1}^{m}\|Ax_j\| \le \pi_1(A)\cdot \mathrm{supp}_{\substack{x^*\in(\ell_\infty^N)^*\\ \|x^*\|=1}}\sum_{j=1}^{m}|x^*(x_j)| = \pi_1(A)\cdot\max_{1\le k\le N}\sum_{j=1}^{m}|x_j(k)|, \tag{3.6}$$

where the last equality follows from the fact that the evaluation functionals $x \mapsto x(k)$ are the extreme points of the unit ball of $\ell_1^N = \left(\ell_\infty^N\right)^*$.

Denoting by $e_1, \ldots, e_N$ the standard basis of $\mathbb{R}^N$, we deduce from (3.6) that $\pi_1(A) \ge \sum_{j=1}^{N}\|Ae_j\|$. Define a probability measure σ on $\{1, \ldots, N\}$ by $\sigma(k) = \frac{\|Ae_k\|}{\sum_{j=1}^{N}\|Ae_j\|}$. Then for every $x \in \ell_\infty^N$ we see that

$$\begin{aligned}
\|Ax\| = \left\|\sum_{k=1}^{N}x(k)Ae_k\right\| &\le \sum_{k=1}^{N}|x(k)|\cdot\|Ae_k\| \\
&= \left(\sum_{j=1}^{N}\|Ae_j\|\right)\int_{\{1,\ldots,N\}}|x(k)|d\sigma(k) \le \pi_1(A)\int_{\{1,\ldots,N\}}|x(k)|d\sigma(k).
\end{aligned}$$

Defining $Rx = Ax$, this implies the required result. □

From now on let R and σ be the operator and probability measure corresponding to $A = \mathscr{F} \circ S \circ \mathrm{Id} \circ T^*$ in Lemma 3.4. Thus $R \circ I = \mathscr{F} \circ S \circ \mathrm{Id} \circ T^*$ and $\|R\| \le 4nL$.

Schematically, we have the following commuting diagram:

$$\begin{tikzcd}
\ell_\infty^N \arrow[r, "T^*"] \arrow[dr, "I"] & \mathrm{Lip}_0 \arrow[r, "\mathrm{Id}"] & W \arrow[r, "S"] & \ell_2(X) \arrow[r, "\mathscr{F}"] & \ell_2(X) \\
& L_1(\sigma) \arrow[urrr, "R"] & & &
\end{tikzcd}$$

We need only one more simple result from classical Banach space theory. This result can be generalized to the case in which the target space ℓ_2 is replaced by a more general Banach lattice. For the sake of simplicity we shall prove here only what is needed to conclude the proof of Theorem 1.1.

LEMMA 3.5. *Let $R : L_1(\sigma) \to \ell_2$ be a linear operator. Fix $f : \{0,\dots,N\} \to [0,\infty)$. Then there is $x \in \ell_2$ with nonnegative coordinates such that*

$$R\left(\{g : \{0,\dots,N\} \to \mathbb{R} : \ \forall j,\ |g(j)| \le f(j)\}\right) \subseteq \{y \in \ell_2 : \ \forall j,\ |y_j| \le x_j\},$$

and $\|x\|_2 \le \|R\| \cdot \|f\|_{L_1(\sigma)}$.

Proof. R is given by a matrix $(R_{ij} : i = 1,\dots,N,\ j \in \mathbb{N})$. In other words, for every j, $(Rf)_j = \sum_{i=1}^N R_{ij} f(i)$. Observe that using this notation,

$$\|R\| = \max_{1\le i\le N} \left(\frac{1}{\sigma(i)^2} \sum_{j=1}^{\infty} R_{ij}^2 \right)^{1/2}. \tag{3.7}$$

Fix $g \in L_1(\sigma)$ such that for all $i \in \{1,\dots,N\}$, $|g(i)| \le f(i)$. Then for all j,

$$|(Rg)_j| \le \sum_{i=1}^N |R_{ij}| f(i) := x_j.$$

Now

$$\begin{aligned}
\|x\|_2 = \left[\sum_{j=1}^{\infty} \left(\sum_{i=1}^N |R_{ij}| f(i) \right)^2 \right]^{1/2} &\le \sum_{i=1}^N \left(\sum_{j=1}^{\infty} |R_{ij}|^2 f(i)^2 \right)^{1/2} \\
&= \sum_{i=1}^n \sigma(i) f(i) \left(\frac{1}{\sigma(i)^2} \sum_{j=1}^{\infty} R_{ij}^2 \right)^{1/2} \le \|R\| \cdot \|f\|_{L_1(\sigma)},
\end{aligned}$$

where we have used (3.7). □

We are now in position to conclude the proof of Theorem 1.1.

Proof of Theorem 1.1. For $(u,v) \in \{1,\dots,n-1\}^2$ define $\varphi_{u,v} : X \to \mathbb{R}$ by

$$\varphi_{u,v}(k,\ell) := \frac{1}{u+v} \cdot \sin\left(\frac{2\pi(uk + v\ell)}{n} \right).$$

Then $\varphi_{u,v}(0) = 0$, and one computes that $\|\varphi_{u,v}\|_{\mathrm{Lip}} < \frac{4\pi}{n}$. By the fact that T^* maps the unit ball of ℓ_∞^N onto the unit ball of Lip_0, it follows that there is $\phi_{u,v} \in \ell_\infty^N$ with $\|\phi_{u,v}\|_\infty \le \frac{4\pi}{n}$ and $T^*\phi_{u,v} = \varphi_{u,v}$. Now the functions $|I(\phi_{u,v})| \in L_1(\sigma)$ are pointwise bounded by the constant $\frac{4\pi}{n}$, and so by Lemma 3.5 there exists $x \in \ell_2(X)$ of norm at most $\frac{4\pi}{n}\|R\| \le 16\pi L$ such that $|R(I(\phi_{u,v}))|$ is bounded pointwise by x.

Note that

$$\begin{aligned}
&R \circ I(\phi_{u,v})(u,v)\\
&= \mathscr{F} \circ S \circ \mathrm{Id} \circ T^*(\phi_{u,v})(u,v)\\
&= \mathscr{F}(\varphi_{u,v})(u,v)\\
&= \frac{1}{n^2} \sum_{(k,\ell)\in X} \frac{1}{u+v} \cdot \sin^2\left(\frac{2\pi(uk+v\ell)}{n}\right)\\
&= \frac{1}{n^2(u+v)} \sum_{k=0}^{n-1}\sum_{\ell=0}^{n-1} \left(\frac{1}{2} - \frac{1}{4}\cdot e^{2\pi i\cdot\frac{2uk}{n}} \cdot e^{2\pi i\cdot\frac{2v\ell}{n}} - \frac{1}{4}\cdot e^{-2\pi i\cdot\frac{2uk}{n}} \cdot e^{-2\pi i\cdot\frac{2v\ell}{n}}\right)\\
&= \begin{cases} \frac{1}{2(u+v)}, & (u,v) \neq \left(\frac{n}{2},\frac{n}{2}\right),\\ 0, & (u,v) = \left(\frac{n}{2},\frac{n}{2}\right). \end{cases}
\end{aligned}$$

But

$$(16\pi L)^2 \ge \|x\|_2^2 \ge \sum_{u,v=1}^{n-1} x_{u,v}^2 \ge \sum_{u,v=1}^{n-1} \Big[R \circ I(\phi_{u,v})(u,v)\Big]^2 \ge \frac{1}{8}\sum_{u,v=1}^{n-1} \frac{1}{(u+v)^2} \ge \frac{\log n}{16},$$

where the last bound follows from comparison with the appropriate integrals. The proof of Theorem 1.1 is complete. □

3.1. Discretization and minimum weight matching. In this section we deduce Theorem 1.2 from Theorem 1.1. The main tool is the following theorem of Bourgain [6], which gives a quantitative version of Ribe's theorem [44].

THEOREM 3.6 (Bourgain's quantitative version of Ribe's theorem [6]). *There exists a universal constant C with the following property. Let Y and Z be Banach spaces,* $\dim(Y) = d$. *Assume that $\mathscr{Y}$ is an ϵ-net in the unit ball of Y, that $f : \mathscr{Y} \to Z$ satisfies* $\mathrm{dist}(f) \le D$, *and that* $\log\log\frac{1}{\epsilon} \ge Cd\log D$. *Then there exists an invertible linear operator $T : Y \to Z$ satisfying $\|T\| \cdot \|T^{-1}\| \le C \cdot D$.*

Proof of Theorem 1.2. Observe that for every $\mu \in \mathscr{M}$, the measure $\frac{1}{\mu^+(X)} \cdot (\mu^+ \otimes \mu^-)$ is in $\Pi(\mu^+,\mu^-)$. Thus

$$\begin{aligned}
\|\mu\|_\tau &\le \frac{1}{\mu^+(X)}\int_{X\times X} \|x-y\|_2 d\mu^+(x)d\mu^-(y)\\
&\le \sqrt{2}\cdot(n-1)\cdot\mu^+(X)\\
&\le 2n \cdot |\mathrm{supp}(\mu^+)| \cdot \|\mu\|_\infty\\
&\le 2n^3\|\mu\|_\infty.
\end{aligned}$$

On the other hand, as we have seen in the proof of Lemma 3.1, for every $\mu \in \mathscr{M}$, $\|\mu\|_\infty \le \|\mu\|_\tau$. It follows from these considerations, and Theorems 1.1 and 3.6, that for every integer $N \ge e^{e^{C'n^2\log\log n}}$, the set of probability measures $\mathscr{Y} \subseteq \mathscr{P}_X$ consisting of measures $\mu \in \mathscr{P}_X$ such that for all $x \in X$, $\mu(x) = k/N$ for some $k \in \{0,\dots,N\}$, satisfies $c_1(\mathscr{Y},\tau) = \Omega\left(\sqrt{\log n}\right)$. We pass to a family of subsets as follows. Let M be an integer which will be determined later. For every $\mu \in \mathscr{Y}$ we assign a subset $S_\mu \subseteq \{0,\dots,nM\}^2$ as follows. For every $(u,v) \in X = \{0,\dots,n-1\}^2$, if $\mu(u,v) = k/N$, where $k \in \{0,\dots,N\}$, then S_μ will contain arbitrary k distinct points from the set $(uM, vM) + \{0, \ldots, \lceil\sqrt{N}\rceil\}^2$. Provided $M \geq 4\sqrt{N}$, the sets $\{S_\mu\}_{\mu\in\mathrm{Y}}$ thus obtained are N point subsets of $\{0, \ldots, nM\}^2$, and it is straightforward to check that the minimum weight matching metric on $\{S_\mu\}_{\mu\in\mathrm{Y}}$ is bi-Lipschitz equivalent to $(\mathscr{Y}, \tau)$ with constant distortion. □

3.2. Uniform and coarse nonembeddability into Hilbert space. In this section we prove Theorem 1.3. We shall prove, in fact, that the space $\mathscr{M}_{[0,1]^2,\tau}$ does not embed uniformly or coarsely into L_2. We first recall the definitions of these important notions (see [4, 37] and the references therein for background on these concepts). Let (X, d_X) and (Y, d_Y) be metric spaces. For $f : X \to Y$ and $t > 0$ we define

$$\Omega_f(t) = \operatorname{supp}\{d_Y(f(x), f(y));\ d_X(x, y) \leq t\}$$

and

$$\omega_f(t) = \inf\{d_Y(f(x), f(y));\ d_N(x, y) \geq t\}.$$

Clearly, Ω_f and ω_f are nondecreasing, and for every $x, y \in X$,

$$\omega_f\left(d_X(x, y)\right) \leq d_Y(f(x), f(y)) \leq \Omega_f\left(d_X(x, y)\right).$$

With these definitions, f is uniformly continuous if $\lim_{t\to 0} \Omega_f(t) = 0$, and f is said to be a uniform embedding if f is injective and both f and f^{-1} are uniformly continuous. Also, f is said to be a coarse embedding if $\Omega_f(t) < \infty$ for all $t > 0$ and $\lim_{t\to\infty} \omega_f(t) = \infty$.

In what follows we will use the following standard notation: Given a sequence of Banach spaces $\left\{(Z_j, \|\cdot\|_{Z_j})\right\}_{j=1}^\infty$, the Banach space $(\bigoplus_{j=1}^\infty Z_j)_1$ is the space of all sequences $\overline{z} = (z_j)_{j=1}^\infty \in \prod_{j=1}^\infty Z_j$ such that $\|\overline{z}\| := \sum_{j=1}^\infty \|z_j\|_{Z_j} < \infty$. If for every $j \in \mathbb{N}$, $Z_j = Z_1$, we write $\ell_1(Z_1) = (\bigoplus_{j=1}^\infty Z_j)_1$.

Theorem 3.7. *The spaces $\{\mathscr{M}^0_{\{0,\ldots,n\}^2,\tau}\}_{n=1}^\infty$ do not admit a uniform or coarse embedding into L_2 with moduli uniformly bounded in n; i.e., there do not exist increasing functions $\omega, \Omega : [0, \infty) \to [0, \infty)$ which either satisfy $\lim_{t\to 0} \omega(t) = \lim_{t\to 0} \Omega(t) = 0$, or $\lim_{t\to\infty} \omega(t) = \infty$, and mappings $f_n : \mathscr{M}^0_{\{0,\ldots,n\}^2} \to L_2$, such that $\omega(\|\mu - \nu\|_\tau) \leq \|f_n(\mu) - f_n(\nu)\|_2 \leq \Omega(\|\mu - \nu\|_\tau)$ for all $\mu, \nu \in \mathscr{M}^0_{\{0,\ldots,n\}^2}$ and all n.*

Proof. If this is not the case, then by passing to a limit along an ultrafilter we easily deduce that $\mathscr{M}^0_{[0,1]^2,\tau}$ uniformly or coarsely embeds into an ultraproduct of Hilbert spaces and thus into L_2 (see [16, 17]). By a theorem of Aharoni, Maurey, and Mityagin [1] in the case of uniform embeddings, and a result of Randrianarivony [43] in the case of coarse embeddings, this implies that $\mathscr{M}^0_{[0,1]^2}$ is linearly isomorphic to a subspace of L_0. By a theorem of Nikišin [39], it follows that $\mathscr{M}^0_{[0,1]^2}$ is isomorphic to a subspace of $L_{1-\epsilon}$ for any $\epsilon \in (0, 1)$. We recall that it is an open problem posed by Kwapien (see the discussion in [28, 4]) whether a Banach space which linearly embeds into L_0 is linearly isomorphic to a subspace of L_1. If this were the case, we would have finished by Theorem 1.1. Since the solution of Kwapien's problem is unknown, we proceed as follows.

Let $\{S_j\}_{j=1}^\infty$ be a sequence of disjoint squares in $[0, 1]^2$ with

$$d(S_j, S_k) = \min_{a\in S_j,\ b\in S_k} \|a - b\|_2 > \max\left\{\operatorname{diam} S_j, \operatorname{diam} S_k\right\}. \tag{3.8}$$

Consider the linear subspace Y of $\mathscr{M}^0_{[0,1]^2}$ consisting of all measures μ satisfying $\operatorname{supp}(\mu) \subseteq \bigcup_{j=1}^\infty S_j$ and $\mu(S_j) = 0$ for all j. It is intuitively clear that in the computation of $\|\mu\|_\tau$ for $\mu \in Y$ the best transportation leaves each of the S_j invariant; i.e., it is enough to take the infimum in (2.1) only over measures $\pi \in \Pi(\mu, \nu)$ which are supported on $\bigcup_{j=1}^{\infty}(S_j \times S_j)$. This is proved formally as follows: Fix $\mu \in Y$, and write $\mu = \sum_{j=1}^{\infty} \mu_j$, where $\mathrm{supp}(\mu_j) \subseteq S_j$ and $\mu_j(S_j) = 0$ for all $j \in \mathbb{N}$. We claim that

$$\|\mu\|_{[0,1]^2,\tau} = \sum_{j=1}^{\infty} \|\mu_j\|_{S_j,\tau}. \tag{3.9}$$

If $\pi_j \in \Pi(\mu_j^+, \mu_j^-)$, then $\pi := \sum_{j=1}^{\infty} \pi_j \in \Pi(\mu^+, \mu^-)$. Thus $\|\mu\|_{[0,1]^2,\tau} \le \sum_{j=1}^{\infty} \|\mu_j\|_{S_j,\tau}$. To prove the reverse inequality take $\pi \in \Pi(\mu^+, \mu^-)$. For every $j = 1, 2, \ldots$ define a measure σ_j on S_j as follows: For $A \subseteq S_j$ set $\sigma_j(A) := \pi(A \times \bigcup_{k \neq j} S_k)$. Thus, in particular, by our assumption (3.8) for every $y \in S_j$,

$$\int_{S_j} \|x - y\|_2 d\sigma_j(x) = \int_{S_j \times \bigcup_{k \neq j} S_k} \|x - y\|_2 d\pi(x, z) \le \int_{S_j \times \bigcup_{k \neq j} S_k} \|x - z\|_2 d\pi(x, z). \tag{3.10}$$

Writing

$$\begin{aligned} \widetilde{\pi} &:= \pi \cdot \mathbf{1}_{\bigcup_{j=1}^{\infty}(S_j \times S_j)} + \sum_{j=1}^{\infty} \frac{1}{\sigma_j(S_j)} \cdot \sigma_j \otimes \sigma_j \\ &= \pi \cdot \mathbf{1}_{\bigcup_{j=1}^{\infty}(S_j \times S_j)} + \sum_{j=1}^{\infty} \frac{1}{\pi\left(S_j \times \bigcup_{k \neq j} S_k\right)} \cdot \sigma_j \otimes \sigma_j, \end{aligned}$$

it follows from our definitions that $\widetilde{\pi} \in \Pi(\mu^+, \mu^-)$ and $\widetilde{\pi}$ is supported on $\bigcup_{j=1}^{\infty}(S_j \times S_j)$. Moreover, for each j, $\widetilde{\pi}_j := \widetilde{\pi}|_{S_j} \in \Pi(\mu_j^+, \mu_j^-)$, so that

$$\begin{aligned} \sum_{j=1}^{\infty} \|\mu_j\|_{S_j,\tau} &\le \sum_{j=1}^{\infty} \int_{S_j \times S_j} \|x - y\|_2 d\widetilde{\pi}_j(x, y) \\ &= \int_{\bigcup_{j=1}^{\infty}(S_j \times S_j)} \|x - y\|_2 d\pi(x, y) \\ &\quad + \sum_{j=1}^{\infty} \frac{1}{\pi\left(S_j \times \bigcup_{k \neq j} S_k\right)} \cdot \int_{S_j \times S_j} \|x - y\|_2 d\sigma_j(x) d\sigma_j(y) \\ &\overset{(3.10)}{\le} \int_{\bigcup_{j=1}^{\infty}(S_j \times S_j)} \|x - y\|_2 d\pi(x, y) + \sum_{j=1}^{\infty} \int_{S_j \times \bigcup_{k \neq j} S_k} \|x - z\|_2 d\pi(x, z) \\ &= \int_{\left(\bigcup_{j=1}^{\infty} S_j\right) \times \left(\bigcup_{j=1}^{\infty} S_j\right)} \|x - y\|_2 d\pi(x, y). \end{aligned}$$

This concludes the proof of (3.9). It follows that Y is isometric to $(\bigoplus_{n=1}^{\infty} \mathscr{M}_{S_n,\tau}^0)_1$, which in turn is isometric to $\ell_1(\mathscr{M}_{[0,1]^2,\tau}^0)$. Now Kalton proved in [28] that if for some Banach space X, $\ell_1(X)$ is isomorphic to a subspace of L_0, then X is isomorphic to a subspace of L_1, and we finish by Theorem 1.1. □

Proof of Theorem 1.3. Assume for the sake of contradiction that there exists $C < \infty$ such that for all $n \in \mathbb{N}$, $c_2\left(\mathscr{P}_{\{0,\ldots,n\}^2}, \sqrt{\tau}\right) < C$. By the proof of Lemma 3.1, we know that the unit ball of $\mathscr{M}_{\{0,\ldots,n\}^2,\tau}$ is isometric to a subset of $(\mathscr{P}_{\{0,\ldots,n\}^2}, \tau)$. Thus by our assumption there exist mappings $f_n : \mathscr{M}_{\{0,\ldots,n\}^2} \to L_2$ such that for every $\mu, \nu \in \mathscr{M}_{\{0,\dots,n\}^2}$ with $\|\mu\|_\tau, \|\nu\|_\tau \le 1$,

$$\sqrt{\|\mu-\nu\|_\tau} \le \|f_n(\mu) - f_n(\nu)\|_2 \le C \cdot \sqrt{\|\mu-\nu\|_\tau}\ . \tag{3.11}$$

Let $\mathscr{U}$ be a free ultrafilter on $\mathbb{N}$. Define $\widetilde{f}_n : \mathscr{M}_{\{0,\dots,n\}^2} \to (L_2)_{\mathrm{U}}$ by $\widetilde{f}_n(\mu) = \left(\sqrt{j} \cdot f_n(\mu/j)\right)_{j=1}^{\infty}/\mathscr{U}$. Inequalities (3.11) imply that all $\mu, \nu \in \mathscr{M}_{\{0,\dots,n\}^2}$ satisfy $\sqrt{\|\mu-\nu\|_\tau} \le \|\widetilde{f}_n(\mu) - \widetilde{f}_n(\nu)\|_{(L_2)_{\mathrm{U}}} \le C \cdot \sqrt{\|\mu-\nu\|_\tau}$. Since the ultrapower $(L_2)_{\mathrm{U}}$ is isometric to a Hilbert space (see [16]), we arrive at a contradiction with Theorem 3.7. □

Remark 3.2. We believe that Theorem 1.3 can be made quantitative; i.e., one can give explicit quantitative estimates on the rate with which $c_2\left(\mathscr{P}_{\{0,\dots,n\}^2}, \sqrt{\tau}\right)$ tends to infinity. This would involve obtaining quantitative versions of the proofs in [1, 28, 43], which seems easy but somewhat tedious. We did not attempt to obtain such bounds.

Remark 3.3. We do not know whether $\left(\mathscr{P}_{[0,1]^2}, \tau\right)$ admits a uniform embedding into Hilbert space. The proof above actually gives that for all $\alpha \in (0,1]$, $\left(\mathscr{P}_{[0,1]^2,\tau}, \tau^\alpha\right)$ does not embed bi-Lipschitzly into Hilbert space. But, our proof exploits the homogeneity of the function $t \mapsto t^\alpha$ in an essential way, and so it does not apply to the case of more general moduli.

4. Upper bounds via Fourier analysis. In this section we prove Theorem 1.4 and discuss some related upper bounds. Given a measure μ on $\mathbb{Z}_n^2$, we decompose it as in (1.2), and we consider the linear operators A and B, from $\mathscr{M}_{\mathbb{Z}_n^2}$ to $L_1\left(\mathbb{Z}_n^2\right)$, defined in (1.3) and (1.4), respectively. One checks that the duals of these operators, $A^*, B^* : L_\infty\left(\mathbb{Z}_n^2\right) \to \mathscr{M}_{\mathbb{Z}_n^2}^* = \mathrm{Lip}_0\left(\mathbb{Z}_n^2\right)$, are given by

$$A^* f = \sum_{(u,v)\in\mathbb{Z}_n^2\setminus\{(0,0)\}} \frac{e^{-\frac{2\pi i u}{n}} - 1}{\left|e^{\frac{2\pi i u}{n}} - 1\right|^2 + \left|e^{\frac{2\pi i v}{n}} - 1\right|^2} \cdot \widehat{f}(u,v) \cdot (e_{uv} - 1) \tag{4.1}$$

and

$$B^* f = \sum_{(u,v)\in\mathbb{Z}_n^2\setminus\{(0,0)\}} \frac{e^{-\frac{2\pi i v}{n}} - 1}{\left|e^{\frac{2\pi i u}{n}} - 1\right|^2 + \left|e^{\frac{2\pi i v}{n}} - 1\right|^2} \cdot \widehat{f}(u,v) \cdot (e_{uv} - 1). \tag{4.2}$$

To check these identities the reader should verify that for all $\mu \in \mathscr{M}_{\mathbb{Z}_n^2}$, $\int_{\mathbb{Z}_n^2} f d(A\mu) = \int_{\mathbb{Z}_n^2} (A^* f) d\mu$, and similarly for B. (To this end, recall that $\mu\left(\mathbb{Z}_n^2\right) = 0$, so that $\widehat{\mu}(0,0) = 0$. This explains the subtraction of 1 in the identities (4.1) and (4.2).)

We claim that for every $\mu \in \mathscr{M}_{Z_n^2}$,

$$\|\mu\|_\tau \le \|A\mu\|_{L_1(\mathbb{Z}_n^2)} + \|B\mu\|_{L_1(\mathbb{Z}_n^2)} \le C \log n \cdot \|\mu\|_\tau, \tag{4.3}$$

where C is a universal constant. This will imply Theorem 1.4 since the mapping $\mu \mapsto \mu - U$, where U is the uniform probability measure on Z_n^2, is an isometric embedding of $\mathscr{P}_{\mathbb{Z}_n^2}$ into $\mathscr{M}_{\mathbb{Z}_2^n}$.

By duality, (4.3) is equivalent to the fact that the mapping $(f,g) \mapsto A^* f + B^* g$ from $L_\infty\left(\mathbb{Z}_n^2\right) \oplus L_\infty\left(\mathbb{Z}_n^2\right)$ to $\mathrm{Lip}_0\left(\mathbb{Z}_n^2\right)$ is a $C \log n$ quotient map; i.e., for every $(f,g) \in L_\infty\left(\mathbb{Z}_n^2\right) \oplus L_\infty\left(\mathbb{Z}_n^2\right)$,

$$\|A^* f + B^* g\|_{\mathrm{Lip}} \le C \log n \cdot \max\left\{\|f\|_\infty, \|g\|_\infty\right\}, \tag{4.4}$$

and for every $h \in \mathrm{Lip}_0\left(\mathbb{Z}_n^2\right)$, there is some $(f,g) \in L_\infty\left(\mathbb{Z}_n^2\right) \oplus L_\infty\left(\mathbb{Z}_n^2\right)$ satisfying $A^*f + B^*g = h$ and $\max\{\|f\|_\infty, \|g\|_\infty\} \le \|h\|_{\mathrm{Lip}}$. The second assertion is proved as follows: Take $f = \partial_1 h$ and $g = \partial_2 h$, where for $j = 1,2$, $\partial_j h(x) = h(x+e_j) - h(x)$ (here $e_1 = (1,0)$ and $e_2 = (0,1)$). Clearly, $\|f\|_\infty, \|g\|_\infty \le \|h\|_{\mathrm{Lip}}$, and

$$
\begin{aligned}
&A^*f + B^*g\\
&\quad= \sum_{(u,v)\in\mathbb{Z}_n^2\setminus\{(0,0)\}} \left(\frac{\left(e^{-\frac{2\pi i u}{n}} - 1\right)\cdot \widehat{\partial_1 h}(u,v) + \left(e^{-\frac{2\pi i v}{n}} - 1\right)\cdot \widehat{\partial_2 h}(u,v)}{\left|e^{\frac{2\pi i u}{n}} - 1\right|^2 + \left|e^{\frac{2\pi i v}{n}} - 1\right|^2}\right)(e_{uv} - 1)\\
&\quad= \sum_{(u,v)\in\mathbb{Z}_n^2\setminus\{(0,0)\}} \left(\frac{\left(e^{-\frac{2\pi i u}{n}} - 1\right)\cdot\left(e^{\frac{2\pi i u}{n}} - 1\right) + \left(e^{-\frac{2\pi i v}{n}} - 1\right)\cdot\left(e^{\frac{2\pi i v}{n}} - 1\right)}{\left|e^{\frac{2\pi i u}{n}} - 1\right|^2 + \left|e^{\frac{2\pi i v}{n}} - 1\right|^2}\right)\\
&\qquad\cdot \widehat{h}(u,v)(e_{uv} - 1)\\
&\quad= \sum_{(u,v)\in\mathbb{Z}_n^2\setminus\{(0,0)\}} \widehat{h}(u,v)e_{uv} - \sum_{(u,v)\in\mathbb{Z}_n^2\setminus\{(0,0)\}} \widehat{h}(u,v)\\
&\quad= \sum_{(u,v)\in\mathbb{Z}_n^2} \widehat{h}(u,v)e_{uv} = h,
\end{aligned}
$$

where we used the fact that $h(0) = 0$.

It remains to prove (4.4). To this end, it is enough to show that $\|A^*f\|_{\mathrm{Lip}} \le O(\log n)\cdot\|f\|_\infty$ and $\|B^*g\|_{\mathrm{Lip}} \le O(\log n)\cdot\|g\|_\infty$. We will establish this for A^*—the case of B^* is entirely analogous. Observe that

$$\|A^*f\|_{\mathrm{Lip}} \le \|\partial_1 A^*f\|_\infty + \|\partial_2 A^*f\|_\infty,$$

and so it is enough to establish the following two inequalities:

$$\left\|\sum_{(u,v)\in\mathbb{Z}_n^2\setminus\{(0,0)\}} \frac{\left|e^{\frac{2\pi i u}{n}} - 1\right|^2}{\left|e^{\frac{2\pi i u}{n}} - 1\right|^2 + \left|e^{\frac{2\pi i v}{n}} - 1\right|^2}\cdot \widehat{f}(u,v)e_{uv}\right\|_\infty \le O(\log n)\cdot\|f\|_\infty \tag{4.5}$$

and

$$\left\|\sum_{(u,v)\in\mathbb{Z}_n^2\setminus\{(0,0)\}} \frac{\left(e^{-\frac{2\pi i u}{n}} - 1\right)\cdot\left(e^{\frac{2\pi i v}{n}} - 1\right)}{\left|e^{\frac{2\pi i u}{n}} - 1\right|^2 + \left|e^{\frac{2\pi i v}{n}} - 1\right|^2}\cdot \widehat{f}(u,v)e_{uv}\right\|_\infty \le O(\log n)\cdot\|f\|_\infty. \tag{4.6}$$

Since for $p > 0$ the norms on $L_\infty\left(\mathbb{Z}_n^2\right)$ and $L_p\left(\mathbb{Z}_n^2\right)$ are equivalent with constant $n^{2/p}$ (by Hölder's inequality), it is enough to show that for $p \ge 2$,

$$\left\|\sum_{(u,v)\in\mathbb{Z}_n^2\setminus\{(0,0)\}} \frac{\left|e^{\frac{2\pi i u}{n}} - 1\right|^2}{\left|e^{\frac{2\pi i u}{n}} - 1\right|^2 + \left|e^{\frac{2\pi i v}{n}} - 1\right|^2}\cdot \widehat{f}(u,v)e_{uv}\right\|_p \le O(p)\cdot\|f\|_p \tag{4.7}$$

and

$$\left\|\sum_{(u,v)\in\mathbb{Z}_n^2\setminus\{(0,0)\}} \frac{\left(e^{-\frac{2\pi i u}{n}} - 1\right)\cdot\left(e^{\frac{2\pi i v}{n}} - 1\right)}{\left|e^{\frac{2\pi i u}{n}} - 1\right|^2 + \left|e^{\frac{2\pi i v}{n}} - 1\right|^2}\cdot \widehat{f}(u,v)e_{uv}\right\|_p \le O(p)\cdot\|f\|_p. \tag{4.8}$$

To prove inequalities (4.7) and (4.8) we will assume that n is odd (all of our results are valid for even n as well, and the proofs in this case require minor modifications). We think of $\mathbb{Z}_n^2$ as $[-(n-1)/2,\ (n-1)/2]^2 \cap \mathbb{Z}^2$. As before, given $m : \mathbb{Z}_n^2 \to \mathbb{C}$, we denote

$$\partial_1 m(x,y) = m(x+1,y) - m(x,y) \quad \text{and} \quad \partial_2 m(x,y) = m(x,y+1) - m(x,y).$$

Thus

$$\begin{aligned} &\partial_1^2 m(x,y) = m(x+2,y) - 2m(x+1,y) + m(x,y) \quad \text{and} \\ &\partial_2^2 m(x,y) = m(x,y+2) - 2m(x,y+1) + m(x,y), \end{aligned}$$

and

$$\partial_1\partial_2 m(x,y) = \partial_2\partial_1 m(x,y) = m(x+1,y+1) - m(x+1,y) - m(x,y+1) + m(x,y).$$

In what follows we think of m as a *Fourier multiplier* in the sense that it corresponds to a translation invariant operator T_m on $L_2\left(\mathbb{Z}_n^2\right)$ given by

$$T_m(f) := \sum_{(u,v)\in\mathbb{Z}_n^2} m(u,v) \cdot \widehat{f}(u,v) \cdot e_{uv}. \tag{4.9}$$

Recall that an operator $T : L_1\left(\mathbb{Z}_n^2\right) \to L_1\left(\mathbb{Z}_n^2\right)$ is said to be weak $(1,1)$ with constant K if for every $f : \mathbb{Z}_n^2 \to \mathbb{C}$ and every $a > 0$,

$$\left|\left\{((u,v) \in \mathbb{Z}_n^2 :\ |Tf(u,v)| \geq a\right\}\right| \leq \frac{K}{a} \cdot \|f\|_1 = \frac{K}{a} \cdot \sum_{(u,v)\in\mathbb{Z}_n^2} |f(u,v)|.$$

We will use the following discrete version of the Hörmander–Mihlin multiplier theorem [38, 18].

THEOREM 4.1 (Hörmander–Mihlin multiplier criterion on $\mathbb{Z}_n^2$). *For $j \in \mathbb{N}$ denote $Q_j = [-2^j, 2^j] \times [-2^j, 2^j]$. Fix $B > 0$ and $m : \mathbb{Z}_n^2 \to \mathbb{C}$ with $m(0,0) = 0$, and assume that for all $j = 0, 1, \ldots, \lfloor \log_2(n-1) \rfloor - 1$,*

$$\begin{aligned} \sum_{(u,v)\in(Q_j\setminus Q_{j-1})\cap\mathbb{Z}_n^2} &\big[2^{-2j}|m(u,v)|^2 + |\partial_1 m(u,v)|^2 + |\partial_2 m(u,v)|^2 \\ &+2^{2j}|\partial_1^2 m(u,v)|^2 + 2^{2j}|\partial_2^2 m(u,v)|^2 + 2^{2j}|\partial_1\partial_2 m(u,v)|^2\big] \leq B^2. \end{aligned}$$

Then the translation invariant operator T_m corresponding to m is weak $(1,1)$ with constant $O(B)$.

While the continuous version of the Hörmander–Mihlin multiplier theorem is a powerful tool which appears in several texts (e.g., in the books [12, 48, 52]), we could not locate a statement of the above discrete version in the literature. It is, however, possible to prove it using several minor modifications of the existing proofs. The standard proof of the Hörmander–Mihlin criterion is usually split into two parts. The first part, which is based on the Calderón–Zygmund decomposition, transfers virtually verbatim to the discrete setting—see Theorem 3 in Chapter 1 of [48] and Remark 8.1 there, which explains how this part of the proof transfers from $\mathbb{R}^n$ to the setting of finitely generated groups of polynomial growth (in fact, the Calderón–Zygmund decomposition itself, as presented in Theorem 2 in Chapter 1 of [48], is valid in the setting of general metric spaces equipped with a doubling measure). The second part of the proof of the Hörmander–Mihlin theorem, as presented in Theorem 2.5 of [18], requires several straightforward modifications in order to pass to the discrete setting. We leave the simple details to the reader. For the sake of readers that are not familiar with these aspects of Fourier analysis, we will later present a complete reduction to a continuous problem whose proof appears in print, which yields slightly worse bounds on the distortion guarantee.

In order to apply Theorem 4.1, we consider the following two multipliers,

(4.10)
$$m_1(u,v) := \frac{\left|e^{\frac{2\pi iu}{n}}-1\right|^2}{\left|e^{\frac{2\pi iu}{n}}-1\right|^2+\left|e^{\frac{2\pi iv}{n}}-1\right|^2} \quad \text{and} \quad m_2(u,v) := \frac{\left(e^{-\frac{2\pi iu}{n}}-1\right)\cdot\left(e^{\frac{2\pi iv}{n}}-1\right)}{\left|e^{\frac{2\pi iu}{n}}-1\right|^2+\left|e^{\frac{2\pi iv}{n}}-1\right|^2},$$

where we set $m_1(0,0) = m_2(0,0) = 0$. A direct (albeit tedious!) computation shows that m_1 and m_2 satisfy the conditions of Theorem 4.1 with $B = O(1)$. Thus, the operators T_{m_1} and T_{m_2} are weak $(1,1)$ with constant $O(1)$. Since m_1 and m_2 are bounded functions, the operator norms $\|T_{m_1}\|_{L_2(\mathbb{Z}_n^2)\to L_2(\mathbb{Z}_n^2)}$ and $\|T_{m_2}\|_{L_2(\mathbb{Z}_n^2)\to L_2(\mathbb{Z}_n^2)}$ are $O(1)$. Since these operators are self-adjoint, by the Marcinkiewicz interpolation theorem (see [56]) it follows that for $p \ge 2$, the operator norms $\|T_{m_1}\|_{L_p(\mathbb{Z}_n^2)\to L_p(\mathbb{Z}_n^2)}$ and $\|T_{m_2}\|_{L_p(\mathbb{Z}_n^2)\to L_p(\mathbb{Z}_n^2)}$ are $O(p)$. This is precisely (4.7) and (4.8).

The above argument is based on Theorem 4.1, which does not appear exactly as stated in the literature, but its proof is a straightforward adaptation of existing proofs (which is too simple to justify rewriting the lengthy argument here). However, making the necessary changes easily does require some familiarity with Calderón–Zygmund theory. We therefore now present another argument which gives a polylog(n) bound on the distortion but uses only statements which appear in the literature. This alternative approach appears to be quite versatile and might be useful elsewhere.

The following lemma reduces the problem of proving inequalities such as (4.7) and (4.8) (with perhaps a different dependence on p) to a continuous inequality. The argument is based on the proof of a theorem of Marcinkiewicz from [56] (see Theorem 7.5 in Chapter X there). In what follows we denote by $\mathbb{T}$ the Euclidean unit circle in the plane.

Proposition 4.2 (transferring multipliers from the torus to $\mathbb{Z}_n^2$). *Fix an odd integer n. Let $\{\lambda(u,v)\}_{u,v=0}^{\infty}$ be complex numbers such that $\lambda(u,v) = 0$ for $\max\{u,v\} \ge n$. Consider the operators $M : L_p\left(\mathbb{T}^2\right) \to L_p\left(\mathbb{T}^2\right)$ and $M_n : L_p\left(\mathbb{Z}_n^2\right) \to L_p\left(\mathbb{Z}_n^2\right)$ given by*

$$M\left(\sum_{u,v=-\infty}^{\infty} \widehat{f}(u,v)e^{2\pi i(ux+vy)}\right) = \sum_{u,v=0}^{\infty} \lambda(u,v)\widehat{f}(u,v)e^{2\pi i(ux+vy)}$$

and

$$M_n\left(\sum_{u,v=0}^{n-1} \widehat{f}(u,v)e^{\frac{2\pi i}{n}(ua+vb)}\right) = \sum_{u,v=0}^{n-1} \lambda(u,v)\widehat{f}(u,v)e^{\frac{2\pi i}{n}(ua+vb)}.$$

Then

$$\|M_n\|_{L_p(\mathbb{Z}_n^2)\to L_p(\mathbb{Z}_n^2)} \le 81\cdot\|M\|_{L_p(\mathbb{T}^2)\to L_p(\mathbb{T}^2)}.$$

Proof. The proof is a variant of the first part of the proof of Theorem 7.5 in Chapter X in [56] and a small twist on the second part. Since the terminology in [56] is different from ours, we repeat the proof of the first part as well. Recall that the Dirichlet kernels $D_\ell : [0,1] \to \mathbb{C}$ are defined as

$$D_\ell(x) = \sum_{j=-\ell}^{\ell} e^{2\pi i j x},$$

and the Fejér kernels $K_m : [0,1] \to \mathbb{C}$ are

$$K_m(x) = \frac{1}{m+1}\sum_{\ell=0}^{m} D_\ell x = \sum_{j=-m}^{m}\left(1 - \frac{|j|}{m+1}\right)e^{2\pi i j x}.$$

A basic property of D_ℓ is that for any trigonometric polynomial $S(x)$ of degree at most ℓ, namely $S(x) = \sum_{j=-\ell}^{\ell} a_j e^{2\pi i j x}$, we have that $S(x) = S * D_\ell(x) = \int_0^1 S(t)D_\ell(x-t)dt$. The same is true with any other function all of whose jth Fourier coefficients for j between $-\ell$ and ℓ are 1, in particular for the de la Vallée Poussin kernel $2K_{2\ell-1} - K_{\ell-1}$ (see [29]). The well-known advantage of the Fejér kernel over the Dirichlet kernel is that it is everywhere (real and) nonnegative. Note also that $\int_0^1 K_m(t)dt = 1$ for all m. Thus, by convexity of the function t^p, for any trigonometric polynomial S of degree at most ℓ, and for all $x \in [0,1]$,

$$\begin{aligned} |S(x)|^p &= |2S * K_{2\ell-1}(x) - S * K_\ell(x)|^p \\ &\le 3^p\left(\frac{2}{3}\int_0^1 |S(t)|^p K_{2\ell-1}(x-t)dt + \frac{1}{3}\int_0^1 |S(t)|^p K_{\ell-1}(x-t)dt\right). \end{aligned} \tag{4.11}$$

Now let $\omega_{2\ell+1}$ be the measure which assigns mass $\frac{1}{2\ell+1}$ to each of $2\ell+1$ equally spaced points on $[0,1]$. Then it is easy to check that

$$\int_0^1 K_m(x-t)d\omega_{2\ell+1}(x) = \int_0^1 K_m(x-t)dx = 1$$

for all $m \le 2\ell$ and for all $t \in [0,1]$. Integrating (4.11) with respect to $\omega_{2\ell+1}$, we get that for any trigonometric polynomial S of degree at most ℓ

$$\int_0^1 |S(x)|^p d\omega_{2\ell+1}(x) \le 3^p \int_0^1 |S(x)|^p dx. \tag{4.12}$$

It follows that if $S(x,y)$ is a two-variable trigonometric polynomial of degree at most ℓ in each of the variables, i.e., $S(x,y) = \sum_{u,v=-\ell}^{\ell} a_{uv} e^{2\pi i(ux+vy)}$, then

$$\int_{[0,1]^2} |S(x,y)|^p d\omega_{2\ell+1}(x)d\omega_{2\ell+1}(y) \le 9^p \int_{[0,1]^2} |S(x,y)|^p dxdy.$$

It follows from this that, since n is odd, for every $f \in L_p\left(\mathbb{T}^2\right)$,

$$\left\|M_n\left(\sum_{u,v=0}^{n-1}\widehat{f}(u,v)e^{\frac{2\pi i}{n}(ua+vb)}\right)\right\|_{L_p(\mathbb{Z}_n^2)} \le 9\left\|M\left(\sum_{u,v=-\infty}^{\infty}\widehat{f}(u,v)e^{2\pi i(ux+vy)}\right)\right\|_{L_p(\mathbb{T}^2)}.$$

For each trigonometric polynomial of the form $P(x,y) = \sum_{u,v=-n+1}^{n-1} a_{uv} e^{2\pi i(ux+vy)}$, note that

$$\int_{[0,1]^2} P(x,y)d\omega_n(x)d\omega_n(y) = a_0 = \int_{[0,1]^2} P(x,y)dxdy.$$

Fix $f \in L_p\left(\mathbb{Z}_n^2\right)$, $1 < p < \infty$. By the first part of the proof and duality, there is $g \in L_{p^*}(\mathbb{T}^2)$ $(p^* = p/(p-1))$ with $\|g\|_{p^*} = 1$ such that

$$
\begin{aligned}
&\|M_n f\|_{L_p(\mathbb{Z}_n^2)} \\
&\quad \le 9 \int_{[0,1]^2} \left(\sum_{u,v=0}^{n-1} \lambda_j \widehat{f}(u,v) e^{2\pi i(ux+vy)} \right) \overline{g(x,y)} dxdy \\
&\quad = 9 \int_{[0,1]^2} \left(\sum_{u,v=0}^{n-1} \lambda(u,v) \widehat{f}(u,v) e^{2\pi i(ux+vy)} \right) \left(\sum_{u,v=0}^{n-1} \overline{\widehat{g}(u,v)} e^{-2\pi i(ux+vy)} \right) dxdy \\
&\quad = 9 \int_{[0,1]^2} \left(\sum_{u,v=0}^{n-1} \lambda(u,v) \widehat{f}(u,v) e^{2\pi i(ux+vy)} \right) \left(\sum_{u,v=0}^{n-1} \overline{\widehat{g}(u,v)} e^{-2\pi i(ux+vy)} \right) d\omega_n(x) d\omega_n(y) \\
&\quad = 9 \int_{[0,1]^2} \left(\sum_{u,v=0}^{n-1} \widehat{f}(u,v) e^{2\pi i(ux+vy)} \right) \left(\sum_{u,v=0}^{n-1} \lambda(u,v) \overline{\widehat{g}(u,v)} e^{-2\pi i(ux+vy)} \right) d\omega_n(x) d\omega_n(y) \\
&\quad \le 9 \left(\int_{[0,1]^2} \left| \sum_{u,v=0}^{n-1} \widehat{f}(u,v) e^{2\pi i(ux+vy)} \right|^p d\omega_n(x) d\omega_n(y) \right)^{1/p} \\
&\qquad \cdot \left(\int_{[0,1]^2} \left| \sum_{u,v=0}^{n-1} \lambda(u,v) \overline{\widehat{g}(u,v)} e^{-2\pi i(ux+vy)} \right|^{p^*} d\omega_n(x) \omega_n(y) \right)^{1/p^*} \\
&\quad \le 81 \cdot \|f\|_{L_p(\mathbb{Z}_n^2)} \left(\int_{[0,1]^2} \left| \sum_{u,v=0}^{n-1} \overline{\lambda(u,v)} \widehat{g}(u,v) e^{2\pi i(ux+vy)} \right|^{p^*} dx \right)^{1/p^*} \\
&\quad \le 81 \cdot \|f\|_{L_p(\mathbb{Z}_n^2)} \cdot \|M\|_{L_p(\mathbb{T}^2) \to L_p(\mathbb{T}^2)},
\end{aligned}
$$

where the second to last inequality follows from (4.12) and the last inequality (that is, the fact that the norm of a multiplier in $L_p\left(\mathbb{T}^2\right)$ is the same as the norm of the conjugate multiplier in $L_{p^*}\left(\mathbb{T}^2\right)$) follows from duality. The case $p = 1$ (and also a similar inequality for the ∞ norm) follows easily from the L_p cases. □

Proposition 4.2 implies that it is enough to obtain L_p to L_p bounds for the operators T_{m_1} and T_{m_2}, where m_1, m_2 are as in (4.10), as operators on functions on the torus $\mathbb{T}^2$. By a theorem of de Leeuw [11], it is enough to obtain such bounds when we think of T_{m_1} and T_{m_2} as operators on functions on $\mathbb{R}^2$ (see [55] for the respective result in the case of weak $(1,1)$ bounds). The continuous version of the Hörmander–Mihlin multiplier theorem now applies, but unfortunately its conditions are not satisfied. However, a (once again tedious) computation shows it is possible to apply the Marcinkiewicz multiplier theorem (see [47, 52]), in combination with bounds on the Hilbert transform [47, 52], to obtain bounds similar to (4.7) and (4.8) with $O(p)$ replaced by $O(\text{poly}(p))$. (It is quite easy to obtain a bound of $O(p^3)$, and with more work this can be reduced to $O(p^2)$. However, we do not see a simple way to obtain $O(p)$ using this approach.)

Remark 4.1. Consider the mapping $S : \mathscr{P}_{\mathbb{Z}_n^2} \to L_1\left(\mathbb{Z}_n^2\right)$ given by

$$
S\mu := \sum_{(u,v) \in \mathbb{Z}_n^2 \setminus \{(0,0)\}} \left(\left| e^{\frac{2\pi i u}{n}} - 1 \right| + \left| e^{\frac{2\pi i v}{n}} - 1 \right| \right) \cdot \widehat{\mu}(u,v) e_{uv}.
$$

Using considerations similar to the above (see Proposition III.A.3 in [54] for a continuous counterpart), it is possible to show that S has distortion $O(\text{polylog}(n))$. However, we were unable to get this bound down to $O(\log n)$ as in Theorem 1.4. Nevertheless, this embedding might be of interest since it reduces the dimension of the ambient L_1 space by a factor of 2.

5. Discussion and open problems. There are several interesting problems that arise from the results presented in this paper—we shall discuss some of them in the list below.

1. The most natural problem that is left open is to determine the asymptotic behavior of $c_1\left(\{0,1\ldots,n\}^2,\tau\right)$. It seems hard to use the ideas in section 4 to obtain an embedding of distortion $O\left(\sqrt{\log n}\right)$, as the known bounds on multipliers usually give a weak $(1,1)$ inequality at best. We do not know the actual distortion of the embedding in Theorem 1.4.
2. Remark 4.1 implies that the Banach–Mazur distance between the (n^2-1)-dimensional normed space $\mathscr{M}_{\mathbb{Z}_n^2,\tau}$ and $\ell_1^{n^2-1}$ is $O(\text{polylog}(n))$. It would be interesting to determine the asymptotic behavior of this distance. In particular, it is not clear whether the L_1 (embedding) distortion of $\mathscr{M}_{\mathbb{Z}_n^2,\tau}$ behaves differently from its Banach–Mazur distance from $\ell_1^{n^2-1}$.
3. We did not attempt to study the L_1 distortion of $\mathscr{M}_{\{0,1,\ldots,n\}^d,\tau}$ for $d \geq 3$. Observe that this space contains $\mathscr{M}_{\{0,1,\ldots,n\}^2,\tau}$, and so the $\Omega\left(\sqrt{\log n}\right)$ lower bound still applies. But the result of [30] shows that the transportation cost metric on the Hamming cube $\{0,1\}^d$ has distortion $\Theta(d)$, and so some improvements are still possible. Note that in higher dimensions it becomes interesting to study the transportation cost distance when $\mathbb{R}^d$ is equipped with other norms. The Banach–Mazur distance between ℓ_1^d and arbitrary d-dimensional norms has been studied in [7, 49, 13]. In particular, the result of [13] states that any d-dimensional Banach space is at distance $O\left(d^{5/6}\right)$ from ℓ_1^d. Combining this fact with the lower bound on the L_1 distortion of the transportation cost distance on the Hamming (ℓ_1) cube cited above, we see that for any norm $\|\cdot\|$ on $\mathbb{R}^d$, $c_1\left(\mathscr{P}_{(\mathbb{R}^d,\|\cdot\|),\tau}\right) = \Omega\left(d^{1/6}\right)$. It would be interesting to study the dependence on d for general norms on $\mathbb{R}^d$.
4. As stated in Remark 3.2, it would be interesting to study the rate with which the Euclidean distortion $c_2\left(\mathscr{P}_{\{0,\ldots,n\}^2},\sqrt{\tau}\right)$ tends to infinity.
5. As stated in Remark 3.3, we do not know whether $\left(\mathscr{P}_{[0,1]^2},\tau\right)$ admits a uniform embedding into Hilbert space.
6. The present paper rules out the "low distortion approach" to nearest neighbor search in the Earthmover metric via embeddings into L_1. However, it might still be possible to find *nearest neighbor preserving embeddings* into L_1 in the sense of [25].
7. On the more "applied side," as stated in the introduction, there is a possibility that the embedding of Theorem 1.4 behaves better than the theoretical distortion guarantee of $O(\log n)$ in "real life" situations, since it is often the case that the bulk of the Fourier spectrum is concentrated on a sparse set of frequencies. Additionally, it might be worthwhile to "thin out" some frequencies of the given set of images before embedding into L_1 (and then using the known L_1 nearest neighbor search databases). It would be interesting to carry out such "tweaking" of our algorithm in a more experimental setting.

Acknowledgments. We are grateful to David Jerison and Terence Tao for several helpful suggestions. The first author wishes to thank Piotr Indyk for his encouragement to work on the planar Earthmover problem. We also thank Guillaume Aubrun for correcting several mistakes in our original manuscript.

REFERENCES

[1] I. Aharoni, B. Maurey, and B. S. Mityagin, *Uniform embeddings of metric spaces and of Banach spaces into Hilbert spaces*, Israel J. Math., 52 (1985), pp. 251–265.

[2] A. Archer, J. Fakcharoenphol, C. Harrelson, R. Krauthgamer, K. Talwar, and É. Tardos, *Approximate classification via earthmover metrics*, in SODA '04: Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, 2004, pp. 1072–1080.

[3] S. Arora, J. R. Lee, and A. Naor, *Euclidean distortion and the sparsest cut*, in STOC '05: Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing, ACM Press, New York, 2005, pp. 553–562.

[4] Y. Benyamini and J. Lindenstrauss, *Geometric Nonlinear Functional Analysis*, Vol. 1, Amer. Math. Soc. Colloq. Publ. 48, AMS, Providence, RI, 2000.

[5] E. Berkson, J. Bourgain, A. Pełczynski, and M. Wojciechowski, *Canonical Sobolev projections of weak type* $(1,1)$, Mem. Amer. Math. Soc., 150 (2001).

[6] J. Bourgain, *Remarks on the extension of Lipschitz maps defined on discrete sets and uniform homeomorphisms*, in Geometrical Aspects of Functional Analysis (1985/86), Lecture Notes in Math. 1267, Springer-Verlag, Berlin, 1987, pp. 157–167.

[7] J. Bourgain and S. J. Szarek, *The Banach-Mazur distance to the cube and the Dvoretzky-Rogers factorization*, Israel J. Math., 62 (1988), pp. 169–180.

[8] M. S. Charikar, *Similarity estimation techniques from rounding algorithms*, in STOC '02: Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing, ACM Press, New York, 2002, pp. 380–388.

[9] C. Chekuri, S. Khanna, J. Naor, and L. Zosin, *Approximation algorithms for the metric labeling problem via a new linear programming formulation*, in SODA '01: Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, 2001, pp. 109–118.

[10] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, *Locality-sensitive hashing scheme based on p-stable distributions*, in SoCG '04: Proceedings of the Twentieth Annual Symposium on Computational Geometry, ACM Press, New York, 2004, pp. 253–262.

[11] K. de Leeuw, *On L_p multipliers*, Ann. of Math. (2), 81 (1965), pp. 364–379.

[12] J. García-Cuerva and J. L. Rubio de Francia, *Weighted Norm Inequalities and Related Topics*, North–Holland Math. Stud. 116, North–Holland, Amsterdam, 1985.

[13] A. Giannopoulos, *A note on the Banach-Mazur distance to the cube*, in Geometric Aspects of Functional Analysis, Oper. Theory Adv. Appl. 77, J. Lindenstrauss and V. D. Milman, eds., Birkhäuser, Basel, 1995, pp. 67–73.

[14] L. J. Guibas, Y. Rubner, and C. Tomassi, *The earth mover's distance as a metric for image retrieval*, International J. Comput. Vis., 40 (2000), pp. 99–121.

[15] L. J. Guibas, Y. Rubner, and C. Tomassi, *A metric for distributions with applications to image databases*, in ICCV '98: Proceedings of the Sixth IEEE International Conference on Computer Vision, IEEE Computer Society Press, Los Alamitos, CA, 2003, pp. 59–66.

[16] S. Heinrich, *Ultraproducts in Banach space theory*, J. Reine Angew. Math., 313 (1980), pp. 72–104.

[17] S. Heinrich and P. Mankiewicz, *Applications of ultrapowers to the uniform and Lipschitz classification of Banach spaces*, Studia Math., 73 (1982), pp. 225–251.

[18] L. Hörmander, *Estimates for translation invariant operators in L^p spaces*, Acta Math., 104 (1960), pp. 93–140.

[19] P. Indyk, *Algorithmic applications of low-distortion geometric embeddings*, in FOCS '01: Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, CA, 2001, pp. 10–33.

[20] P. Indyk, *Stable distributions, pseudorandom generators, embeddings and data stream computation*, in FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, CA, 2001, pp. 189–197.

[21] P. Indyk, *Algorithms for dynamic geometric problems over data streams*, in STOC '04: Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing, ACM Press, New York, 2004, pp. 373–380.

[22] P. Indyk, *Nearest neighbors in high-dimensional spaces*, in Handbook of Discrete and Computational Geometry, 2nd ed., CRC Press, Boca Raton, FL, 2004, pp. 877–892.

[23] P. Indyk and J. Matoušek, *Low distortion embeddings of finite metric spaces*, in Handbook of Discrete and Computational Geometry, 2nd ed., CRC Press, Boca Raton, FL, 2004, pp. 177–196.

[24] P. Indyk and R. Motwani, *Approximate nearest neighbors: Towards removing the curse of dimensionality*, in STOC '98: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, ACM Press, New York, 1998, pp. 604–613.

[25] P. Indyk and A. Naor, *Nearest neighbor preserving embeddings*, ACM Trans. Algorithms, to appear.

[26] P. Indyk and N. Thaper, *Fast image retrieval via embeddings*, in ICCV '03: Proceedings of the 3rd International Workshop on Statistical and Computational Theories of Vision, 2003.

[27] S. Kakutani, *Concrete representation of abstract* (L)*-spaces and the mean ergodic theorem*, Ann. of Math. (2), 42 (1941), pp. 523–537.

[28] N. J. Kalton, *Banach spaces embedding into* L_0, Israel J. Math., 52 (1985), pp. 305–319.

[29] Y. Katznelson, *An Introduction to Harmonic Analysis*, 3rd ed., Cambridge Mathematical Library, Cambridge University Press, Cambridge, UK, 2004.

[30] S. Khot and A. Naor, *Nonembeddability theorems via Fourier analysis*, Math. Ann., 334 (2006), pp. 821–852.

[31] S. Khot and N. Vishnoi, *The unique games conjecture, integrality gap for cut problems, and embeddability of negative type metrics into* L_1, in FOCS '05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, CA, 2005, pp. 53–62.

[32] S. Kislyakov, *Sobolev imbedding operators and the nonisomorphism of certain Banach spaces*, Funct. Anal. Appl., 9 (1975), pp. 290–294.

[33] J. Lindenstrauss and H. P. Rosenthal, *The* $\mathscr{L}_p$ *spaces*, Israel J. Math., 7 (1969), pp. 325–349.

[34] J. Lindenstrauss and L. Tzafriri, *Classical Banach Spaces.* II. *Function Spaces*, Ergeb. Math. Grenzgeb. 97, Springer-Verlag, Berlin, 1979.

[35] J. Matoušek, *Open Problems on Embeddings of Finite Metric Spaces*, http://kam.mff.cuni.cz/~matousek/metrop.ps.gz.

[36] J. Matoušek, *Lectures on Discrete Geometry*, Grad. Texts in Math. 212, Springer-Verlag, New York, 2002.

[37] M. Mendel and A. Naor, *Metric cotype*, Ann. Math., to appear.

[38] S. G. Mihlin, *On the multipliers of Fourier integrals*, Dokl. Akad. Nauk SSSR (N.S.), 109 (1956), pp. 701–703.

[39] E. M. Nikišin, *A resonance theorem and series in eigenfunctions of the Laplace operator*, Izv. Akad. Nauk SSSR Ser. Mat., 36 (1972), pp. 795–813.

[40] A. Pełczyński, *Boundedness of the canonical projection for Sobolev spaces generated by finite families of linear differential operators*, in Analysis at Urbana. Vol. 1: Analysis in Function Spaces, London Math. Soc. Lecture Note Ser. 137, Cambridge University Press, Cambridge, UK, 1989, pp. 395–415.

[41] A. Pełczyński and M. Wojciechowski, *Sobolev spaces*, in Handbook of the Geometry of Banach Spaces, Vol. 2, North–Holland, Amsterdam, 2003, pp. 1361–1423.

[42] S. Peleg, M. Werman, and H. Rom, *A unified approach to the change of resolution: Space and gray-level*, IEEE Trans. Pattern Anal. Mach. Intell., 11 (1989), pp. 739–742.

[43] N. L. Randrianarivony, *Characterization of quasi-Banach spaces which coarsely embed into a Hilbert space*, Proc. Amer. Math. Soc., 134 (2006), pp. 1315–1317.

[44] M. Ribe, *On uniformly homeomorphic normed spaces*, Ark. Mat., 14 (1976), pp. 237–244.

[45] D. N. Rockmore, *Efficient computation of Fourier inversion for finite groups*, J. Assoc. Comput. Mach., 41 (1994), pp. 31–66.

[46] W. Rudin, *Real and Complex Analysis*, 3rd ed., McGraw–Hill, New York, 1987.

[47] E. Stein, *Singular Integrals and Differentiability Properties of Functions*, Princeton University Press, Princeton, NJ, 1970.

[48] E. M. Stein, *Harmonic Analysis: Real-Variable Methods, Orthogonality, and Oscillatory Integrals*, Princeton Math. Ser. 43, Princeton University Press, Princeton, NJ, 1993.

[49] S. J. Szarek and M. Talagrand, *An "isomorphic" version of the Sauer-Shelah lemma and the Banach-Mazur distance to the cube*, in Geometric Aspects of Functional Analysis (1987–88), Lecture Notes in Math. 1376, Springer-Verlag, Berlin, 1989, pp. 105–112.

[50] M. Talagrand, *Embedding subspaces of* L_1 *into* l_1^N, Proc. Amer. Math. Soc., 108 (1990), pp. 363–369.

[51] N. Tomczak-Jaegermann, *Banach-Mazur Distances and Finite-Dimensional Operator Ideals*, Pitman Monogr. Surveys Pure Appl. Math. 38, Longman Scientific and Technical, Harlow, UK, 1989.

[52] A. Torchinsky, *Real-Variable Methods in Harmonic Analysis*, Dover, Mineola, NY, 2004.

[53] C. Villani, *Topics in Optimal Transportation*, Grad. Stud. Math. 58, AMS, Providence, RI, 2003.

[54] P. Wojtaszczyk, *Banach Spaces for Analysts*, Cambridge Stud. Adv. Math. 25, Cambridge University Press, Cambridge, UK, 1991.

[55] K. Woźniakowski, *A new proof of the restriction theorem for weak type* $(1,1)$ *multipliers on* $\mathbf{R}^n$, Illinois J. Math., 40 (1996), pp. 479–483.

[56] A. Zygmund, *Trigonometric Series*, Vols. I and II, 3rd ed., Cambridge Mathematical Library, Cambridge University Press, Cambridge, UK, 2002.

SIAM J. COMPUT.
Vol. 37, No. 3, pp. 827–844

LEARNING MONOTONE DECISION TREES IN POLYNOMIAL TIME*

RYAN O'DONNELL[†] AND ROCCO A. SERVEDIO[‡]

Abstract. We give an algorithm that learns any monotone Boolean function $f\colon \{-1,1\}^n \to \{-1,1\}$ to any constant accuracy, under the uniform distribution, in time polynomial in n and in the decision tree size of f. This is the first algorithm that can learn arbitrary monotone Boolean functions to high accuracy, using random examples only, in time polynomial in a reasonable measure of the complexity of f. A key ingredient of the result is a new bound showing that the average sensitivity of any monotone function computed by a decision tree of size s must be at most $\sqrt{\log s}$. This bound has proved to be of independent utility in the study of decision tree complexity [O. Schramm, R. O'Donnell, M. Saks, and R. Servedio, *Every decision tree has an influential variable*, in Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 2005, pp. 31–39]. We generalize the basic inequality and learning result described above in various ways—specifically, to partition size (a stronger complexity measure than decision tree size), p-biased measures over the Boolean cube (rather than just the uniform distribution), and real-valued (rather than just Boolean-valued) functions.

Key words. learning, monotone, decision trees

AMS subject classifications. 42A16, 68Q25, 68T05

DOI. 10.1137/060669309

1. Introduction.

1.1. Computationally efficient learning from random examples.

In the two decades since Valiant introduced the probably approximately correct (PAC) learning model [31], a major goal in computational learning theory has been the design of computationally efficient algorithms for learning Boolean functions from random examples. The original distribution-free PAC learning model of Valiant required that for *any* distribution $\mathcal{D}$ over the domain of examples (which throughout this paper is $\{-1,1\}^n$), the learning algorithm must with high probability succeed in generating a hypothesis for the unknown target function which is highly accurate relative to $\mathcal{D}$. Despite much effort over a twenty year span, very few efficient learning algorithms have been obtained in this demanding model. Thus the focus of much work has shifted to the natural *uniform distribution* PAC learning model, in which the examples used for learning are uniformly distributed over $\{-1,1\}^n$ (we give a precise definition of this learning model in section 2).

An easy information-theoretic argument shows that no poly(n)-time algorithm can learn arbitrary Boolean functions $f : \{-1,1\}^n \to \{-1,1\}$ to accuracy nonnegligibly better than 1/2. Consequently, the most ambitious conceivable goal in uniform

*Received by the editors September 7, 2006; accepted for publication (in revised form) January 22, 2007; published electronically July 27, 2007.

http://www.siam.org/journals/sicomp/37-3/66930.html

[†] Theory Group, Microsoft Research, Redmond, WA 98052. Current address: Department of Computer Science, Carnegie Mellon Univeristy, Pittsburgh, PA 15213 (odonnell@microsoft.com, odonnell@cs.cmu.edu). Some of this work was done while this author was at the Institute for Advanced Study, supported by the National Science Foundation under agreement CCR-0324906. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

[‡] Department of Computer Science, Columbia University, New York, NY 10027 (rocco@cs.columbia.edu). This author's research was supported in part by NSF CAREER award CCF-0347282 and a Sloan Foundation Fellowship.

distribution learning is to obtain an algorithm that can learn any Boolean function $f : \{-1,1\}^n \to \{-1,1\}$ in time polynomial in n and in a reasonable measure of the "size" or complexity of f. Different complexity measures for Boolean functions thus give rise to different notions of efficient learnability; for example, one might hope for an algorithm that can learn any Boolean function f in time polynomial in n and $DT(f)$ the number of leaves in the smallest Boolean decision tree that computes f (this is the well-studied—and notoriously difficult—problem of "learning decision trees under the uniform distribution"). A more ambitious goal would be to learn in time polynomial in $DNF(f)$, the number of terms in the smallest disjunctive normal form formula for f, or $AC^0_d(f)$, the size of the smallest depth-d AND/OR/NOT circuit for f.

Unfortunately, learning arbitrary Boolean functions in polynomial time in this sense has proved to be intractably difficult for all "reasonable" size measures. For the strongest reasonable size measure (Boolean circuit size), Valiant already observed in [31] that the existence of cryptographic pseudorandom functions [11] implies the nonexistence of uniform distribution algorithms that can learn any function f in time polynomial in the Boolean circuit size of f. This negative result was strengthened by Kharitonov [20], who showed that (under a strong but plausible assumption on the hardness of integer factorization) no uniform distribution algorithm can learn every f in time polynomial in $AC^0_d(f)$ for some fixed constant d. In fact, despite intensive research, no algorithm is currently known that learns arbitrary Boolean functions in time polynomial in *any* reasonable size measure; such an algorithm would constitute a tremendous breakthrough in computational learning theory; see, e.g., [2]. (We stress that simple arguments such as those in [5] show that there is no *information-theoretic* impediment to learning from a polynomial number of examples; the apparent difficulty is in designing a *polynomial-time* algorithm.)

1.2. Background: Learning monotone functions. Confronted with the difficulties described above, researchers have tried to learn various restricted classes of Boolean functions. The most natural and intensively studied such class is the class of all *monotone* functions $f : \{-1,1\}^n \to \{-1,1\}$, i.e., functions that satisfy $f(x) \geq f(y)$ whenever $x \geq y$ in the partial order on $\{-1,1\}^n$.

Many partial results on learning restricted subclasses of monotone functions under the uniform distribution have been obtained. Sakai and Maruoka [27] gave a poly(n)-time algorithm that can learn any monotone size-$O(\log n)$ disjunctive normal form (DNF) under the uniform distribution; this result was subsequently generalized by Bshouty [6] to a somewhat broader class than the $O(\log n)$-term DNF. The main result of Bshouty and Tamon in [7] is a proof that any monotone function can be learned to accuracy ϵ in $2^{\tilde{O}(\sqrt{n}/\epsilon)}$ time; they used this result to obtain a poly(n)-time algorithm (for ϵ constant) that can learn a class of functions that includes monotone $O(\log^2 n/(\log\log n)^3)$-term DNF. More recently, Servedio [30] showed that monotone $2^{O(\sqrt{\log n})}$-term DNF can be learned to constant accuracy ϵ in poly(n) time. Other researchers have also studied the problem of learning monotone functions under the uniform distribution (see, e.g., [18, 3, 34, 12, 21]), but prior to the current work no algorithms were known for learning arbitrary monotone functions in time polynomial in a reasonable size measure.

1.3. The main learning result. We give the first algorithm that learns any monotone Boolean function f, under the uniform distribution, in time polynomial in a reasonable measure of the size of f. Given a Boolean function $f : \{-1,1\}^n \to \{-1,1\}$, the *partition size* $P(f)$ of f is the minimum size partition of the Boolean cube $\{-1,1\}^n$

into disjoint subcubes such that f is constant on each subcube. Note that this is a strictly stronger measure of complexity than decision tree size; i.e., $P(f) \leq DT(f)$. Our main learning result is the following.

THEOREM 1. *There is an algorithm that (with confidence $1 - \delta$) can learn any monotone Boolean function $f : \{-1,1\}^n \to \{-1,1\}$ to accuracy ϵ, given uniform random examples $(x, f(x))$, in time poly$(n, P(f)^{1/\epsilon^2}) \cdot \log(1/\delta)$.*

For any constant accuracy $\epsilon = \Theta(1)$, the algorithm runs in time polynomial in the partition size of f and hence also in the decision tree size of f. We feel that this constitutes progress toward learning monotone functions in time polynomial in their DNF size, an open problem in computational learning theory (see, e.g., the open questions posed in [15, 3, 1]).

1.4. The approach: Bounding average sensitivity of monotone functions. The main ingredient of our learning algorithm is a new inequality bounding the average sensitivity (sum of influences of all coordinates) of monotone Boolean functions. We give here a simplified version of the theorem (the full result is given in Theorem 3).

THEOREM 2. *Every monotone Boolean function f has average sensitivity at most $\sqrt{\log P(f)}$.*[1]

This edge-isoperimetric-type result is of independent interest; indeed, our most general version of it, Theorem 7, recently played a critical role in a new lower bound on the randomized decision tree complexity of monotone graph properties—see [29].

Combining this new inequality with a result of Friedgut [8] that says that Boolean functions with low average sensitivity essentially depend on only a small number of coordinates, we can show that (i) there is a set of $P(f)^{O(1/\epsilon^2)}$ many Fourier coefficients of f which contain all but ϵ of the "Fourier weight" of f, and (ii) this set of Fourier coefficients can be efficiently identified from uniform random examples only. Applying standard machinery on approximating Boolean functions via their Fourier representations, we obtain Theorem 1.

Our approach seems quite robust. We generalize the basic scenario described above by (i) considering *real-valued* monotone functions that map $\{-1,1\}^n$ into the continuous interval $[-1,1]$ rather than the discrete range $\{-1,1\}$, and (ii) considering general p-biased product measures over $\{-1,1\}^n$ rather than the uniform distribution. We show that suitable variants of all of our intermediate results hold and that our main learning result holds exactly as before (i.e., runs in time $P(f)^{O(1/\epsilon^2)}$) in these generalized scenarios.

2. Preliminaries.

2.1. Boolean functions and complexity measures. As is standard in complexity theory and learning theory, we will be interested in complexity measures for Boolean functions f given by the syntactic size of the smallest representation of f under various natural representation schemes. We will chiefly be concerned with partition size and decision tree size, two complexity measures that we now define.

Given a Boolean function $f : \{-1,1\}^n \to \{-1,1\}$, the *decision tree size* of f, denoted $DT(f)$, is the number of leaves in the smallest Boolean decision tree (with variables $x_1, \dots, x_n$ at the internal nodes and bits -1, 1 at the leaves) that computes f. The *partition size* $P(f)$ of f is the minimum number of disjoint subcubes that the Boolean cube $\{-1,1\}^n$ can be partitioned into such that f is constant on each subcube.

[1] Here and throughout the paper "log" denotes logarithm to the base two.

Since any s-leaf decision tree induces a partition of $\{-1,1\}^n$ into s disjoint subcubes (corresponding to the root-to-leaf paths in the tree), we have that $P(f) \leq DT(f)$ for all f. In fact, $P(\cdot)$ is known to be a superpolynomially stronger measure than $DT(\cdot)$ even for monotone functions; Savický [28] has given a monotone Boolean function $g : \{-1,1\}^n \to \{-1,1\}$ which has $P(g) = \mathrm{poly}(n)$ and $DT(g) = 2^{\Omega(\log^{1.26}(n))}$.

2.2. Background: Uniform distribution learning. A *concept class* $\mathcal{F}$ is a collection $\cup_{n\geq 1}\mathcal{F}_n$ of Boolean functions where each $f \in \mathcal{F}_n$ is a function from $\{-1,1\}^n$ to $\{-1,1\}$. Throughout this paper we consider the concept class consisting of all monotone Boolean functions.

The uniform distribution probably approximately correct (PAC) learning model has been studied by many authors; see, e.g., [4, 7, 13, 14, 20, 22, 24, 27, 30, 33]. In this framework a learning algorithm has access to an *example oracle* $EX(f)$, where $f \in \mathcal{F}_n$ is the unknown *target function* the algorithm is trying to learn. The oracle $EX(f)$ takes no inputs and, when queried, in one time step outputs a labeled example $(x, f(x))$, where x is drawn from the uniform distribution $\mathcal{U}$ over $\{-1,1\}^n$.

We say that a Boolean function $h\colon \{-1,1\}^n \to \{-1,1\}$ is an *ϵ-approximator for f* if it satisfies $\Pr_{x\in\mathcal{U}}[h(x) = f(x)] \geq 1-\epsilon$. The goal of a uniform distribution PAC learning algorithm is to generate an ϵ-approximator for the unknown target function f. More precisely, an algorithm A is a *learning algorithm for concept class $\mathcal{F}$* if the following condition holds: for all $n \geq 1$, all $f \in \mathcal{F}_n$, and all $0 < \epsilon, \delta < 1$, if A is given ϵ and δ as input and has access to $EX(f)$, then with probability at least $1-\delta$ algorithm A outputs an ϵ-approximator for f. We further say that *A PAC learns $\mathcal{F}$ in time t* if A runs for at most t time steps and outputs a hypothesis h which can be evaluated on any point $x \in \{-1,1\}^n$ in time t. Here t will depend on the dimension n and the size s of f under some complexity measure, as well as on ϵ and δ. We note that for the learning algorithm presented and analyzed in this paper, the dominant ingredient in the running time t is collecting a sample of essentially t labeled examples using the oracle $EX(f)$; the computation that is performed after this sample has been obtained is relatively simple and inexpensive.

2.3. Fourier representation. Fourier techniques have proven to be a powerful tool for obtaining uniform distribution learning algorithms; see the survey of Mansour [23] for an overview.

Except in section 5, we will always view $\{-1,1\}^n$ as a probability space under the uniform distribution which we denote by $\mathcal{U}$. Let $f : \{-1,1\}^n \to \mathbf{R}$ be a real-valued function. Recall that the *Fourier expansion* of f is

$$f(x) = \sum_{S\subseteq[n]} \hat{f}(S)\chi_S(x),$$

where $\chi_S(x)$ denotes $\prod_{i\in S} x_i$ and $\hat{f}(S)$ denotes $\mathbf{E}_{x\in\mathcal{U}}[f(x)\chi_S(x)]$. It is well known that every f has a unique Fourier expansion. Parseval's theorem states that for any $f : \{-1,1\}^n \to \mathbf{R}$ we have $\sum_{S\subseteq[n]} \hat{f}(S)^2 = \mathbf{E}_{x\in\mathcal{U}}[f(x)^2]$, which is clearly 1 if f's range is $\{-1,1\}$.

For Boolean-valued functions $f\colon \{-1,1\}^n \to \{-1,1\}$, the *influence of coordinate i on f* is defined as $\mathrm{Inf}_i(f) = \Pr_{x\in\mathcal{U}}[f(x) \neq f(x^{(\oplus i)})]$, where $x^{(\oplus i)}$ denotes x with the ith bit flipped. In general we have $\mathrm{Inf}_i(f) = \sum_{S\ni i} \hat{f}(S)^2$; it is also well known (see, e.g., [17]) that if f is monotone then $\mathrm{Inf}_i(f) = \hat{f}(\{i\})$. For notational ease we will henceforth write $\hat{f}(i)$ in place of $\hat{f}(\{i\})$. The *average sensitivity* of a Boolean function

f is $\mathrm{I}(f) = \sum_{i=1}^n \mathrm{Inf}_i(f)$; this is the expected number of sensitive coordinates for a random input $x \in \{-1,1\}^n$. Note that $\mathrm{I}(f) = \sum_{i=1}^n \hat{f}(i)$ for monotone f.

3. The average sensitivity of monotone functions.

A well-known, folkloric edge-isoperimetric inequality for the Boolean cube states that for any monotone function $f\colon \{-1,1\}^n \to \{-1,1\}$, we have $\mathrm{I}(f) \le I(\mathsf{Maj}_n) = \Theta(\sqrt{n})$. (This follows from, e.g., the Kruskal–Katona theorem; see [9] for an explicit proof.) This bound $\mathrm{I}(f) \le O(\sqrt{n})$ is the key to the main result of [7] that any monotone Boolean function can be learned to accuracy ϵ in time $2^{\tilde{O}(\sqrt{n}/\epsilon)}$.

In this section we give a more refined bound on $\mathrm{I}(f)$ that depends on $P(f)$, the partition size of f. Our new bound states that $\mathrm{I}(f) \le \sqrt{\log P(f)}$ for any monotone f. This yields the usual isoperimetric inequality mentioned as a special case but is much stronger for functions f which have partition size $P(f) = 2^{o(n)}$.

3.1. Subcube partitions.

Let $f\colon \{-1,1\}^n \to \{-1,1\}$ be a Boolean function and let $\mathcal{C} = \{C^1, \dots, C^s\}$ be a subcube partition for f, so $C^1, \dots, C^s$ partition $\{-1,1\}^n$ into s subcubes on each of which f is constant. By abuse of notation we will also identify a cube C^t with a length-n vector over $\{-1,0,1\}$ in the obvious way; i.e., the ith coordinate of the string C^t is

$$(C^t)_i = \begin{cases} 1 & \text{if } x_i = 1 \text{ for all } x \in C^t, \\ -1 & \text{if } x_i = -1 \text{ for all } x \in C^t, \\ 0 & \text{otherwise.} \end{cases}$$

Let us also introduce notation for the sets of coordinates which cubes fix:

$$\mathrm{pluses}(C^t) = \{i : (C^t)_i = 1\}, \qquad \mathrm{minuses}(C^t) = \{i : (C^t)_i = -1\},$$

$$\mathrm{fixed}(C^t) = \mathrm{pluses}(C^t) \cup \mathrm{minuses}(C^t).$$

Given an input $x \in \{-1,1\}^n$, we write $C(x)$ to denote the subcube C^t in $\mathcal{C}$ to which x belongs. We also write δ_i to denote $\Pr_{x \in \mathcal{U}}[i \in \mathrm{fixed}(C(x))]$, the probability that the subcube partition "queries" x_i. Note that $\sum_{i=1}^n \delta_i$ equals $\mathbf{E}_{x \in \mathcal{U}}[|\mathrm{fixed}(C(x))|]$, the average number of coordinates $\mathcal{C}$ "queries."

When we draw $x \in \mathcal{U}$, this determines $C(x)$. However, we can equally well view the random determination of $(x, C(x))$ the other way around. Indeed, we will almost always consider choosing a uniformly random string x as follows:

1. Pick a random subcube R from $\mathcal{C}$ by choosing each C^t with probability $2^{-|\mathrm{fixed}(C^t)|}$. In general we will write $R \in \mathcal{C}$ to indicate that R is a random variable given by choosing a subcube from among $C^1, \dots, C^s$ according to this natural probability distribution on subcubes.
2. Now choose x uniformly at random from the strings in R. We will write $x \in R$ to indicate that x is chosen randomly in this way.

After this procedure, x indeed has the uniform distribution. Furthermore, note that the value $f(x)$ is determined as soon as R is chosen; thus we may abuse notation and write $f(R)$ for this quantity.

We will require the following very easy lemmas.

LEMMA 1. *Let R be any subcube and let $i \ne j$ be in $[n]$. Then $\mathbf{E}_{x \in R}[x_i] = R_i$ and $\mathbf{E}_{x \in R}[x_i x_j] = R_i R_j$.*

Proof. The proof is immediate from the definitions. □

LEMMA 2. *Let $i \neq j$ be in $[n]$. Then $\mathbf{E}_{R\in\mathcal{C}}[R_i] = 0$ and $\mathbf{E}_{R\in\mathcal{C}}[R_iR_j] = 0$.*

Proof. We prove the second statement, with the first being even easier:

$$0 = \mathop{\mathbf{E}}_{x\in\mathcal{U}}[x_ix_j] = \mathop{\mathbf{E}}_{R\in\mathcal{C}}\mathop{\mathbf{E}}_{x\in R}[x_ix_j] = \mathop{\mathbf{E}}_{R\in\mathcal{C}}[R_iR_j],$$

where in the last step we used Lemma 1. □

3.2. Proof of the main inequality. The proof requires one basic lemma.

LEMMA 3. *Let $f\colon \{-1,1\}^n \to \{-1,1\}$ be a Boolean function with a subcube partition $\mathcal{C} = \{C^1,\dots,C^s\}$. Then for each $i = 1,\dots,n$ we have $\hat{f}(i) = \mathbf{E}_{R\in\mathcal{C}}[f(R)R_i]$, and hence we have $\sum_{i=1}^n \hat{f}(i) = \mathbf{E}_{R\in\mathcal{C}}[f(R)\cdot\sum_{i=1}^n R_i]$.*

Proof. Fix any i in $\{1,\dots,n\}$. We have

$$\hat{f}(i) = \mathop{\mathbf{E}}_{x\in\mathcal{U}}[f(x)x_i] = \mathop{\mathbf{E}}_{R\in\mathcal{C}}\mathop{\mathbf{E}}_{x\in R}[f(x)x_i] = \mathop{\mathbf{E}}_{R\in\mathcal{C}}\left[f(R)\mathop{\mathbf{E}}_{x\in R}[x_i]\right] = \mathop{\mathbf{E}}_{R\in\mathcal{C}}[f(R)R_i],$$

where in the last step we used Lemma 1. □

With this lemma in hand we can give the proof that $\mathrm{I}(f) \le \sqrt{\log P(f)}$ for monotone f.

THEOREM 3. *Let $f\colon \{-1,1\}^n \to \{-1,1\}$ be a Boolean function with a subcube partition $\mathcal{C} = \{C^1,\dots,C^s\}$. Then we have*

$$\sum_{i=1}^n \hat{f}(i) \le \sqrt{\sum_{i=1}^n \delta_i} \le \sqrt{\log s},$$

and if f is monotone we may thus write $\mathrm{I}(f) \le \sqrt{\log s}$.

Proof. Since f is ± 1-valued, from Lemma 3 we have

$$\sum_{i=1}^n \hat{f}(i) \le \mathop{\mathbf{E}}_{R\in\mathcal{C}}\left[\left|\sum_{i=1}^n R_i\right|\right] \tag{1}$$

with equality iff $f(x) = \mathrm{sgn}(\sum_{i=1}^n C(x)_i)$ for all x, i.e., $f(x)$ is the majority of the bits that are set in $C(x)$. Applying Cauchy–Schwarz, we have

$$\begin{aligned}\mathop{\mathbf{E}}_{R\in\mathcal{C}}\left[\left|\sum_{i=1}^n R_i\right|\right] \le \sqrt{\mathop{\mathbf{E}}_{R\in\mathcal{C}}\left[\left(\sum_{i=1}^n R_i\right)^2\right]} &= \sqrt{\mathop{\mathbf{E}}_{R\in\mathcal{C}}\left[\sum_{i=1}^n R_i^2 + 2\sum_{i<j} R_iR_j\right]}\\ &= \sqrt{\mathop{\mathbf{E}}_{R\in\mathcal{C}}\left[|\mathrm{fixed}(R)|\right]} \qquad (2)\\ &= \sqrt{\mathop{\mathbf{E}}_{x\in\mathcal{U}}\left[|\mathrm{fixed}(C(x))|\right]} = \sqrt{\sum_{i=1}^n \delta_i},\end{aligned}$$

where (2) uses Lemma 2.

This proves the first inequality; to finish the proof we must show that $\sum_{i=1}^n \delta_i \le \log s$. We have

$$\sum_{i=1}^n \delta_i = \mathop{\mathbf{E}}_{R\in\mathcal{C}}[|\mathrm{fixed}(R)|] = \sum_{t=1}^s 2^{-|\mathrm{fixed}(C^t)|}\cdot|\mathrm{fixed}(C^t)| = H(R),$$

where $H(R)$ denotes the binary entropy of the random variable $R \in \mathcal{C}$. Since $\mathcal{C}$, the support of R, is of cardinality s, this entropy is at most $\log s$. □

Remarks.

1. We note that our proof can easily be used to recover the standard upper bound $\mathrm{I}(f) \le \mathrm{I}(\mathsf{Maj}_n)$ for any monotone Boolean function f on n variables. (Recall that $\mathrm{I}(\mathsf{Maj}_n) \sim \sqrt{\frac{2}{\pi}}\sqrt{n}$.) This is because in upper-bounding $\mathbf{E}_{R\in\mathcal{C}}[|\sum_{i=1}^{n} R_i|]$, we may assume without loss of generality that each subcube $C^t \in \mathcal{C}$ fixes exactly n bits. (To see this, suppose that C^t fixes $n' < n$ bits and we subdivide C^t into two subcubes each fixing one more bit. If $\sum_{i=1}^{n}(C^t)_i \neq 0$ then the contribution of C^t to $\mathbf{E}_{R\in\mathcal{C}}[|\sum_{i=1}^{n} R_i|]$ is unchanged by this subdivision, and if $\sum_{i=1}^{n}(C^t)_i = 0$ then the contribution increases.) But now observe that equality occurs in inequality (1), as noted above, if $f(x)$ always equals the majority of the bits set in $C(x)$, i.e., if $f(x) = \mathsf{Maj}_n(x)$ for all x.
2. The bound $\mathrm{I}(f) \le \sqrt{\log P(f)}$ need not hold for nonmonotone f; an easy example is the parity function on n variables for which $I(f) = \log P(f) = n$.

4. Learning monotone Boolean functions.

4.1. Spectral concentration. In this subsection we show that any monotone Boolean function has all but ϵ of its Fourier spectrum concentrated on a set of $P(f)^{O(1/\epsilon^2)}$ many Fourier coefficients.

In [8] Friedgut showed that any Boolean function with "low" average sensitivity is well approximated by a function that depends only on a "small" number of coordinates. In particular, the proof of Corollary 3.2 in [8] yields the following.

THEOREM 4. *There is a universal constant $K < \infty$ such that for all $f\colon \{-1,1\}^n \to \{-1,1\}$ and $\epsilon > 0$, if*

$$t = 2\mathrm{I}(f)/\epsilon, \quad J = \{i : \mathrm{Inf}_i(f) \ge K^{-t}\}, \quad \mathcal{S} = \{S : \ S \subseteq J, \ |S| \le t\},$$

then $\sum_{S\notin\mathcal{S}} \hat{f}(S)^2 \le \epsilon$.

Combining this result with Theorem 3, we obtain the following theorem.

THEOREM 5. *Let $f\colon \{-1,1\}^n \to \{-1,1\}$ be a monotone function, $\epsilon > 0$, and $t = 2\sqrt{\log P(f)}/\epsilon$. Let J and $\mathcal{S}$ be as in Theorem 4. Then $|\mathcal{S}| = P(f)^{O(1/\epsilon^2)}$ and $\sum_{S\notin\mathcal{S}} \hat{f}(S)^2 \le \epsilon$.*

Proof. The second part of the conclusion follows immediately from combining Theorems 3 and 4. As for bounding $|\mathcal{S}|$, we have $|\mathcal{S}| = \sum_{i=0}^{t}\binom{|J|}{i} \le O(|J|^t)$. But we also have $|J| \le \mathrm{I}(f)K^t \le tK^t$ using Theorem 3, and so $|J|^t \le 2^{O(t^2)} = P(f)^{O(1/\epsilon^2)}$, as claimed. □

4.2. Approximating Boolean functions with spectral concentration. The following proposition is a straightforward generalization of the "low-degree" algorithm of Linial, Mansour, and Nisan [22].

PROPOSITION 4. *There is an algorithm A with the following property: Let $f : \{-1,1\}^n \to [-1,1]$ and let $\mathcal{S} \subseteq 2^{[n]}$ be a collection of subsets of $[n]$ with the property that $\sum_{S\in\mathcal{S}} \hat{f}(S)^2 \ge 1-\epsilon$. Then if A is given $\mathcal{S}$, access to $EX(f)$, and parameters $\delta, \theta > 0$, it runs in poly$(n, |\mathcal{S}|, 1/\theta)\cdot\log(1/\delta)$ time and with probability $1-\delta$ outputs a real-valued function $g : \{-1,1\}^n \to \mathbf{R}$ of the form $g(x) = \sum_{S\in\mathcal{S}} c_S\chi_S(x)$ such that $\mathbf{E}_{x\in\mathcal{U}}[(f(x)-g(x))^2] \le \epsilon + \theta$.*

Proof sketch. Algorithm A draws a sample of m labeled examples from $EX(f)$ and uses them to empirically estimate each of the Fourier coefficients $\hat{f}(S)$ for $S \in \mathcal{S}$, using the fact that $\hat{f}(S) = E[f(x)\chi_S(x)]$; the coefficients c_S are the empirical estimates thus obtained. A standard analysis (see, e.g., Theorem 4.3 of [23]) shows that $m = \mathrm{poly}(|\mathcal{S}|, 1/\theta)\cdot\log(1/\delta)$ suffices to give the proposition. □

We remark that if $f : \{-1,1\}^n \to \{-1,1\}$ is Boolean-valued, and $g : \{-1,1\}^n \to \mathbf{R}$ satisfies $\mathbf{E}_{x\in\mathcal{U}}[(f(x) - g(x))^2] \le \epsilon'$, then defining $h : \{-1,1\}^n \to \{-1,1\}$ by $h(x) = \mathrm{sgn}(g(x))$, it is easily seen that $\Pr_{x\in\mathcal{U}}[h(x) \ne f(x)] \le \epsilon'$ (see, e.g., [22, 23]).

4.3. Learning monotone Boolean functions in polynomial time. We now give the proof of Theorem 1. Given Theorem 5 and Proposition 4, the idea behind our main learning algorithm is obvious: Given uniform examples from a target function f, identify all coordinates with influence at least $2^{-O(\sqrt{\log P(f)}/\epsilon)}$, and then run the algorithm from Proposition 4 using the set $\mathcal{S}$ from Theorem 5. (We note that a similar algorithm is used by Servedio in [30], though the analysis is completely different.)

By a standard doubling argument, we may assume the partition size $P(f)$ is known to the learner (see Exercise 1.5 of [19]). We now show that the learner can actually identify the sufficiently influential coordinates. This is because f is monotone, and consequently $\mathrm{Inf}_i(f) = \hat{f}(i) = \mathbf{E}_{x\in U}[f(x)x_i]$. Since the learner can empirically estimate this latter quantity to within $\pm\theta$ in time $\mathrm{poly}(n, 1/\theta) \cdot \log(1/\delta)$ (with confidence $1 - \delta$) by sampling, the learner can determine each influence $\mathrm{Inf}_i(f)$ of f to within an additive $2^{-O(\sqrt{\log P(f)}/\epsilon)}$ in $\mathrm{poly}(n, 2^{O(\sqrt{\log P(f)}/\epsilon)})$ time steps, and it is easy to see this is sufficient to maintain correctness and the same time bounds. Complete details can be found in a more general setting in Appendix B.

5. Generalizations: Real-valued functions and p-biased measures.

In this section we extend our learning result to real-valued functions $f : \{-1,1\}^n \to [-1,1]$ on the p-biased discrete cube. As in the Boolean case, we say a real-valued function f is monotone if $f(x) \ge f(y)$ whenever $x \ge y$. The partition size $P(f)$ of $f : \{-1,1\}^n \to [-1,1]$ is still defined as the minimum number of disjoint subcubes that $\{-1,1\}^n$ can be partitioned into such that f is constant on each subcube.

The p-biased measure on $\{-1,1\}^n$ is the probability distribution assigning probability $p^{|\mathrm{pluses}(x)|}q^{|\mathrm{minuses}(x)|}$ to the input $x \in \{-1,1\}^n$. (Here and throughout q denotes $1-p$.) We will write $\{-1,1\}^n_{(p)}$ to indicate that $\{-1,1\}^n$ is endowed with the p-biased measure and write $\Pr_p[\cdot]$ and $\mathbf{E}_p[\cdot]$ to denote probabilities and expectations over $x \in \{-1,1\}^n_{(p)}$.

We use standard notions of PAC learning for functions $f : \{-1,1\}^n_{(p)} \to [-1,1]$. This involves only slightly altering the definitions from section 2.2. Specifically, examples are now from the p-biased distribution $\{-1,1\}^n_{(p)}$ instead of the uniform distribution;[2] and, the definition of an ϵ-approximator is a function $h : \{-1,1\}^n_{(p)} \to \mathbf{R}$ satisfying $\mathbf{E}_p[(h - f)^2] \le \epsilon$ (note that we use the "square loss" as is common in learning or approximating real-valued functions). For other work studying PAC learning under the p-biased distribution, see, e.g., [10, 12, 25, 30].

Our main learning theorem completely extends to the p-biased, real-valued case, as follows.

THEOREM 6. *There is an algorithm that (with confidence $1 - \delta$) can learn any monotone Boolean function $f : \{-1,1\}^n_{(p)} \to [-1,1]$ to accuracy ϵ, given p-biased random examples $(x, f(x))$, in time poly$(n, P(f)^{1/\epsilon^2}) \cdot \log(1/\delta)$.*

Again, note that for any constant accuracy $\epsilon = \Theta(1)$, the algorithm runs in polynomial time in the partition size of f. Further note that unlike some p-biased PAC learning algorithms such as [10, 30], our algorithm's running time has no dependence

[2] There is a question as to whether or not the learning algorithm "knows" the value of p in advance. We show in Appendix B that we may assume without loss of generality that the learning algorithm knows p.

on p and thus we have the claimed runtime bound even if p depends on n or $P(f)$, such as $p = 1/\sqrt{n}$.

5.1. Background: Fourier analysis under *p*-biased measures. Given two functions $f, g : \{-1,1\}^n_{(p)} \to \mathbf{R}$, the p-biased inner product is defined as $\langle f, g\rangle_p = \mathbf{E}_p[f(x)g(x)]$. For $S \subseteq [n]$ the function $\phi_S(x) : \{-1,1\}^n_{(p)} \to \mathbf{R}$ is defined by

$$\phi_S(x) = \prod_{i\in S} \phi(x_i), \qquad \text{where } \phi(x_i) = \begin{cases} \sqrt{q/p} & \text{if } x_i = 1, \\ -\sqrt{p/q} & \text{if } x_i = -1. \end{cases}$$

The functions $\{\phi_S\}_{S\subseteq[n]}$ form an orthonormal basis with respect to $\langle \cdot, \cdot \rangle_p$. The *p-biased Fourier expansion* of $f : \{-1,1\}^n_{(p)} \to \mathbf{R}$ is $f(x) = \sum_{S\subseteq[n]} \tilde{f}(S)\phi_S(x)$, where $\tilde{f}(S) = \mathbf{E}_p[f(x)\phi_S(x)]$; note that we write $\tilde{f}$ rather than $\hat{f}$ to denote p-biased Fourier coefficients. Parseval's identity continues to hold: $\mathbf{E}_p[f^2] = \sum_S \tilde{f}(S)^2$.

We define the operator D_i on functions $f : \{-1,1\}^n_{(p)} \to \mathbf{R}$ by $(D_i f)(x) = \sqrt{pq}\ (f(x^{(i=1)}) - f(x^{(i=-1)}))$, where $x^{(i=b)}$ denotes x with the ith bit set to b. It is not difficult to verify that $(D_i f)(x) = \sum_{S\ni i} \tilde{f}(S)\phi_{S\setminus i}(x)$. We now give the definition of p-biased influence.

DEFINITION 1. *The* p-biased influence of the ith coordinate on $f : \{-1,1\}^n_{(p)} \to \mathbf{R}$ *is*

$$\mathrm{Inf}_i^{(p)}(f) = \mathbf{E}_p[(D_i f)^2] = \sum_{S\ni i} \tilde{f}(S)^2.$$

Note that if $f : \{-1,1\}^n_{(p)} \to \{-1,1\}$, *then* $\mathrm{Inf}_i^{(p)}(f) = 4pq \Pr_p[f(x) \neq f(x^{\oplus i})]$.

We remark that this definition differs from the ones in [8, 9] by a multiplicative factor of $4pq$. We define the *p-biased average sensitivity* to be $\mathrm{I}^{(p)}(f) = \sum_{i=1}^n \mathrm{Inf}_i^{(p)}(f) = \sum_{S\subseteq[n]} |S|\tilde{f}(S)^2$. Note that in the case when $p = 1/2$ and f's range is $\{-1,1\}$, these definitions agree with the standard uniform distribution definitions from section 2.3.

We conclude this section with a useful relationship in the p-biased case between influences of monotone real-valued functions and singleton Fourier coefficients.

FACT 5. *For any monotone* $f : \{-1,1\}^n \to [-1,1]$ *we have* $\mathrm{Inf}_i^{(p)}(f) \leq 2\sqrt{pq}\cdot\tilde{f}(i)$, *with equality iff the range of* f *is* $\{-1,1\}$.

Proof. We have $\mathrm{Inf}_i^{(p)}(f) = \mathbf{E}_p[(D_i f)^2]$. Since f is monotone and has range $[-1,1]$ it is easy to see that $0 \leq (D_i f)(x) \leq 2\sqrt{pq}$ for all x. Thus $(D_i f)^2 \leq 2\sqrt{pq} \cdot (D_i f)$ with equality iff f's range is $\{-1,1\}$, and hence $\mathrm{Inf}_i^{(p)}(f) \leq 2\sqrt{pq} \cdot \mathbf{E}_p[D_i f] = 2\sqrt{pq} \cdot \tilde{f}(i)$. ☐

5.2. Bounding influence in monotone real-valued functions under *p*-biased measures. In this section we describe our analogue of Theorem 3 for real functions under p-biased measures. We first set up some p-biased preliminaries before proving the theorem. Let $\mathcal{C} = \{C^1, \ldots, C^s\}$ be a subcube partition of $\{-1,1\}^n$. We now identify the C^t's with length-n vectors in a way compatible with the ϕ-basis, i.e.,

$$(C^t)_i = \begin{cases} \phi(1) = \sqrt{q/p} & \text{if } x_i = 1 \text{ for all } x \in C^t, \\ \phi(-1) = -\sqrt{p/q} & \text{if } x_i = -1 \text{ for all } x \in C^t, \\ 0 & \text{otherwise.} \end{cases}$$

The definitions of pluses(C^t), minuses(C^t), and fixed(C^t) are as before. We now define $\delta_i^{(p)}$ to be the p-biased version of δ_i:

$$\delta_i^{(p)} = \Pr_{x \in \{-1,1\}^n_{(p)}} [i \in \text{fixed}(C(x))].$$

The observation that choosing $x \in \{-1,1\}^n_{(p)}$ and considering $(x, C(x))$ can be viewed as choosing $R \in \mathcal{C}$ and then $x \in R$ still holds with the obvious p-biased interpretation. Specifically, the random choice $R \in \mathcal{C}$ means selecting the cube C^t with probability $p^{|\text{pluses}(C^t)|} q^{|\text{minuses}(C^t)|}$; then the choice $x \in R$ means picking the unfixed coordinates according to the p-biased distribution.

The analogue of Lemma 2 and the analogue of Lemma 3 (for functions $f : \{-1,1\}^n_{(p)} \to \mathbf{R}$) now hold with no changes in the statements. To prove them we simply repeat their proofs and also the statement and proof of Lemma 1, everywhere replacing x_i and x_j with $\phi(x_i)$ and $\phi(x_j)$. As a consequence, we have the following additional lemma.

Lemma 6. *Given $\alpha, \beta \in \mathbf{R}$, the quantity $\mathbf{E}_{R \in \mathcal{C}}[\alpha \cdot |\text{pluses}(R)| + \beta \cdot |\text{minuses}(R)|]$ depends only on $p\alpha + q\beta$.*

Proof. Sum the first statement of the p-biased analogue of Lemma 2 over all i, and then expand the definition of R_i; one gets

$$\mathop{\mathbf{E}}_{R \in \mathcal{C}}\Big[\sqrt{q/p} \cdot |\text{pluses}(R)| - \sqrt{p/q} \cdot |\text{minuses}(R)|\Big] = 0$$

$$\Rightarrow \mathop{\mathbf{E}}_{R \in \mathcal{C}}[|\text{minuses}(R)|] = (q/p) \cdot \mathop{\mathbf{E}}_{R \in \mathcal{C}}[|\text{pluses}(R)|]. \tag{3}$$

So substituting this in, we get

$$\begin{aligned}\mathop{\mathbf{E}}_{R \in \mathcal{C}}[\alpha \cdot |\text{pluses}(R)| + \beta \cdot |\text{minuses}(R)|] &= \mathop{\mathbf{E}}_{R \in \mathcal{C}}[\alpha \cdot |\text{pluses}(R)| + (q/p)\beta \cdot |\text{pluses}(R)|] \\ &= (1/p) \mathop{\mathbf{E}}_{R \in \mathcal{C}}[(p\alpha + q\beta) \cdot |\text{pluses}(R)|],\end{aligned}$$

completing the proof. □

With this preparation in hand, we now give our p-biased, real-valued generalization of Theorem 3.

Theorem 7. *Let $f : \{-1,1\}^n_{(p)} \to \mathbf{R}$ be a function with subcube partition $\mathcal{C} = \{C^1, \dots, C^s\}$. Then we have*

$$\sum_{i=1}^n \tilde{f}(i) \le \|f\|_2 \cdot \sqrt{\sum_{i=1}^n \delta_i^{(p)}} \le \|f\|_2 \cdot \sqrt{\log s} / \sqrt{H(p)},$$

where $H(p) = p\log(1/p) + q\log(1/q)$. If $f : \{-1,1\}^n_{(p)} \to [-1,1]$ is monotone then by Fact 5 we may write $\mathrm{I}^{(p)}(f) \le \sqrt{4pq/H(p)}\sqrt{\log s}$.

Proof. Applying Cauchy–Schwarz directly to the analogue of Lemma 3, we have

$$\sum_{i=1}^n \tilde{f}(i) \le \sqrt{\mathop{\mathbf{E}}_{R \in \mathcal{C}}\Big[f(R)^2\Big]} \cdot \sqrt{\mathop{\mathbf{E}}_{R \in \mathcal{C}}\Big[\Big(\sum_{i=1}^n R_i\Big)^2\Big]} = \|f\|_2 \cdot \sqrt{\mathop{\mathbf{E}}_{R \in \mathcal{C}}\Big[\sum_{i=1}^n R_i^2\Big]},$$

where we used the p-biased analogue of the second statement of Lemma 2 in the equality, just as in the proof of Theorem 3. Let us now consider the quantity inside the square root. By definition,

$$\mathop{\mathbf{E}}_{R\in\mathcal{C}}\Big[\sum_{i=1}^{n} R_i^2\Big] = \mathop{\mathbf{E}}_{R\in\mathcal{C}}\Big[(q/p)\cdot \mathrm{pluses}(R) + (p/q)\cdot \mathrm{minuses}(R)\Big]. \tag{4}$$

Using (3) twice, we have

$$\begin{aligned}\mathop{\mathbf{E}}_{R\in\mathcal{C}}\Big[(q/p)\cdot \mathrm{pluses}(R) + (p/q)\cdot \mathrm{minuses}(R)\Big] &= \mathop{\mathbf{E}}_{R\in\mathcal{C}}[\mathrm{pluses}(R) + \mathrm{minuses}(R)]\\ &= \mathop{\mathbf{E}}_{R\in\mathcal{C}}[|\mathrm{fixed}(R)|] = \sum_{i=1}^{n} \delta_i^{(p)},\end{aligned}$$

completing the proof of the first inequality. As for the second inequality, note that the binary entropy $H(R)$ of the random variable $R \in \mathcal{C}$ is

$$\begin{aligned}H(R) &= \mathop{\mathbf{E}}_{R\in\mathcal{C}}\Big[\log(1/\Pr[R])\Big]\\ &= \mathop{\mathbf{E}}_{R\in\mathcal{C}}\Big[\log(1/p)\cdot \mathrm{pluses}(R) + \log(1/q)\cdot \mathrm{minuses}(R)\Big]\\ &= H(p)\cdot \mathop{\mathbf{E}}_{R\in\mathcal{C}}\left[\frac{\log(1/p)}{H(p)}\cdot \mathrm{pluses}(R) + \frac{\log(1/q)}{H(p)}\cdot \mathrm{minuses}(R)\right].\end{aligned}$$

But since $p\frac{\log(1/p)}{H(p)} + q\frac{\log(1/q)}{H(p)} = 1$ as well, applying Lemma 6 again yields

$$(4) = H(R)/H(p).$$

But $H(R) \le \log s$ as observed in the proof of Theorem 3, and the proof is complete. □

Using the bound $pq\log(1/pq) \le H(p)$, we have the following corollary.

COROLLARY 7. *If* $f : \{-1,1\}^n_{(p)} \to [-1,1]$ *is monotone then* $\mathrm{I}^{(p)}(f) \le 2\sqrt{\log P(f)}/\sqrt{\log(1/pq)}$.

5.3. Spectral concentration under p-biased measures. We now need to extend Friedgut's result to the p-biased, real-valued case. There are some difficulties involved. In [8], Friedgut gave a p-biased version of Theorem 4; however, he left the quantitative details of the dependence on p unspecified. More seriously, Friedgut's theorem is simply not true for $[-1,1]$-valued functions, even in the $p = 1/2$ case. (See Appendix A for an example demonstrating this.)

However, we are able to circumvent this problem. The necessary insight is the following: A real-valued function with small average sensitivity depends on only a small number of coordinates *if its range is sufficiently "discrete."* And for the purposes of learning an unknown function to some prescribed accuracy, we do not lose much by "rounding" the function's values to a discrete range.

For $\gamma > 0$, let $\gamma\mathbf{Z}$ denote the set of real numbers of the form γm, where m is an integer. By making some small changes to Friedgut's proof we can derive the following result (the proof is in Appendix A).

THEOREM 8. *There is a universal constant $K < \infty$ such that for all $0 < \epsilon, \gamma < 1/2$ and all $f : \{-1,1\}^n_{(p)} \to [-1,1] \cap (\gamma\mathbf{Z})$, if*

$$t = 2\mathrm{I}^{(p)}(f)/\epsilon, \quad \tau = \gamma^K (pq)^{Kt}, \quad J = \{i : \mathrm{Inf}_i^{(p)}(f) \geq \tau\}, \quad \mathcal{S} = \{S : S \subseteq J, |S| \leq t\},$$

then $\sum_{S \notin \mathcal{S}} \tilde{f}(S)^2 \leq \epsilon$.

We now combine Theorem 8 with Corollary 7, exactly in the manner of Theorem 5. The $\sqrt{\log(1/pq)}$ saved in Corollary 7 cancels with the pq paid in the τ from Theorem 8, and the factor of $\gamma^{O(1)}$ becomes negligible if we take $\gamma = \epsilon$ (indeed, even $\gamma = 2^{-O(1/\epsilon)}$ would be negligible). We get the following theorem.

THEOREM 9. *Let $\epsilon > 0$, $f : \{-1,1\}^n_{(p)} \to [-1,1] \cap (\epsilon\mathbf{Z})$ be a monotone function, and let $t = 4\sqrt{\log P(f)}/(\epsilon\sqrt{\log(1/pq)})$. Let $J = \{i : \mathrm{Inf}_i(f) \geq (K')^{-t\log(1/pq)}\}$, where $K' < \infty$ is a universal constant, and let $\mathcal{S} = \{S : S \subseteq J, |S| \leq t\}$. Then $|\mathcal{S}| = P(f)^{O(1/\epsilon^2)}$ and $\sum_{S \notin \mathcal{S}} \tilde{f}(S)^2 \leq \epsilon$.*

5.4. Learning monotone real-valued functions under *p*-biased measures. With Theorem 9 in hand, the proof of our main learning result Theorem 6 is now not very difficult. Given an unknown target function $f : \{-1,1\}^n_{(p)} \to [-1,1]$ and $\epsilon > 0$, let f_ϵ denote f with its values "rounded" to the nearest integer multiples of ϵ. Clearly, given examples from $EX(f,p)$, we can simulate examples from $EX(f_\epsilon, p)$. We now simply try to learn f_ϵ. It is easy to check that an ϵ-approximator hypothesis for f_ϵ is also an $O(\epsilon)$-approximator for f. Further, we have $P(f_\epsilon) \leq P(f)$ so a $P(f_\epsilon)^{O(1/\epsilon^2)}$ runtime is also $P(f)^{O(1/\epsilon^2)}$ as desired. The p-biased analogue of Proposition 4 holds with essentially the same proof. The only new difficulty is that we cannot exactly estimate the quantities $\mathrm{Inf}_i(f_\epsilon)$. However from Fact 5, the quantities $\tilde{f}(i)$—which we can estimate empirically—are upper bounds on the influences; so by taking all the coordinates i with $\tilde{f}(i) \geq \tau$, we get all the sufficiently influential coordinates. There cannot be too many coordinates with large $\tilde{f}(i)$, since $\sum_{i=1}^n \tilde{f}(i)^2 \leq 1$.

For completeness, we give all the details of the proof of Theorem 6 in Appendix B.

6. Extension to stronger complexity measures? It is natural to wonder whether our results can be extended to stronger complexity measures than decision tree size and partition size. An obvious next complexity measure to consider is the minimum number of (not necessarily disjoint) subcubes that cover $\{-1,1\}^n$ and are such that f is constant on each subcube. We refer to this as the *subcube covering complexity* of f and denote it by $CDNF(f)$, since it is equal to the minimum number of terms in any DNF formula for f plus the minimum number of clauses in any CNF formula for f.

The following theorem shows that Theorem 3 does not hold for subcube covering complexity.

THEOREM 10. *There is a monotone Boolean function $g : \{-1,1\}^n \to \{-1,1\}$ for which $\mathrm{I}(g) = \Omega(n^{\log_4(6-2\sqrt{5})}) = \Omega(n^{0.305})$ but $\sqrt{\log CDNF(g)} = O(n^{1/4})$.*

The proof is by the probabilistic method. We define a distribution $\mathcal{D}$ over monotone Boolean functions and show that some function g that is assigned nonzero weight under $\mathcal{D}$ must satisfy the bounds of the theorem. See Appendix C.

7. Conclusion. In this paper we established a new bound on average sensitivity of monotone functions and used this bound to give the first algorithm that uses random examples to learn any monotone function to high accuracy in time polynomial in the function's decision tree or partition size.

A natural goal for future work is to obtain even stronger learning results for monotone functions. Can the boosting methods used by Jackson in his harmonic sieve algorithm [13] be applied here? We note that while the harmonic sieve algorithm makes essential use of membership queries, related algorithms that combine boosting with Fourier techniques have been successfully developed for the framework of learning from random examples only [14].

Appendix A. Proof of Theorem 8 and a counterexample.

Proof of Theorem 8. Recall that we have

$$D_i f(x) = \sqrt{pq}(f(x^{(i=1)}) - f(x^{(i=-1)})) = \sum_{S:\ i \in S} \tilde{f}(S)\phi_{S\setminus i}(x)$$

and that $\mathrm{Inf}_i^{(p)}(f) = \mathbf{E}_p[(D_i f)^2] = \|D_i f\|_2^2$, where throughout this section $\|\cdot\|$ denotes the norm induced by the p-biased measure.

Since $\mathrm{I}^{(p)}(f) = \sum_S |S|\tilde{f}(S)^2$, Markov's inequality immediately gives that $\sum_{S:\ |S|>t} \tilde{f}(S)^2 < \epsilon/2$. Let $J' = [n] \setminus J$. It now suffices to show that

$$\sum_{S:\ S\cap J' \neq \emptyset, |S| \le t} \tilde{f}(S)^2 \le \epsilon/2. \tag{5}$$

Certainly the left side of (5) is at most

$$\sum_{i \in J'} \sum_{S:\ i \in S, |S| \le t} \tilde{f}(S)^2 = \sum_{i \in J'} \|D_i f^{\le t}\|_2^2 = \sum_{i \in J'} \langle D_i f^{\le t}, D_i f\rangle, \tag{6}$$

where we use the notation $f^{\le t}$ to denote the function $f^{\le t}(x) = \sum_{|S| \le t} \tilde{f}(S)\phi_S$. Now we have

$$\langle D_i f^{\le t}, D_i \rangle \le \|D_i f^{\le t}\|_4 \, \|D_i f\|_{4/3} \tag{7}$$

$$\le (1 + 1/\sqrt{pq})^{t/2} \, \|D_i f^{\le t}\|_2 \, \|D_i f\|_{4/3} \tag{8}$$

$$\le (1/pq)^t \, \mathrm{Inf}_i^{(p)}(f)^{1/2} \, \mathbf{E}[|D_i f|^{4/3}]^{3/4}. \tag{9}$$

Here (7) is Hölder's inequality, inequality (8) follows from a p-biased version of Bonami–Beckner (here with the best bounds provided by [26]), and inequality (9) uses the generous bound $(1 + 1/\sqrt{pq})^{1/2} < (1/pq)$ and also $\|D_i f^{\le t}\|_2 \le \|D_i f\|_2 = \mathrm{Inf}_i^{(p)}(f)^{1/2}$.

We now observe that by virtue of the assumption that f's range is contained in $\gamma\mathbf{Z}$, we have that $|D_i f(x)|$ is always either 0 or at least $\gamma\sqrt{pq}$. This implies that (9) is at most

$$(1/pq)^t \, \mathrm{Inf}_i^{(p)}(f)^{1/2} \, \mathbf{E}_p[(\gamma\sqrt{pq})^{-2/3} \cdot |D_i f|^2]^{3/4} = (1/pq)^{t+1/4} \, \gamma^{-1/2} \, \mathrm{Inf}_i^{(p)}(f)^{5/4}$$

$$\le (1/pq)^{t+1/4} \, \gamma^{-1/2} \, \mathrm{Inf}_i^{(p)}(f) \, \tau^{1/4}, \tag{10}$$

where we have used the definitions of $\mathrm{Inf}_i^{(p)}(f)$ and τ. Using the fact that $\sum_{i \in J'} \mathrm{Inf}_i^{(p)}(f) \le \mathrm{I}^{(p)}(f)$, we can sum (10) and conclude that (6) is at most

$$(1/pq)^{t+1/4} \, \gamma^{-1/2} \, \mathrm{I}^{(p)}(f) \, \tau^{1/4}.$$

Thus to ensure (5) holds we need only

$$t(1/pq)^{t+1/4}\,\gamma^{-1/2}\,\tau^{1/4} \leq 1;$$

upper-bounding $t(1/pq)^{t+1/4}$ by $(1/pq)^{O(t)}$ (acceptable for all $t \geq 0$), we see that $\tau = \gamma^{O(1)}(pq)^{O(t)}$ suffices. Thus the choice of τ given in the definition of Theorem 8 suffices and the proof of this theorem is complete. □

We now justify the remark from section 5.3 indicating that Friedgut's theorem does not in general hold for real-valued functions; in other words, the condition that f's range is contained in $\gamma\mathbf{Z}$ cannot be removed.

To see this, consider (in the uniform measure case) the function $f : \{-1,1\}^n \to [-1,1]$ defined by

$$f(x) = \begin{cases} \operatorname{sgn}(\sum_{i=1}^n x_i) & \text{if } |\sum_{i=1}^n x_i| > \sqrt{n}, \\ \frac{1}{\sqrt{n}}\sum_{i=1}^n x_i & \text{if } |\sum_{i=1}^n x_i| \leq \sqrt{n}. \end{cases}$$

It is easy to see that for each $i = 1,\dots,n$, $D_i f(x)$ is always either 0 or $1/\sqrt{n}$ and is $1/\sqrt{n}$ for a $\Theta(1)$ fraction of all x's. Consequently we have $\mathrm{Inf}_i(f) = \Theta(1/n)$ and thus $\mathrm{I}(f) = \Theta(1)$. In addition, it is clear that both $\mathbf{E}[f(x)] = 0$ and $|f(x)| \geq 1/2$ for a $\Theta(1)$ fraction of all x's; hence we have $\sum_{|S|>0}\hat{f}(S)^2 \geq \Omega(1)$. But now if we take ϵ to be any constant smaller than this $\Omega(1)$, then we get a contradiction, since the choice of τ in Theorem 8 will be a constant, and so J and hence $\mathcal{S}$ will be empty (for all n sufficiently large).

Appendix B. Technical details for learning. We begin with some basic learning details for the p-biased measure. First, as mentioned earlier, we may assume without loss of generality that the learning algorithm "knows" p. The proof is quite similar to the proof that a noise-tolerant learning algorithm can be assumed to know the exact noise rate (see [19]). The basic idea is that we can run the learning algorithm repeatedly using successively finer estimates (easily obtained from sampling) for the value of p. If the original algorithm runs for T time steps, then if the guessed value for p is within Δ/T of the true value, the statistical distance between the algorithm's output when run with the guessed value versus the true value will be at most Δ. It can be shown that at most a polynomial factor runtime overhead is incurred in coming up with a sufficiently accurate guess.

Next, we remark that low-degree algorithm of Linial, Mansour, and Nisan, Proposition 4, easily carries over to the real-valued p-biased case with essentially the same proof.

PROPOSITION 8. *There is an algorithm A with the following property: Let $f : \{-1,1\}^n_{(p)} \to [-1,1]$ and let $\mathcal{S} \subseteq 2^{[n]}$ be a collection of subsets of $[n]$ with the property that $\sum_{S:S\notin\mathcal{S}}\tilde{f}(S)^2 \leq \epsilon$. Then if A is given p, $\mathcal{S}$, access to a source $EX(f,p)$ of p-biased random examples, and parameters $\delta, \theta > 0$, it runs in poly$(n, |\mathcal{S}|, 1/\theta)\cdot\log(1/\delta)$ time and with probability $1-\delta$ outputs a real-valued function $g : \{-1,1\}^n \to \mathbf{R}$ of the form $g(x) = \sum_{S\in\mathcal{S}} c_S\phi_S(x)$ such that $\mathbf{E}_p[(f-g)^2] \leq \epsilon + \tau$.*

We now proceed to discuss the proof of Theorem 6. Let $f : \{-1,1\}^n_{(p)} \to [-1,1]$ be the target function. Given $\epsilon > 0$, let f_ϵ denote the "rounded" version of f in which each of its values is rounded to the nearest integer multiple of ϵ. It is clear that given access to $EX(p,f)$ we can simulate access to $EX(p,f_\epsilon)$. Our algorithm will use $EX(p,f_\epsilon)$ to learn f_ϵ in time $\mathrm{poly}(n, P(f_\epsilon)^{O(1/\epsilon^2)})\cdot\log(1/\delta)$. This is sufficient for learning f in the same time bound, because $P(f_\epsilon) \leq P(f)$ and because if $\mathbf{E}_p[(h - f_\epsilon)^2] \leq \epsilon$ then

$$\mathbf{E}_p[(h-f)^2] = \mathbf{E}_p[((h-f_\epsilon)+(f_\epsilon - f))^2] \leq 2\mathbf{E}_p[(h-f_\epsilon)^2] + 2\mathbf{E}_p[(f_\epsilon - f)^2]$$
$$\leq 2\epsilon + \epsilon^2/2 = O(\epsilon).$$

Our goal is now essentially to use Proposition 8 given Theorem 9. As mentioned in section 5.4, unlike in the algorithm for Boolean-valued functions, we cannot estimate the influences of f_ϵ directly since the relationship $\mathrm{Inf}_i^{(p)}(f_\epsilon) = \tilde{f}_\epsilon(i)$ does not hold in the real-valued case. We may, however, use Fact 5 which says that $\tilde{f}_\epsilon(i)$—a quantity we can empirically estimate—is an upper bound on $\mathrm{Inf}_i^{(p)}(f_\epsilon)$.

We now describe the algorithm to learn f_ϵ using $EX(p, f_\epsilon)$. As in section 4.3 we may assume that the partition size $P(f_\epsilon)$ is known. The algorithm is as follows:

1. For $i = 1, \ldots, n$ empirically estimate $\tilde{f}_\epsilon(i) = \mathbf{E}_p[f_\epsilon(x)\phi_i(x)]$ to within an additive $\pm\tau/4$ (with confidence $1-\delta$), where $\tau = (C')^{-t\log(1/pq)}$ and t is defined in Theorem 9. Let $J \subseteq [n]$ be the set of those i for which the obtained estimate is greater than $\tau/2$.
2. Now run algorithm A from Proposition 8 with $\mathcal{S} = \{S : \ S \subseteq J, \ |S| \leq t\}$ and $\theta = \epsilon$, outputting its hypothesis g.

Let us first confirm the running time of this algorithm. In step 1, standard sampling bounds ensure that $\mathrm{poly}(n, 1/\tau) \cdot \log(1/\delta)$ samples suffice. We may then conclude that $|J| \leq O(1/\tau^2)$, since $\sum_{i=1}^n \tilde{f}_\epsilon(S)^2 \leq 1$. It follows that $|\mathcal{S}| \leq \mathrm{poly}(1/\tau^t) = P(f_\epsilon)^{O(1/\epsilon^2)}$, as necessary to bound the running time. Finally, we still have $\sum_{S \notin \mathcal{S}} \tilde{f}_\epsilon(S)^2 \leq \epsilon$ because (with confidence $1-\delta$) the J the algorithm finds is a superset of the J from Theorem 9. Hence the algorithm correctly gives an $O(\epsilon)$-approximator hypothesis g with confidence $1 - O(\delta)$, and the proof of Theorem 6 is complete.

Appendix C. Proof of Theorem 10. Let $n = 4^k$. Let $f_1(a,b,c,d)$ be the "AND/OR" function on four Boolean variables $f_1(a,b,c,d) = (a \wedge b) \vee (c \wedge d)$. An important property of f_1 is that if each of its four arguments is independently set to be 1 (true) with probability p, then $\Pr[f_1 = 1]$ equals $2p^2 - p^4$. For $i = 2, 3, \ldots$, we define the function f_i on 4^i variables to be $f_i = f_1(f_{i-1}^1, f_{i-1}^2, f_{i-1}^3, f_{i-1}^4)$, where the superscripts indicate distinct copies of f_{i-1} on disjoint sets of variables. Thus f_k is a function on n variables computed by a read-once Boolean formula that is a tree of ANDs and ORs at alternating levels.

We now define distributions $\mathcal{D}_1, \ldots, \mathcal{D}_k$ over monotone Boolean functions, where $\mathcal{D}_i$ is a distribution over functions from $\{-1,1\}^{4^i}$ to $\{-1,1\}$. The distribution $\mathcal{D}_i$ is defined in the following way: a random draw from $\mathcal{D}_i$ is obtained by independently substituting 1 for each of the 4^i Boolean arguments to f_i with probability α, where $\alpha = \sqrt{5} - 2 \approx 0.236$. (This construction and some of the subsequent analysis are reminiscent of [32].) Note that for a random g drawn from $\mathcal{D}_1$ and a random x drawn uniformly from $\{-1,1\}^4$, we have that each of the four arguments to f_1 is independently 1 with probability $\frac{1}{2} + \frac{\alpha}{2} = \frac{\sqrt{5}-1}{2}$; we denote this value by ρ. Consequently we have $\Pr_{g \in \mathcal{D}_1, x \in \{-1,1\}^4}[g(x) = 1] = 2\rho^2 - \rho^4$, but this is easily seen to equal ρ. It follows from the recursive definition of f_i that for all $i = 1, 2, \ldots$ we have $\Pr_{g \in \mathcal{D}_i, x \in \{-1,1\}^{4^i}}[g(x) = 1] = \rho$.

It is not difficult to show (see Theorem 2.4 of [16]) that $CDNF(f_k) \leq 2^{2^k+1}$; as an immediate consequence we have that $CDNF(g) \leq 2^{2^k+1}$ (and thus $\sqrt{\log CDNF(g)} = O(2^{k/2}) = O(n^{1/4})$) for every g that is in the support of $\mathcal{D}_k$. But by Lemma 9 below we have that $\mathbf{E}_{g\in\mathcal{D}_k}[\mathrm{I}(g)] = \Theta((6-2\sqrt{5})^k)$; clearly this implies that there is some g in the support of $\mathcal{D}_k$ for which $\mathrm{I}(g)$ is $\Omega((6-2\sqrt{5})^k) = \Omega(n^{\log_4(6-2\sqrt{5})})$. This proves Theorem 10.

LEMMA 9. *For $i = 1, 2, \ldots$ we have $\mathbf{E}_{g\in\mathcal{D}_i}[\mathrm{I}(g)] = (3-\sqrt{5})(6-2\sqrt{5})^i$.*

Proof. It is clear from symmetry that $\mathbf{E}_{g\in\mathcal{D}_i}[\mathrm{I}(g)] = 4^i \cdot \mathbf{E}_{g\in\mathcal{D}_i}[\mathrm{Inf}_1(g)]$. We have

$$\mathbf{E}_{g\in\mathcal{D}_i}[\mathrm{Inf}_1(g)] = \mathbf{E}_{g\in\mathcal{D}_i, x\in\{-1,1\}^{4^i}}[\Pr[g(1,x_2,\ldots,x_{4^i}) \neq g(-1,x_2,\ldots,x_{4^i})]]$$

$$= \Pr_{g\in\mathcal{D}_i, x\in\{-1,1\}^{4^i}}[g(1,x_2,\ldots,x_{4^i}) \neq g(-1,x_2,\ldots,x_{4^i})].$$

From the definition of $\mathcal{D}_i$, we have that with probability $\alpha = \sqrt{5}-2$ the constant 1 is substituted for the first argument of f_i in g; if this occurs, then clearly $g(1,x_2,\ldots,x_{4^i}) = g(-1,x_2,\ldots,x_{4^i})$ for all x since g does not depend on its first argument. If this does not occur, then we have (for a random $g \in \mathcal{D}_i$ and a uniform $x \in \{-1,1\}^{4^i}$) that each of the other 4^{i-1} arguments to f_i independently takes value 1 with probability $\rho = \frac{\sqrt{5}-1}{2}$.

Under the distribution on inputs to f_i described in the previous paragraph, if $i = 1$ it is easy to see that flipping the first argument of f_1 flips the value of f_1 iff the second argument is 1 (probability ρ) and the AND of the third and fourth arguments is 0 (probability $1-\rho^2$). Thus flipping the first argument of f_1 flips the value of f_1 with probability precisely $\rho(1-\rho^2)$ which is easily seen to equal $1-\rho$, using the fact that $2\rho^2-\rho^4 = \rho$. Similarly, if $i = 2$, then flipping the first of the 16 arguments to $f_2 = f_1(f_1^1, f_1^2, f_1^3, f_1^4)$ (again under the distribution of inputs to f_i described above) will flip the value of f_2 iff the value of f_1^1 flips (probability $1-\rho$ as shown above), f_1^2 equals 1 (probability ρ), and $f_1^3 \wedge f_1^4$ equals 0 (probability $1-\rho^2$). We thus have that flipping the first argument of f_2 flips the value of f_2 with probability $(1-\rho)\rho(1-\rho^2) = (1-\rho)^2$. An easy induction in this fashion shows that for all i, under the distribution of inputs described above flipping the first argument of f_i causes f_i to flip with probability $(1-\rho)^i$.

We thus have that

$$\Pr_{g\in\mathcal{D}_i, x\in\{-1,1\}^{4^i}}[g(1,x_2,\ldots,x_{4^i}) \neq g(-1,x_2,\ldots,x_{4^i})] = (1-\alpha)(1-\rho)^i$$

$$= (3-\sqrt{5})\left(\frac{3-\sqrt{5}}{2}\right)^i,$$

which proves the lemma. □

Acknowledgment. We are grateful to an anonymous referee for several suggestions for simplifications of the proof of Theorem 3.

REFERENCES

[1] A. BLUM, *Machine learning: A tour through some favorite results, directions, and open problems*, Tutorial slides, in the 44th IEEE Symposium on Foundations of Computer Science, Cambridge, MA, 2003, http://www-2.cs.cmu.edu/~avrim/Talks/FOCS03/tutorial.ppt, 2003.

[2] A. BLUM, *Learning a function of r relevant variables (open problem)*, in Proceedings of the 16th Annual Conference on Learning Theory, Washington, D.C., 2003, pp. 731–733.

[3] A. Blum, C. Burch, and J. Langford, *On learning monotone Boolean functions*, in Proceedings of the Thirty-Ninth Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 1998, pp. 408–415.

[4] A. Blum, M. Furst, J. Jackson, M. Kearns, Y. Mansour, and S. Rudich, *Weakly learning DNF and characterizing statistical query learning using Fourier analysis*, in Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, ACM, New York, 1994, pp. 253–262.

[5] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth, *Occam's razor*, Inform. Process. Lett., 24 (1987), pp. 377–380.

[6] N. Bshouty, *Exact learning via the monotone theory*, Inform. and Comput., 123 (1995), pp. 146–153.

[7] N. Bshouty and C. Tamon, *On the Fourier spectrum of monotone functions*, J. ACM, 43 (1996), pp. 747–770.

[8] E. Friedgut, *Boolean functions with low average sensitivity depend on few coordinates*, Combinatorica, 18 (1998), pp. 474–483.

[9] E. Friedgut and G. Kalai, *Every monotone graph property has a sharp threshold*, Proceedings of the AMS, 124 (1996), pp. 2993–3002.

[10] M. Furst, J. Jackson, and S. Smith, *Improved learning of AC^0 functions*, in Proceedings of the Fourth Annual Workshop on Computational Learning Theory, Cambridge, MA, 1991, pp. 317–325.

[11] O. Goldreich, S. Goldwasser, and S. Micali, *How to construct random functions*, J. Assoc. Comput. Mach., 33 (1986), pp. 792–807.

[12] T. Hancock and Y. Mansour, *Learning monotone k-μ DNF formulas on product distributions*, in Proceedings of the Fourth Annual Conference on Computational Learning Theory, Santa Cruz, CA, 1991, pp. 179–193.

[13] J. Jackson, *An efficient membership-query algorithm for learning DNF with respect to the uniform distribution*, J. Comput. System Sci., 55 (1997), pp. 414–440.

[14] J. Jackson, A. Klivans, and R. Servedio, *Learnability beyond AC^0*, in Proceedings of the 34th ACM Symposium on Theory of Computing, ACM, New York, 2002, pp. 776–784.

[15] J. Jackson and C. Tamon, *Fourier analysis in machine learning*, Tutorial slides, ICML/COLT 1997, http://learningtheory.org/resources.html, 1997.

[16] S. Jukna, A. Razborov, P. Savický, and I. Wegener, *On P versus NP$\cap$co-NP for decision trees and read-once branching programs*, Comput. Complexity, 8 (1999), pp. 357–370.

[17] J. Kahn, G. Kalai, and N. Linial, *The influence of variables on Boolean functions*, in Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 1988, pp. 68–80.

[18] M. Kearns, M. Li, and L. Valiant, *Learning Boolean formulas*, J. ACM, 41 (1994), pp. 1298–1328.

[19] M. Kearns and U. Vazirani, *An Introduction to Computational Learning Theory*, MIT Press, Cambridge, MA, 1994.

[20] M. Kharitonov, *Cryptographic hardness of distribution-specific learning*, in Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, ACM, New York, 1993, pp. 372–381.

[21] L. Kučera, A. Marchetti-Spaccamela, and M. Protassi, *On learning monotone DNF formulae under uniform distributions*, Inform. and Comput., 110 (1994), pp. 84–95.

[22] N. Linial, Y. Mansour, and N. Nisan, *Constant depth circuits, Fourier transform and learnability*, J. ACM, 40 (1993), pp. 607–620.

[23] Y. Mansour, *Learning Boolean Functions via the Fourier Transform*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994, pp. 391–424.

[24] Y. Mansour, *An $O(n^{\log\log n})$ learning algorithm for DNF under the uniform distribution*, J. Comput. System Sci., 50 (1995), pp. 543–550.

[25] Y. Mansour and M. Parnas, *Learning conjunctions with noise under product distributions*, Inform. Process. Lett., 68 (1998), pp. 189–196.

[26] K. Oleszkiewicz, *On a nonsymmetric version of the Khinchine-Kahane inequality*, in Stochastic Inequalities and Applications, Prog. Probab. 56, Birkhäuser, Basel, 2003, pp. 156–168.

[27] Y. Sakai and A. Maruoka, *Learning monotone log-term DNF formulas under the uniform distribution*, Theory Comput. Syst., 33 (2000), pp. 17–33.

[28] P. Savický, *On Determinism versus Unambiguous Nondeterminism for Decision Trees*, ECCC report TR02-009, University of Trier, Trier, Germany, available from http://eccc.uni-trier.de/eccc-reports/2002/TR02-009/, 2002.

[29] O. Schramm, R. O'Donnell, M. Saks, and R. Servedio, *Every decision tree has an influential variable*, in Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 2005, pp. 31–39.

[30] R. Servedio, *On learning monotone DNF under product distributions*, Inform. and Comput., 193 (2004), pp. 57–74.

[31] L. Valiant, *A theory of the learnable*, Commun. ACM, 27 (1984), pp. 1134–1142.

[32] L. Valiant, *Short monotone formulae for the majority function*, J. Algorithms, 5 (1984), pp. 363–366.

[33] K. Verbeurgt, *Learning DNF under the uniform distribution in quasi-polynomial time*, in Proceedings of the Third Annual Workshop on Computational Learning Theory, ACM, New York, 1990, pp. 314–326.

[34] K. Verbeurgt, *Learning sub-classes of monotone DNF on the uniform distribution*, in Proceedings of the Ninth Conference on Algorithmic Learning Theory, Otzenhausen, Germany, 1998, pp. 385–399.

SIAM J. Comput.
Vol. 37, No. 3, pp. 845–869

LOWER BOUNDS FOR LOVÁSZ–SCHRIJVER SYSTEMS AND BEYOND FOLLOW FROM MULTIPARTY COMMUNICATION COMPLEXITY*

PAUL BEAME†, TONIANN PITASSI‡, AND NATHAN SEGERLIND§

Abstract. We prove that an $\omega(\log^4 n)$ lower bound for the three-party number-on-the-forehead (NOF) communication complexity of the set-disjointness function implies an $n^{\omega(1)}$ size lower bound for treelike Lovász–Schrijver systems that refute unsatisfiable formulas in conjunctive normal form (CNFs). More generally, we prove that an $n^{\Omega(1)}$ lower bound for the $(k+1)$-party NOF communication complexity of set disjointness implies a $2^{n^{\Omega(1)}}$ size lower bound for all treelike proof systems whose formulas are degree k polynomial inequalities.

Key words. propositional proof complexity, zero-one programming, communication complexity, lower bounds

AMS subject classifications. 68Q17, 03F20, 90C09, 90C60

DOI. 10.1137/060654645

1. Introduction.

Zero-one programming is the problem of optimizing a linear objective function over the zero-one points of a polytope. It is a useful framework for expressing optimization problems. In particular, Boolean conjunctive normal form (CNF) satisfiability can be easily recast as a zero-one programming problem, and for this reason zero-one programming was among the first discrete optimization problems proved to be NP-complete. In contrast, linear programming, the problem of optimizing a linear objective function over all points of a polytope, is polynomial-time solvable [19]. Many attempts have been made to transfer efficient techniques from linear programming to zero-one programming, and among them are the Lovász–Schrijver "lift-and-project" methods. In this paper we establish limitations on using such methods to prove unsatisfiability for CNFs, modulo a conjecture in communication complexity.

Techniques for zero-one programming often come from the slightly more general realm of optimizing over the integral points of a polytope. One approach for reducing these integer programming problems to linear programming problems is to begin with the polytope defined by the original linear program without integrality constraints and systematically pare down the polytope by repeatedly refining the linear program with "cutting planes" that remove only nonintegral solutions until we are left with the convex hull of the integral solutions. These are local methods in which an initial polytope Q is transformed by a sequence of local operations to smaller and smaller subpolytopes until the integral hull of Q is reached. At this point, rational linear

*Received by the editors March 20, 2006; accepted for publication (in revised form) March 22, 2007; published electronically August 8, 2007.
http://www.siam.org/journals/sicomp/37-3/65464.html

†Computer Science & Engineering, University of Washington, Seattle, WA 98195-2350 (beame@cs.washington.edu). The work of this author was supported by NSF grants CCR-0098066, CCR-0514870, and ITR-0219468.

‡Department of Computer Science, University of Toronto, Toronto, ON M5S 1A4, Canada (toni@cs.toronto.edu). The work of this author was supported by an Ontario Premiere's Research Excellence Award, an NSERC grant, and the Institute for Advanced Study. Research was done while at the Institute for Advanced Study.

§Department of Computer Science, Portland State University, Portland, OR (nsegerli@cs.pdx.edu). The work of this author was supported by NSF grant DMS-0303258.

programming finds the correct solution. Note that, for decision problems, this procedure terminates with the empty polytope if and only if the initial polytope contains no integral points. A well-known method of this kind is the use of Gomory–Chvátal cuts [10] which derive each new cutting plane as a linear combination and shift of existing facet constraints.

For zero-one programming, there are more subtle methods available. Lovász and Schrijver [21] introduced a variety of cutting planes methods that derive new cutting planes by first "lifting" the inequalities to higher degree polynomial inequalities (in particular, quadratic inequalities) and then "projecting" them down to linear inequalities using polynomial identities and the fact that $x^2 = x$ for $x \in \{0, 1\}$.

The Lovász–Schrijver methods for solving zero-one programs can be naturally used for propositional proof systems. Consider the problem of proving that a CNF is unsatisfiable (equivalently, proving that a formula in disjunctive normal form (DNF) is a tautology). Each clause is mapped to an equivalent linear inequality; for example, $x \vee \neg y \vee z$ is mapped to $x + 1 - y + z \geq 1$. By repeated application of the lift-and-project rules and elementary linear algebra, inequalities of quadratic polynomials are derived from the translated clauses, and we can arrive at the inconsistent inequality $1 \geq 0$ if and only if the CNF is unsatisfiable.[1] In this way, we obtain propositional proof systems for CNF unsatisfiability in which the formulas are quadratic inequalities, and the rules of inference are the algebraic manipulations coming from the lift-and-project steps and elementary linear algebra. Collectively, these propositional proof systems are known as *Lovász–Schrijver (LS) systems*. An important feature of the LS systems is that they can provide exponentially smaller proofs for certain tautologies, such as the pigeonhole principle, than the ones possible with systems such as resolution or constant-depth Frege systems.

There are two complexity measures that are commonly studied for cutting-planes-based proof systems such as the LS and the Gomory–Chvatal cutting planes system: *size* and *rank*. Intuitively, rank is the number of intermediate polytopes that must be passed through before arriving at the integral hull. In [21] it was shown that for any (relaxed) polytope P, if the rank of P is d, then the optimization and decision problems for P can be solved exactly deterministically in time $n^{O(d)}$. This makes LS systems especially appealing for solving or approximating NP-hard optimization problems via semidefinite programming. A variety of rank lower bounds for the exact solution are known, even for the case of unsatisfiable systems [4, 11, 14, 9, 15]. Moreover, interesting bounds on the rank required to obtain good approximations to the problems of finding a minimum sized vertex cover for a graph and finding an assignment that satisfies a maximum number of clauses in a CNF. This, in turn, implies inapproximability results for these problems for *any* polynomial-time algorithm based on rank.

While there is a rich and growing body of results concerning rank, very little is known about the size of LS proofs. From the proof theoretic perspective, the size of a proof is defined in the usual manner, but from an informal geometric perspective, the size of a LS procedure with respect to some polytope P is the smallest number of hyperplanes defining all of the polytopes that we need to pass through before arriving at the integral hull. Clearly size lower bounds imply rank lower bounds, and indeed, size lower bounds for treelike proofs[2] imply rank lower bounds, but whether the converse holds is open.

[1] The proof systems are made more precise in subsection 2.2.

[2] A proof is treelike if each formula is used at most once as an antecedent to an inference. That is, each time a formula is reused, it must be rederived. For many proof systems, there are CNFs for which the smallest treelike proofs of unsatisfiability are exponentially larger than the smallest unrestricted proofs of unsatisfiability, but it is open whether or not this holds for LS proofs.

At the time of this writing, it is unknown whether or not every unsatisfiable CNF Φ possesses a treelike LS proof of unsatisfiability whose size is bounded by a polynomial in the number of symbols in Φ. Of course, if every unsatisfiable CNF has such a small refutation, then $\mathsf{NP} = \mathsf{coNP}$, so one might say that because this is unlikely, the problem is "resolved modulo a plausible complexity theoretic conjecture." However, this is really begging the question as one would expect that establishing limitations for specific proof methods is prerequisite to establishing limitations for *all* propositional proof systems. Moreover, there are several similar results and conditional lower bounds based on weaker assumptions:

1. The results of Pudlák [25], extended by Dash [12], establish that certain formulations of the LS refutation systems possess "effective interpolation." Therefore, under the conjecture that there are disjoint NP pairs that are not separable by a polynomial-size circuit, these systems require superpolynomial size to refute some CNFs. The hypothesis that some NP disjoint pairs cannot be separated by polynomial-size circuits is not known to imply $\mathsf{NP} \neq \mathsf{coNP}$, so these results provide further evidence that LS proofs require superpolynomial size to refute some CNFs.
2. Grigoriev, Hirsch, and Pasechnik showed that there are unconditional superpolynomial size lower bounds known for treelike LS proofs that certain "non-CNF" polytopes contain no zero-one points [15].
3. Several unconditional lower bounds are known for similar systems that are incomparable with or apparently weaker than treelike LS systems with respect to proof size. An exciting series of papers, culminating in the celebrated result of Pudlák, unconditionally showed that the cutting planes proof system (a kind of logic whose formulas are linear inequalities and whose inference rules are based on Gomory–Chvatal cuts) requires superpolynomial size to refute certain CNFs [16, 5, 24]. Dash has extended this work and proved unconditionally that a restricted form of the LS system (one that makes only "noncommutative cuts") requires superpolynomial size to refute certain CNFs [12].

In this paper, we develop a new method for approaching size lower bounds for treelike LS and for systems that generalize treelike LS. Our main result is that lower bounds on the three-party communication complexity of set disjointness (in the number-on-the-forehead (NOF) model) imply lower bounds on the size of treelike LS proofs for a particular family of unsatisfiable CNF formulas. We also generalize this result to a much more general family of proof systems known as semantic $\mathbf{Th(k)}$, where lines are now degree k polynomial inequalities. All versions of LS are special cases of $\mathbf{Th(2)}$, and Chvátal's cutting planes proof system is a special case of $\mathbf{Th(1)}$.

More generally, we show that proving lower bounds on the $(k+1)$-party communication complexity of set disjointness implies lower bounds on the size of treelike semantic $\mathbf{Th(k)}$ proofs. A lower bound showing that $\textsc{Disj}_k$ is not in $(k+1)$-RP^{cc} would give excellent lower bounds for $\mathbf{Th(k)}$ proofs.

Admittedly, this is another conditional result towards proving size lower bounds for treelike LS proofs. However, we feel that there is a significant difference between an approach that is based on the conjecture "set disjointness requires $\omega(\log^4 n)$ bits of communication in the three-player number-on-the-forehead model" and assumptions such as "$\mathsf{NP} \neq \mathsf{coNP}$" or "there exists an NP disjoint pair that cannot be separated by a polynomial size circuit." The latter problems both imply that $\mathsf{P} \neq \mathsf{NP}$, one of the most famous and difficult problems in contemporary computer science and mathematics, with few persons making serious claims of substantial progress towards its resolution, and the task of establishing proof-size lower bounds can be viewed as establishing partial evidence in support of these conjectures. On the other hand, NOF communication complexity is an area in which there has been substantial progress in the decades since its introduction in 1983 [8]. There are specific, concrete functions on n-bit inputs for which the three-player NOF communication complexity is known to require $\Omega(n)$ bits of communication, so the problem is one of establishing the bound for the set-disjointness function in particular. Moreover, in the two-player model, set disjointness is known to require $\Omega(n)$ bits of communication. Finally, the authors of this paper with Avi Wigderson have established $n^{\Omega(1)}$ lower bounds for the computation of the set-disjointness function by certain restricted protocols in the NOF model.

Our proof can be seen as a generalization of [16] to arbitrary k, but the extension requires a number of new ideas and a substantially more complicated argument that includes a detailed analysis of large sets of vertex-disjoint paths in expander graphs.

Our work is incomparable to the interpolation-based results of Pudlák and Dash. While our bound is conditional, based upon what seems to be a more earthly conjecture than strengthenings of $\mathsf{P} \neq \mathsf{NP}$, our bounds apply only for the treelike case, whereas the results of Pudlák and Dash apply to the DAG-like case as well. On the other hand, their interpolation theorems depend highly on the form of the cuts used, whereas our semantic approach allows the result to apply to almost any system for manipulating polynomial inequalities with reasonable inference rules.

While the results of Grigoriev, Hirsch, and Pasechnik [15] are unconditional, they do not apply to systems of inequalities that arise from the translation of CNFs. Rather, an exponential size lower bound was proved for all treelike LS refutations of the equality $x_1 + \cdots + x_n = \alpha$, where α is a noninteger in the range $(\lceil n/4 \rceil, \lfloor 3n/4 \rfloor)$. As this equality cannot arise as the translation of a CNF into inequalities, their bound says nothing about the LS systems for propositional unsatisfiability. Indeed, proving treelike size lower bounds for CNF polytopes was given as one of the main problems left open in their paper.

Recently Kojevnikov and Itsykson [17] have released a proof of a $2^{\Omega(n)}$ size lower bound for treelike LS_+ refutations of the Tseitin principles over a constant degree expander. This is result is incomparable to ours because while their proof is unconditional, it does not apply to systems that use inference rules besides those of LS_+, nor does it clearly generalize to systems that take inequalities of degree higher than two.

2. Definitions.

2.1. Multiparty communication complexity and set disjointness.

The *k-party NOF model of communication complexity* computes functions (or relations) of input vectors $(x_1, \ldots, x_k) \in X_1 \times \cdots \times X_k$ distributed among k parties such that party $i \in [k]$ sees all x_j for all $j \in [k]$, $j \neq i$. It is as if player i has the ith input on his forehead, hence the name. The players communicate by transmitting bits over a channel shared by all players. The communication complexity of a protocol is the number of bits exchanged. For a function $f : X_1 \times \cdots \times X_k \to \{0,1\}$, we define $R^k_\epsilon(f)$ to be the minimum cost of a randomized protocol that computes f with a probability of error at most ϵ. For a more thorough treatment of communication complexity, see the monograph by Kushilevitz and Nisan [20].

The *k-party set-disjointness problem* $\textsc{Disj}_{k,m} : (\{0,1\}^m)^k \to \{0,1\}$ is defined by $\textsc{Disj}_{k,m}(\vec{x}) = 1$ if and only if there is some $j \in [n]$ such that $x_{i,j} = 1$ for all $i \in [k]$. Although it might be more appropriate to call this function set intersection rather than disjointness, we follow standard terminology. A *$(0,\epsilon)$-error k-party NOF communication protocol* for set disjointness is a protocol that for every disjoint input produces output 0 and for intersecting inputs outputs 1 with probability at least $1-\epsilon$.

It is conjectured that for all constants $k \geq 2$ the k-party set-disjointness problem of length n requires randomized NOF communication complexity that is $\Omega(n/2^k)$ [3]. This conjecture is equivalent to showing that nondeterministic k-party communication complexity can be almost optimally separated from randomized k-party communication complexity. The conjecture is proven for $k = 2$ [18], but the best known lower bound for $k \geq 3$ is $\Omega(\log n)$ for general models and $\Omega(n^{1/k})$ for more restricted models [3].

2.2. Threshold logics. The best known classes of threshold logics are Gomory–Chvátal cutting planes [10], the matrix cuts of Lovász and Schrijver [21], and the lift-and-project relaxations of Sherali and Adams [27]. First we briefly describe Gomory–Chvátal cutting planes, which is referred to in the literature as simply cutting planes (CP). A CP proof of unsatisfiability of a set of integer linear inequalities $f = \{\vec{a}_1 \cdot \vec{x} \geq b_1, \ldots \vec{a}_m \cdot \vec{x} \geq b_m\}$ is a sequence of integer linear inequalities $\vec{c}_1 \cdot \vec{x} \geq t_1, \ldots, \vec{c}_q \cdot \vec{x} \geq t_q$ such that each $\vec{c}_i \cdot \vec{x} \geq t_i$ is either an inequality from f, an axiom ($x \geq 0$ or $1-x \geq 0$), or is obtained by one of the two rules: (i) $\vec{c}_i \cdot \vec{x} \geq t_i$ is a positive integer linear combination of some previously derived inequalities, or (ii) $\vec{c}_i \cdot \vec{x} \geq t_i$ is obtained from a previous inequality $d\vec{c_i} \cdot \vec{x} \geq t_i$ by rounding (to obtain $\vec{c}_i \cdot \vec{x} \geq \lceil t_i/d \rceil$).

There are several cutting planes proof systems defined by Lovász and Schrijver [21], collectively referred to as matrix cuts. These systems allow one to "lift" the linear inequalities to degree-two polynomials and then project back to degree one, using the fact that $x^2 = x$ for $x \in \{0,1\}$. To see that the definitions below are equivalent to the original definitions of Lovász and Schrijver, see [12].

Definition 2.1. *Given a polytope $P \subseteq \mathbb{Q}^n$ defined by $\vec{a}_i \cdot \vec{x} \geq b_i$ for $i = 1, 2, \ldots, m$:*

(1) *An inequality $d - \vec{c} \cdot \vec{x} \geq 0$ is called an N-cut for P if*

$$d - \vec{c} \cdot \vec{x} = \sum_{i,j} \alpha_{ij}(b_i - \vec{a}_i \cdot \vec{x})x_j + \sum_{ij} \beta_{ij}(b_i - \vec{a}_i \cdot \vec{x})(1-x_j) + \sum_j \lambda_j(x_j^2 - x_j),$$

where $\alpha_{ij}, \beta_{ij} \geq 0$ and $\lambda_j \in R$ for $i = 1, \ldots, m$, $j = 1, \ldots, n$.

(2) *A weakening of N-cuts, called N_0-cuts, can be obtained if, when simplifying to the term $d - \vec{c} \cdot \vec{x}$, we view $x_i x_j$ as distinct from $x_j x_i$.*

(3) *An inequality $d - \vec{c} \cdot \vec{x}$ is called an N_+-cut if*

$$d - \vec{c} \cdot \vec{x} = \sum_{i,j} \alpha_{ij}(b_i - \vec{a}_i \cdot \vec{x})x_j + \sum_{ij} \beta_{ij}(b_i - \vec{a}_i \cdot \vec{x})(1-x_j) + \sum_j \lambda_j(x_j^2 - x_j) + \sum_k (g_k + \vec{h}_k \cdot \vec{x})^2,$$

where again $\alpha_{ij}, \beta_{ij} \geq 0$, $\lambda_j \in R$ for $i = 1, \ldots, m$, $j = 1, \ldots, n$, and $g_k + \vec{h}_k \cdot \vec{x}$ is an affine function for $k = 1, \ldots, n+1$.

The operators N, N_0, and N_+ are called the *commutative*, *noncommutative*, and *semidefinite* operators, respectively. All three are collectively called *matrix-cut* operators.

Definition 2.2. *A LS refutation for f is a sequence of inequalities $g_1, \ldots, g_q$ such that each g_i is either an inequality from f or follows from previous inequalities* *by an N-cut as defined above and such that the final inequality is* $0 \geq 1$. *Similarly, a LS_0 refutation uses N_0-cuts and LS_+ uses N_+-cuts.*

DEFINITION 2.3. *Let $\mathcal{P}$ be one of the proof systems CP, LS, LS_0, or LS_+. Let S be an $\mathcal{P}$-refutation of f, viewed as a directed acyclic graph. If the underlying directed acyclic graph is a tree, then S is a treelike $\mathcal{P}$-refutation of f. The inequalities in S are represented with all coefficients in binary notation. The size of S is the sum of the sizes of all inequalities in S; the rank of S is the depth of the underlying directed acyclic graph. For a set of Boolean inequalities f, the $\mathcal{P}$-size of f is the minimal size over all $\mathcal{P}$ refutations of f. Similarly the $\mathcal{P}$-treesize of f is the minimal size over all treelike $\mathcal{P}$-refutations of f.*

The inference rules and axioms for the CP, LS, LS_0, and LS_+ systems are easily seen to be sound. Furthermore, it has been shown that, in their treelike forms, each of CP, LS, LS_0, and LS_+ can p-simulate treelike resolution (cf. [3]). Therefore, by the completeness of treelike resolution, the treelike systems CP, LS, LS_0, and LS_+ can refute every unsatisfiable CNF.

All of the above proof systems are special cases of the more general *semantic* threshold logic proof systems which we define now. A *k-threshold formula* over Boolean variables $x_1, \ldots, x_n$ is a formula of the form $\sum_j \gamma_j m_j \geq t$, where γ_j, t are integers, and for all j, m_j is a multilinear monomial of degree at most k. The *size* of a k-threshold formula is the sum of the sizes of γ_j and t, written in binary notation. Let f_1, f_2, g be k-threshold formulas in the variables $\vec{x}$. We say that g *is semantically entailed* by f_1 and f_2 if, for every $0/1$ assignment to $\vec{x}$ that satisfies both f_1 and f_2, g is also satisfied.

Let f be an unsatisfiable CNF formula over $x_1, \ldots, x_n$, and let $t_1, \ldots, t_m$ be the underlying set of clauses of f, written as 1-threshold inequalities. A **Th(k)** *refutation of* f, $\mathcal{P}$, is a sequence of k-threshold formulas $L_1, \ldots, L_q$, where each L_j is one of the inequalities t_i, $i \in [m]$, or is semantically entailed by two formulas L_i and $L_{i'}$, with $i, i' < j$, and the final formula L_q is $0 \geq 1$. The *size* of $\mathcal{P}$ is the sum of the sizes of all k-threshold formulas occurring in $\mathcal{P}$. The proof is *treelike* if the underlying directed acyclic graph, representing the implication structure of the proof, is a tree. (That is, every formula in the proof is used at most once as an antecedent of an implication. It is allowed, and quite often necessary, that $L_i = L_j$, and L_i and L_j are used as antecedents for two different inferences. In this way, a formula must be rederived each time it is used.)

Note that, in our definition of these cutting planes systems, we can derive a new inequality from any number of previous inequalities in one step, whereas in the $\mathbf{Th(k)}$ proof system, we are restricted to fan-in two. Because the vector space of degree-at-most-one inequalities has dimension at most $n+1$, in light of Caratheodory's theorem, every inequality derived by purely linear operations in a CP refutation can be derived from at most $n+2$ many previous equations. Therefore, we can assume without loss of generality that the fan-in is at most $n+2$ in CP and, similarly, at most $\binom{n}{2} + n + 2$ in LS, LS_0, and LS_+. Because of this bound on fan-in, refutation size increases by at most an $O(n^2)$ factor when the sums are taken by fan-in two inferences of the form "from $f \geq a$ and $g \geq b$ infer $f + g \geq a + b$." Thus, superpolynomial size lower bounds for treelike **Th(2)** semantic refutations imply superpolynomial size lower bounds for all treelike LS systems.

Because the inference rule in $\mathbf{Th(k)}$ is semantic entailment, lower bounds for the $\mathbf{Th(k)}$ system apply to almost any treelike system for deriving polynomial inequalities with reasonable axioms and inference rules, not only the LS systems. For example, division operators such as "from $\vec{c} \cdot \vec{x} > 0$ and $(b - \vec{a} \cdot \vec{x})\vec{c} \cdot \vec{x} \geq 0$ infer $\vec{a} \cdot \vec{x} \geq b$"

are semantically valid inference rules of fan-in two, and variants of the LS system incorporating such rules fall under our analysis. Furthermore, CP refutations are a special case of $\mathbf{Th(1)}$ semantic refutations, and thus lower bounds for treelike $\mathbf{Th(1)}$ semantic refutations imply similar lower bounds for treelike CP. This connection was exploited to prove lower bounds for treelike CP [16].

2.3. Miscellaneous notation. We use the standard asymptotic notation of Ω, O, ω, and o that is found in theoretical computer science and discrete mathematics. We use the ± notation in the following nonstandard way: When we write $x = a \pm b$, we mean that $x \in [a - b, a + b]$. We use this in the asymptotic sense as well. When we write $x = (1 \pm o(1))M$, we mean that there is a value t with $|t| = o(1)$ so that $x \in [(1 - t)M, (1 + t)M]$.

3. Relating the size of threshold logic refutations to the communication complexity of search problems. Let f be an unsatisfiable CNF formula. We will be interested in the following search problem, Search_f associated with f: Given a truth assignment α, find a clause from f which is falsified by α. The model for this computation is a decision tree whose nodes evaluate polynomial threshold functions.

A *k-threshold decision tree* is a rooted, directed tree whose vertices are labeled with k-threshold functions and edges are labeled with either 0 or 1. The leaves of the tree are labeled with clauses of f. A k-threshold decision tree solves Search_f in the obvious way: Start at the root and evaluate the threshold function; follow the edge that is consistent with the value of the threshold function; continue until the computation reaches a leaf and output the associated clause. The size S of a k-threshold decision tree is the sum of the sizes of all threshold formulas in the tree, where the coefficients are written in binary. The depth of a k-threshold decision tree is the depth of the underlying tree.

The following lemma, similar to the degree 1 case in [16], shows that from a small treelike $\mathbf{Th(k)}$ refutation of an unsatisfiable formula f a small-size, small-depth k-threshold decision tree for Search_f can be extracted.

LEMMA 3.1. *Let $\mathcal{P}$ be a treelike $\mathbf{Th(k)}$ refutation of f of size S. Then there is a k-threshold decision tree for Search_f of depth $O(\log S)$ and size $O(S)$. Furthermore, every threshold formula labeling a node of the decision tree is either a formula in the refutation $\mathcal{P}$ or the vacuously true inequality $0 \geq 0$.*

Proof. Assume that $\mathcal{P}$ is a size S treelike $\mathbf{Th(k)}$ refutation of f. We will describe a depth $O(\log S)$, size $O(S)$, k-threshold decision tree which computes the search problem associated with f. The proof is by induction on S; clearly if $S = 1$, then the unsatisfiable formula is a single, false threshold formula, so the lemma holds. For the inductive statement, assume that the size of $\mathcal{P}$ is $S > 1$. Because the DAG of the proof is a binary tree, there is an intermediate formula f in $\mathcal{P}$ such that the number of formulas above f (ancestors in the tree) is at least $S/3$ and at most $2S/3$. Let the subtree of $\mathcal{P}$ with root formula f be denoted by $\mathcal{A}$, and write $\mathcal{B}$ for the remainder of $\mathcal{P}$, that is, all formulas of $\mathcal{P}$ that are not in $\mathcal{A}$ and with f replaced by $0 \geq 0$. In the decision tree, the root is labeled with f. Beneath the edge labeled 0, we inductively apply the lemma to the subtree $\mathcal{A}$, and beneath the edge labeled 1, we inductively apply the lemma on the subtree $\mathcal{B}$. Both $\mathcal{A}$ and $\mathcal{B}$ have size at most $2S/3$, so we may apply the induction hypothesis and conclude that the height of the decision tree obtained will be at most $\log_{3/2}(S) + 1$, which is $O(\log S)$. To see that the decision tree computes the search function, notice that if f evaluates to false on a given truth assignment ϕ, then we proceed on the subproof $\mathcal{A}$. By soundness of the proof, at least one of the leaf formulas of $\mathcal{A}$ must be falsified by ϕ. A similar argument holds when f evaluates to true. □

The next lemmas, adapted from arguments in [23], show that any relation computed by a shallow k-threshold decision tree can also be efficiently computed by a $k+1$ player communication complexity protocol in the NOF model, over any partition of the variables.

LEMMA 3.2. *Suppose that a relation $R(x_1, \ldots, x_{kn})$ is computed by a depth d k-threshold decision tree in which all coefficients are bounded by $N \geq n$. For any partition of the inputs into k sets, there is a $k+1$-party deterministic NOF communication complexity protocol for R in which $O(d \log N)$ bits are communicated in total.*

Proof. Fix a partition of $x_1, \ldots x_{kn}$. Observe that for each monomial in each k-threshold formula there is at least one party that can evaluate the monomial. Let $\alpha_1 m_1 + \cdots + \alpha_q m_q \geq t$ be the k-threshold formula queried at the root of the k-threshold decision tree for f. The set of monomials m_j can be partitioned into $k+1$ groups, where group i contains monomials that can be "seen" by the ith player. Each player (in turn) communicates the weighted linear combination of their monomials to the other players. After all players have spoken, each player can simply add up the total sum and see if it is greater than the target t, in order to evaluate the k-threshold formula. The $k+1$ players then continue on the half of the decision tree which agrees with the value of this formula. The protocol terminates after d rounds, and each round requires $O(\log N)$ bits of communication. □

In order to prove the randomized version of the above lemma we use a standard randomized protocol for testing linear inequalities. The protocol works in the *number-in-hand* model which is more restricted than the NOF model. In the number-in-the-hand model, each player $i = 1, \ldots k$ has private access to the input x_i (whereas in the NOF model, player i sees inputs $x_1, \ldots x_{i-1}, x_{i+1}, \ldots x_k$).

LEMMA 3.3. *Let $y_1, \ldots, y_{k+1}$ be (signed) integers with n-bit binary representations, and let $c > 0$. Then there is an $O(k \log^2 n)$-bit $(k+1)$-player number-in-hand probabilistic protocol with an error less than $1/n^c$ for determining whether $y_1 + \cdots + y_{k+1} \geq 0$.*

Proof. The players follow a binary search strategy on the bits of the y_i.

Suppose $n \geq 2$. (If $n \leq 2$, the parties simply send their inputs.) Let y_i^H be the high order $\lceil n/2 \rceil$ bits that underapproximate $y_i/2^{\lfloor n/2 \rfloor}$ and y_i^L be the corresponding low order bits for $1 \leq i \leq k+1$. (Some of the y_i may be negative, but then the y_i^L will all be positive.) If $\sum_i y_i^H > 0$, then $\sum_i y_i > 0$; similarly, if $\sum_i y_i^H < -k$, then $\sum_i y_i < 0$. Thus, unless $\sum_i y_i^H \in \{-k, \ldots, 0\}$, the answer can be found by determining whether $y_1^H + \cdots + y_{k+1}^H \geq 0$. If $\sum_i y_i^H = -j \in \{-k, \ldots, 0\}$, then the answer can be found by comparing $y_1^L + \cdots + y_k^L + y_{k+1}^L - j \cdot 2^{\lfloor n/2 \rfloor}$ to 0.

Player 1 randomly selects a prime number $p \in [n^{c+2} \log n, 2n^{c+2} \log n]$ and sends $(p, y_1^H \bmod p)$. For $i = 2$ to k, player i sends $y_i^H \bmod p$. Then, using these values and his own private input, player $k+1$ computes $z = \sum_{i=1}^{k+1} y_i^H \bmod p$.

If $z \not\equiv -j \pmod p$ for $j \in \{0, \ldots, k\}$, then player $k+1$ sends the bit 1, and the protocol continues recursively, using $y_1^H, \ldots, y_{k+1}^H$ instead of $y_1, \ldots, y_{k+1}$.

If $z \equiv -j \pmod p$ for $j \in \{0, \ldots, k\}$, then player $k+1$ sends the bit 0 and j, and the protocol continues recursively with players 1 to j using $(y_1^L - 2^{\lfloor n/2 \rfloor}), \ldots, (y_j^L - 2^{\lfloor n/2 \rfloor})$ instead of $y_1, \ldots, y_j$ and players $j+1$ to $k+1$ using $y_{j+1}^L, \ldots, y_{k+1}^L$ instead of $y_{j+1}, \ldots, y_{k+1}$.

In both of the recursive calls the integers each have at most $\lfloor n/2 \rfloor + 1$ bits. At each stage, we send $O(\log n)$ bits, and the total number of stages is $O(\log n)$ for a total of $O(\log n)^2$ bits sent. The probability of error at each stage is $O(1/n^{c+1})$, and therefore the total error is less than $1/n^c$ (for sufficiently large n). □

LEMMA 3.4. *Suppose that a relation $R(x_1, \dots, x_{kn})$ is computed by a depth d k-threshold decision tree in which all coefficients are bounded by $N \geq n$. For any partition of the inputs into k sets, there is a $(k+1)$-party randomized NOF communication complexity protocol for R in which $O(d(\log \log N)^2)$ bits are communicated in total, which is correct with probability at least $1 - 1/n$.*

Proof. As in the proof of Lemma 3.2, the players proceed in d rounds, at each step evaluating the threshold formula and proceeding on the consistent subtree. Let $p(\vec{x}) \geq t$ be the threshold formula at the root of the decision tree. As before, partition the monomials of $p(\vec{x})$ into $k+1$ groups where the ith player can "see" the monomials in group i. Each of the $k+1$ players computes the weighted sum of their respective monomials. Call these sums $y_1, \dots, y_{k+1}$, respectively. Player $k+1$ uses $y'_{k+1} = y_{k+1} - t$, and by applying Lemma 3.3 with $n = \log_2 N$ and c such that $1/n^c < 1/(dn)$, there is a probabilistic protocol allowing the players to determine with an error at most $1/(dn)$ whether the sum of the y_i's is at least t, where $O((\log \log N)^2)$ bits are exchanged. After evaluating $p(\vec{x}) \geq t$, the players then continue on the branch of the decision tree which agrees with the value of $p(\vec{x}) \geq t$. The protocol terminates after d rounds, for a total of $O(d(\log \log N)^2)$ bits of communication. By the union bound, the probability of encountering an error at some level of the recursion is at most $d \cdot 1/(dn) = 1/n$. □

The following theorem is an easy corollary of the above lemmas.

THEOREM 3.5. *Suppose that f has a treelike* **Th(k)** *refutation of size S. Then there exists a $(k+1)$-party randomized NOF communication complexity protocol for $Search_f$ (over any partition of the variables into k groups) that communicates $O(\log^3 S)$ bits and has an error probability at most $1/n$.*

Further, if all k-threshold formulas in the **Th(k)** *refutation have coefficients bounded by a polynomial in n, then there is a randomized protocol using $O(\log S(\log \log n)^2)$ many bits and an error probability at most $1/n$ and a deterministic protocol using $O(\log S \log n)$ bits.*

Proof. We apply Lemma 3.1 to produce a k-threshold decision tree for $Search_f$ of depth $O(\log S)$ and size $O(S)$. Because every label of a node of the decision tree is a formula of the refutation, or the triviality $0 \geq 0$, N is no larger than the maximum absolute value of a coefficient in the refutation.

For the first claim, set N to be the maximum absolute value of any coefficient appearing in the decision tree; by the definition of the size of a proof, $N = 2^{O(S)}$. For the second claim, we find by hypothesis that $N = n^{O(1)}$. We apply Lemmas 3.2 and 3.4 to this decision tree to yield the claimed size and error bounds. □

4. The difficult CNFs, their search problems, and an outline of the lower bound proof.

Our hard examples are based on the well-known Tseitin graph formulas. Let $G = (V, E)$ be any connected, undirected graph, and let $\vec{c} \in \{0,1\}^V$. The *Tseitin formula for G with respect to charge vector $\vec{c}$, $TS(G, \vec{c})$*, has variables $\mathrm{Vars}(G) = \{y_e \mid e \in E\}$. The formula states that, for every vertex $v \in V$, the parity of the edges incident with v is equal to the charge c_v at node v. It is expressed propositionally as the conjunction of the clauses obtained by expanding $\oplus_{e \ni v} y_e = c_v$ for each $v \in V$. Note that, for a graph with maximum degree d, each clause is of width at most d and the number of clauses is at most $|V|2^d$.

Notice that $TS(G,\vec{c})$ is satisfiable if and only if $\sum_{v\in V} c_v$ is even. For odd $\vec{c}$, the search problem $Search_{TS(G,\vec{c})}$ takes a 0/1 assignment α to Vars(G) and outputs a clause of $TS(G,\vec{c})$ that is unsatisfied. In other words, a solution to $Search_{TS(G,\vec{c})}$ on input α is a vertex v such that a parity equation at the vertex v is violated by α.

To make the search problem hard for k-party NOF communication protocols and, by Theorem 3.5, hard for $(k-1)$-threshold decision trees, we modify $TS(G,\vec{c})$ by replacing each variable y_e by the conjunction of k variables $\bigwedge_{i=1}^{k} y_e^i$ and expanding the result into clauses by use of deMorgan's law. We call the resulting *k-fold Tseitin formula* $TS^k(G,\vec{c})$ and its variable set $\text{Vars}^k(G) = \{y_e^i \mid e \in E, i \in [k]\}$.

For a fixed graph G and different odd-charge vectors $\vec{c} \in \{0,1\}^{V(G)}$, the problems $Search_{TS^k(G,\vec{c})}$ are very closely related. Define $\text{OddCharge}^k(G)$ to be the k-party NOF communication search problem which takes as input an odd charge vector $\vec{c} \in \{0,1\}^{V(G)}$, seen by all players, and an assignment α to $\text{Vars}^k(G)$, in which player i sees all values but the assignment α_e^i to y_e^i for $e \in E(G)$, and requires that the players output a vertex v that is a solution to $Search_{TS^k(G,\vec{c})}$.

The communication complexity of $\text{OddCharge}^k(G)$ depends on the graph G, and we use a carefully modified expander to obtain our lower bounds. We use a family of graphs H_n such that each H_n is the union of two edge-disjoint graphs on the same set of n vertices $[n]$, G_n, and T_n. G_n is a Δ-regular expander graph of the form defined by Lubotzky, Phillips, and Sarnak [22] for $\Delta = \Theta(\log n)$. Since $\overline{G_n}$ has degree $> n/2$, there is a spanning tree T_n of maximum degree 2 (one can take the Hamiltonian path guaranteed by Dirac's theorem; cf. [13]) in $\overline{G_n}$. Clearly H_n also has maximum degree $\Theta(\log n)$, and thus $TS^k(H_n,\vec{c})$ has size $n^{O(k)}$. (Notice that the graph H_n has degree $O(\log n)$, so the CNF $TS^k(H_n,\vec{c})$ has size $n^{O(1)}$ and width $O(\log n)$.)

Now we are ready to describe the sequence of reductions to show that an efficient k-party NOF communication complexity protocol for $\text{OddCharge}^k(H_n)$ will imply an efficient 1-sided error randomized k-party NOF protocol for the set-disjointness relation. The reduction passes through two intermediate problems: a search problem called $\text{EvenCharge}^k(H_n)$ and set disjointness with the promise that, for every input under consideration, the size of the intersection is either zero or one. Reducing the general set-disjointness problem to this *zero/one set-disjointness* problem is a standard application of Valiant–Vazirani isolation (Lemma 5.4, after [28]). Our reduction from zero/one set disjointness to $\text{OddCharge}^k(H_n)$ goes via an intermediate problem $\text{EvenCharge}^k(H_n)$, which is the exact analog of $\text{OddCharge}^k(H_n)$ except that the input charge vector $\vec{c}$ is even rather than odd and the task is either to find a charge violation or to determine that no charge violation exists. For an assignment α and a charge vector $\vec{c}$, we define $\text{Err}(\alpha,\vec{c})$ to be the set of vertices at which the parity constraints are violated by α.

THEOREM 4.1. *Let $k \geq 2$ and $m = n^{1/3}/\log n$. For each n there is an odd charge vector $\vec{c} \in \{0,1\}^n$ such that for any $\epsilon < 1/2$ the size of any treelike* **Th(k-1)** *refutation of $TS^k(H_n,\vec{c})$ is at least $2^{\Omega((R_\epsilon^k(\text{Disj}_{k,m})/\log n)^{1/3})}$. Further, if the coefficients in the* **Th(k-1)** *refutations are bounded by a polynomial in n, then the refutation size must be at least $2^{\Omega(R_\epsilon^k(\text{Disj}_{k,m})/(\log n(\log\log n)^2))}$.*

The proof of Theorem 4.1 is presented at the end of section 5. Here we provide a high-level outline of the proof and its component lemmas. In the sketch, quantities are left out, and definitions are not precise.

Proof sketch. Suppose for the sake of contradiction that there is a small $\mathbf{Th(k-1)}$ refutation of $TS^k(H_n,\vec{c})$.

1. We apply the refutation-to-search conversion of Theorem 3.5 to obtain a low-communication k-player NOF protocol for the ODDCHARGE$^k(H_n)$ search problem.
2. Using Lemma 5.1, we convert the search protocol for ODDCHARGE$^k(G)$ to a search protocol for EVENCHARGE$^k(H_n)$ that correctly solves "most" EVENCHARGE$^k(H_n)$ instances. "Most" is measured by a distribution $\mathcal{D}_t$ on the EVENCHARGE$^k(H_n)$ instances in which there are exactly $2t$ nodes at which the parity constraints are violated. The distribution is $\mathcal{D}_t$ defined in Definition 5.2 of section 5.
3. In Lemma 5.2 we show that the 0/1 set-disjointness problem randomly reduces to EVENCHARGE$^k(H_n)$ in the following sense: For each set-disjointness instance $\vec{x}$, there is a distribution $R(\vec{x})$ on EVENCHARGE$^k(H_n)$ instances so that if $|\cap\vec{x}| = 0$, the instance generated satisfies all parity constraints, and if $|\cap\vec{x}| = 1$, the instance generated has exactly two nodes at which the parity constraints are violated.
4. The distributions $R(\vec{x})$ and $\mathcal{D}_t$ do not coincide, but they are close enough. In Lemma 5.3, it is shown that when $|\cap\vec{x}| = 0$, $R(\vec{x})$ and $\mathcal{D}_0$ are ϵ-close in l_1 distance, and similarly, when $|\cap\vec{x}| = 1$, $R(\vec{x})$ and $\mathcal{D}_1$ are ϵ-close in l_1 distance. Therefore, using the protocol of Lemma 5.1 on inputs generated by $R(\vec{x})$ correctly solves 0/1 set disjointness with an added probability of error at most ϵ. Lemma 5.3 is the most delicate part of the argument, and it is where most of the work is invested.

A simple argument shows that the lower bound of Theorem 4.1 holds for all odd charge vectors.

THEOREM 4.2. *The same lower bounds as Theorem* 4.1 *hold for* every *odd charge vector* $\vec{c} \in \{0,1\}^n$.

Proof. Observe that distributions $\mathcal{D}_t$ and $R(\vec{x})$ on the assignments to Vars$^k(H_n)$ both have the property that for each edge e of T_n, $\alpha_e^1 = \cdots = \alpha_e^k$. Therefore in the proof of Theorem 4.1 observe that we can replace $TS^k(H_n,\vec{c})$ by $\widetilde{TS}^k(H_n,\vec{c}) = TS^k(H_n,\vec{c}) \wedge EQ(T_n)$, where $EQ(T_n)$ is the conjunction of $(\neg y_e^i \vee y_e^j)$ for every $i \neq j \in [k]$ and every $e \in T_n$. The size of any **Th(k-1)** refutation of $TS^k(H_n,\vec{c})$ is at least that of $\widetilde{TS}^k(H_n,\vec{c})$. Moreover, it is not hard to see that, for any odd weight vectors $\vec{c},\vec{d} \in \{0,1\}^n$, $\widetilde{TS}^k(H_n,\vec{c})$ and $\widetilde{TS}^k(H_n,\vec{d})$ have proof sizes that differ by at most a polynomial additive term: Given a small proof of $\widetilde{TS}^k(H_n,\vec{d})$, let $S \subset [n]$ be the set of vertices v for which $c_v \neq d_v$. Since both c and d are odd weight vectors, $|S|$ is even. Let $M \subset E(T_n)$ be the set of edges of corresponding to $|S|/2$ disjoint subpaths in T_n that match the elements in S.

Applying the substitution of $y_e^i = \neg y_e^i$ for each $e \in M$ and $i \in [k]$ *almost* converts a refutation of $\widetilde{TS}^k(H_n,\vec{d})$ into a refutation of $\widetilde{TS}^k(H_n,\vec{c})$. A positive literal y_e^i in a clause from $TS^k(H_n,\vec{c})$, with e on a toggled path, becomes $\neg y_e^i$ rather than $\neg y_e^1 \vee \cdots \vee \neg y_e^k$, which is the proper form for negative literals in clauses of $TS^k(H_n,\vec{d})$. This is corrected by application of the subsumption rule. A disjunction of negative literals $\neg y_e^1 \vee \cdots \vee \neg y_e^k$ in a clause from $TS^k(H_n,\vec{c})$, with e on a toggled path, becomes $y_e^1 \vee \cdots \vee y_e^k$ rather than one of $y_e^1,\ldots,y_e^k$, as is the proper form for positive literals in clauses of $TS^k(H_n,\vec{d})$. Application of the axioms $\neg y_e^j \vee y_e^i$, with resolution steps, corrects this. These corrections increase the size of the proof by at most an $O(k)$ factor. □

5. Reduction from set disjointness to OddCharge. The reduction from EvenCharge$^k(H_n)$ to OddCharge$^k(H_n)$ works by planting a single randomly chosen additional charge violation. This yields a protocol for EvenCharge$^k(H_n)$ that works well on average for each class of inputs with a given number of charge violations. This is similar in spirit to a reduction by Raz and Wigderson [26], and the reader might profit by first becoming familiar with that argument.

The difficult part of our argument is the reduction from zero/one set disjointness to EvenCharge$^k(H_n)$. The key idea is that for even $\vec{c}$, charge violations of $TS^k(H_n, \vec{c})$ come in pairs: Given an instance $\vec{x} \in (\{0,1\}^m)^k$ of zero/one set disjointness, using the public coins, the players randomly choose an even charge vector $\vec{c}$ and m vertex-disjoint paths in H_n, $p_1, \ldots, p_m$, for each $j \in [m]$, the players plant the $x_{1,j}, \ldots, x_{k,j}$ as the assignment along each edge of path p_j, in a random solution that otherwise meets the chosen charge constraint. By construction, a charge violation can occur only at the end points of a path and only if there is an intersection in the set-disjointness problem.

The challenges arise when we would like to apply the average case properties of the EvenCharge$^k(H_n)$ protocol to the instances created by the above distribution. Unfortunately, this distribution is not quite uniform, and we need the distribution to be close to uniform. The bulk of the work is in using the properties of H_n, rapid mixing, modest degree, and high girth, to show that the distribution generated by the reduction is sufficiently close to uniform.

Distributions on labeled graphs. Let n be given, let H_n be the graph described in section 4, and let $\vec{c}$ be an even charge vector.

Definition 5.1. *We define $Sol(H_n, \vec{c})$ to be the set of all $0/1$ assignments to the edges of H_n so that, for each vertex $v \in [n]$, the parity of edges incident with v is equal to c_v. A uniform random distribution over $Sol(H_n, \vec{c})$ can be obtained by first selecting $0/1$ values uniformly at random for all edges in G_n and then choosing the unique assignment to the edges of T_n that fulfill the charge constraints given by $\vec{c}$.*

Given a bit value b associated with an edge $e \in G_n$, we can define a uniform distribution $\mathcal{L}_k(b)$ over the corresponding variables y_e^i, $i \in [k]$. Such an assignment is chosen randomly from $\mathcal{L}_k$ on input b by the following experiment. If $b = 1$, then set all variables associated with edge e, y_e^i, $i \in [k]$ to 1. Otherwise, if $b = 0$, set the vector $(\vec{y}_e)_{i \in [k]}$ by choosing uniformly at random from the set of $2^k - 1$ not-all-1 vectors (i.e., $\{0,1\}^k \setminus \{0^k\}$).

Definition 5.2. *For any $t \geq 0$ let $\mathcal{D}_t$ be a distribution given by the following experiment on input $H_n = G_n \cup T_n$:*

1. *Choose an even charge vector $\vec{c} \in \{0,1\}^n$ uniformly at random.*
2. *Choose $\beta \in Sol(H_n, \vec{c})$ uniformly at random.*
3. *For each $e \in G_n$, select the values for the vector $(y_e)_{i \in [k]}$ from $\mathcal{L}_k(\beta_e)$, and, for each $e \in T_n$, set $y_e^i = \beta_e$ for all $i \in [k]$.*
4. *Select a random subset $U \subseteq [n]$ of $2t$ vertices, and produce charge vector $\vec{c}^{\,U}$ from $\vec{c}$ by toggling all bits c_v for $v \in U$.*
5. *Return the pair $(\alpha, \vec{c}^{\,U})$, where α is the Boolean assignment to the variables y_e^i, $i \in [k]$, $e \in H_n$.*

Reduction from EvenCharge to OddCharge.

Lemma 5.1. *Let n be given, and let Δ be the maximum degree of a vertex in H_n. Suppose that Π_{odd} is a randomized k-party NOF protocol for* OddCharge$^k(H_n)$ *that produces a vertex with a probability at least $1 - \epsilon$, is correct whenever it produces a* *vertex, and uses at most s bits of communication. Then there is a randomized k-party NOF protocol Π_{even} for* $\textsc{EvenCharge}^k(H_n)$ *that uses $s + \Delta$ bits of communication and has the following performance:*

$$\Pr_{(\alpha,\vec{c})\in\mathcal{D}_0}[\Pi_{even}(\alpha,\vec{c}) = \text{true}] = 1,$$
$$\Pr_{(\alpha,\vec{c})\in\mathcal{D}_t}[\Pi_{even}(\alpha,\vec{c}) \in \text{Err}(\alpha,\vec{c})] \geq 2/3 - \epsilon \textit{ for } t \geq 1.$$

Proof. Let Π_{odd} be a protocol for $\textsc{OddCharge}^k(H_n)$. We give a protocol Π_{even} for $\textsc{EvenCharge}^k(H_n)$. On input $(\alpha, \vec{c})$ and random public string r, using r, choose a random vertex $v \in [n]$. Check whether the parity equation associated with vertex v is satisfied by α using at most $\Delta(G)$ bits of communication. (This can be done by having player 1 broadcast y_e^2 for each $e \ni v$ and then having player 2 compute whether the constraint at v is obeyed or violated.) If it is not, return v. Otherwise, create an odd charge vector $\vec{c}^{\,\{v\}}$, which is just like $\vec{c}$ except that the value of c_v is replaced by $1 - c_v$. Now run Π_{odd} on input $(\vec{c}^{\,\{v\}}, \alpha)$. If Π_{odd} returns the planted error v or if Π_{odd} does not return a value, then return "true"; if Π_{odd} returns $u \neq v$, output u.

Suppose that $(\alpha, \vec{c}) \in \mathcal{D}_0$. Then α satisfies all charges specified by $\vec{c}$, so when Π_{odd} returns a vertex, the above protocol must output "true" because Π_{odd} has a one-sided error–that is, Π_{odd} will return a vertex u only when there is an error on the parity equation associated with u. Now suppose that $(\alpha, \vec{c}) \in \mathcal{D}_t$, so exactly $2t$ parity equations are violated. If the parity constraint about the vertex v is not satisfied, then the protocol detects this and correctly reports the location of the error. The remaining case is when the parity constraint at v is satisfied, and in this case we call Π_{odd} on a pair $(\alpha, \vec{c}^{\,\{v\}})$ where exactly $2t + 1$ parity equations are violated.

We show the probability bound by conditioning separately on the events $\text{Err}(\alpha, \vec{c}^{\,\{v\}}) = T$ for each $T \in \binom{[n]}{2t+1}$. Because the events $\text{Err}(\alpha, \vec{c'}) = T$ partition the probability space, this proves the claim. By symmetry, for $T \in \binom{[n]}{2t+1}$ and *any* function g whose range is a subset of T, we have that $\Pr_{\alpha,\vec{c},v}[g(\alpha, \vec{c}^{\,\{v\}}) = v \mid \text{Err}(\alpha, \vec{c}^{\,\{v\}}) = T] = 1/(2t+1)$ since it is equally likely for $\vec{c'} = \vec{c}^{\,\{v\}}$ to be generated as $\vec{c}^{\,\{u\}}$ for any $u \in T$. Thus we obtain

$$\begin{aligned}
&\Pr_{\alpha,\vec{c},v}[\Pi_{even}(\alpha, \vec{c}^{\,\{v\}}) \text{ errs} \mid \text{Err}(\alpha, \vec{c}^{\,\{v\}}) = T] \\
&= \Pr_{\alpha,\vec{c},v}[\Pi_{odd}(\alpha, \vec{c}^{\,\{v\}}) = v \text{ or } \Pi_{odd}(\alpha, \vec{c}^{\,\{v\}}) \text{ is not defined } \mid \text{Err}(\alpha, \vec{c}^{\,\{v\}}) = T] \\
&\leq 1/(2t+1) + \epsilon \;\leq\; 1/3 + \epsilon
\end{aligned}$$

for $t \geq 1$. □

Reduction from zero/one set disjointness to EvenCharge. We now show how to use a k-party NOF communication complexity protocol Π_{even} for $\textsc{EvenCharge}^k(H_n)$ as guaranteed by Lemma 5.1 to produce a k-party NOF protocol for the zero/one set-disjointness problem which uses the following definition. In this reduction, we place the set-disjointness variables on the variables labeling some randomly chosen paths in the graph G_n. For the purposes of analyzing the distribution, the paths are chosen to be of length $l = \lceil \frac{\log n}{\log\log n} \rceil$ where $c_1 > 0$ is a constant. The constant c_1 is determined by Proposition 6.6. This is necessary for the proof of Lemma 5.3. For a more thorough discussion of this choice, see section 6.2.

Definition 5.3. *Let $P_l^{(m)}$ be the set of all sequences of m vertex-disjoint length l paths in G_n.*

Lemma 5.2. *Let $m = n^{1/3}/\log n$. For sufficiently large n and for any even charge vector $\vec{c}$, if there is a probabilistic k-party NOF communication complexity protocol Π_{even} for* $\text{EvenCharge}^k(H_n)$ *using s bits, satisfying the conditions in Lemma* 5.1 *for $\mathcal{D}_0$ and $\mathcal{D}_1$, then there is a randomized $(0, 1/3 + \epsilon + o(1))$ error k-party NOF communication complexity protocol Π_{01disj} for zero/one set disjointness on input $\vec{x} \in (\{0,1\}^m)^k$ that uses s bits of communication.*

Proof. Let $\vec{x}$ be an instance of zero/one set disjointness. Protocol Π_{01disj} will call Π_{even} on the graph H_n, on a pair $(\alpha, \vec{c})$ chosen according to the following distribution/experiment:

1. On input $\vec{x}$ with public coins r:
   (a) Using public coins r, choose a random even charge vector $\vec{c} \in \{0,1\}^n$.
   (b) Using public coins r, choose a sequence of m vertex-disjoint length l paths $p_1, \dots p_m$ uniformly at random from $P_l^{(m)}$.
   (c) Using the public coins r, choose $\beta \in Sol(H_n - \bigcup_{j=1}^m p_j, \vec{c})$.
2. For all edges $e \in H_n$, all players other than player i compute α_e^i as follows:
   (a) If $e \in p_j$ for $j \in [m]$, set $\alpha_e^i = x_{i,j}$.
   (b) If $e \in G_n$ and $e \notin \bigcup_{j=1}^m p_j$, choose the vector $\alpha_e^1 \dots \alpha_e^k$ according to the distribution $\mathcal{L}_k(\beta_e)$.
   (c) For the remaining edges $e \in T_n$, set all variables α_e^i for $i \in [k]$ equal to β_e.
3. Return $(\alpha, \vec{c})$.

We write $\mathcal{R}(\vec{x})$ to denote the distribution on assignment/charge pairs produced by reduction Π_{01disj} when given an input $\vec{x}$. The following lemma shows that when $|\cap \vec{x}| = 1$, although $\mathcal{R}(\vec{x})$ is not the same as $\mathcal{D}_1$, $\mathcal{R}(\vec{x})$ is close to the distribution $\mathcal{D}_1$ in the ℓ_1 norm. This is the main technical lemma in the proof. The proof of this lemma can be found in the next section.

Lemma 5.3. *Let $\vec{x} \in (\{0,1\}^m)^k$ and $|\cap \vec{x}| = 1$. Then $||\mathcal{R}(\vec{x}) - \mathcal{D}_1||_1$ is $o(1)$.*

The protocol Π_{01disj} will output 0 if Π_{even} returns "true" and 1 otherwise. If $\cap\vec{x} = \emptyset$, by the above construction, the support of $\mathcal{R}(\vec{x})$ is contained in that of $\mathcal{D}_0$ and thus on $\mathcal{R}(\vec{x})$, Π_{even} must answer "true" and the vector $\vec{x}$ is correctly identified as being disjoint. In the case that $\cap\vec{x}$ contains exactly one element, $\Pr[\Pi_{01disj}(\vec{x}) = 0] \geq 2/3 - \epsilon - o(1)$. This completes the proof of Lemma 5.2. ◻

Reduction from set disjointness to zero/one set disjointness.

Lemma 5.4. *If there is an $(0, \epsilon)$ randomized NOF protocol for the k-party zero-one set-disjointness problem that uses s bits of communication where ϵ is a constant < 1, then there is a $(0, \frac{1}{3})$ randomized NOF protocol for the k-party set-disjointness problem that uses $O(s \log n)$ bits of communication.*

Naturally, our starting point is the well-known result of Valiant and Vazirani [28].

Lemma 5.5 (Valiant–Vazirani). *Let a be a positive integer. Fix a nonempty $S \subseteq \{0,1\}^a$, and choose $w_1, \dots w_a \in \{0,1\}^a$ independently and uniformly. With probability at least $1/4$, there exists $j \in \{0, \dots, a\}$ so that $|\{x \in S \mid \forall i \leq j,\ x \cdot w_i = 0\}| = 1$.*

Proof of Lemma 5.4. Let Π be the protocol for the promise problem. Set $a = \lceil \log n \rceil$. Using public coins, independently and uniformly choose $w_1, \dots w_l \in \{0,1\}^a$. For $j \in \{0, \dots a\}$, the players run the protocol Π, using the following rule for evaluating the input $x_{i,r}$ for $i \in [k], r \in [m]$: Interpret r as a vector in $\{0,1\}^a$, and replace the value of $x_{i,r}$ by zero if for some $j' \leq j$, $w_{j'} \cdot r \neq 0$, and use the value $x_{i,r}$ if for all $j' \leq j$, $w_{j'} \cdot r = 0$. If the protocol Π returns 1, the players halt and output 1; otherwise, the players proceed to round $j + 1$. If no intersection is found after all $a + 1$ rounds, the players announce that the inputs are disjoint.

Clearly, this protocol uses $O(s \log n)$ bits of communication, and by the 0-error property of Π on disjoint inputs, it never outputs 1 when the inputs are disjoint. When the inputs are nondisjoint, the Valiant–Vazirani construction ensures that, with probability at least 1/4, at some round j the protocol Π is used on an input with a unique intersection, and therefore, conditioned on this event, the correct answer is returned with a probability at least $1 - \epsilon$. Therefore, the correct answer is returned with a probability at least $\frac{1}{4} - \frac{\epsilon}{4}$. Because ϵ is bounded away from 1 and the error is one-sided, a constant number of repetitions decreases the probability of error to 1/3. □

Combining the reductions to prove Theorem 4.1.

Proof of Theorem 4.1. By Theorem 3.5 and the definition of $\textsc{OddCharge}^k(H_n)$, if for every $\vec{c} \in \{0,1\}^n$ there is treelike **Th(k-1)** refutation of $TS^k(H_n, \vec{c})$ of size at most S, then there is a $1/n$-error randomized k-party NOF communication complexity protocol for $\textsc{OddCharge}^k(H_n)$ in which at most $O(\log^3 S)$ bits are communicated. By communicating the values of the edges incident to the vertex to be output by this $\textsc{OddCharge}^k(H_n)$ protocol, the players can check that this vertex is indeed in error and not produce a vertex otherwise. This gives a 0-error protocol that outputs the correct answer with a probability at least $1 - 1/n$. By Lemma 5.1 this yields a randomized 0-error k-party NOF protocol Π_{even} for $\textsc{EvenCharge}^k(H_n)$ that uses $O(\log^3 S + \log n)$ bits, produces the correct answer for all inputs in the support of $\mathcal{D}_0$, and for inputs randomly chosen according to $\mathcal{D}_1$ produces a correct answer with a probability at least $2/3 - 1/n$. Applying Lemma 5.2 this yields a $(0, 1/3 + 1/n + o(1))$-error k-party protocol for zero/one set disjointness on $(\{0,1\}^m)^k$ also of complexity $O(\log^3 S + \log n)$. Finally applying Lemma 5.4 yields an error 1/3 randomized k-party NOF protocol for $\textsc{Disj}_{k,m}$ of complexity $O(\log^3 S \log n + \log^2 n)$ bits in total. The deterministic bound is obtained by applying a similar reduction using the other parts of Theorem 3.5. □

6. Proximity of distributions $\mathcal{D}_1$ and $\mathcal{R}(\vec{x})$ when $|\cap \vec{x}| = 1$.

In this section we prove Lemma 5.3, that for $|\cap \vec{x}| = 1$ the distributions $\mathcal{R}(\vec{x})$ and $\mathcal{D}_1$ are close in the ℓ_1 norm. Let $\mu_{\mathcal{D}_1}$ and $\mu_{\mathcal{R}(\vec{x})}$ be their associated probability measures. We will show that, for all but a set of $(\alpha, \vec{c})$ with $\mu_{\mathcal{D}_1}$ measure $o(1)$, $\mu_{\mathcal{D}_1}(\alpha, \vec{c}) = (1 \pm o(1))\mu_{\mathcal{R}(\vec{x})}(\alpha, \vec{c})$.

Given an instance of the set-disjointness variables $\vec{x} = (\{0,1\}^m)^k$, for $j \in [m]$ we say that the *color* of j is the tuple $(x_{1,j}, \dots, x_{k,j}) \in \{0,1\}^k$. By construction, the assignment $\mathcal{R}(\vec{x})$ has color $(x_{1,j}, \dots, x_{k,j})$ on each edge of the path p_j.

Definition 6.1. *Given an ordered sequence of paths $\vec{p} \in P_l^{(m)}$, an $\vec{x} \in (\{0,1\}^m)^k$, and an assignment α, write $\chi(\alpha_{\vec{p}}) = \vec{x}$ if and only if every edge on path p_j has color $(x_{1,j}, \dots, x_{k,j})$ for every $j \in [m]$.*

We first observe that for any $(\alpha, \vec{c})$ with $|\mathrm{Err}(\alpha, \vec{c})| = 2$ the probability $\mu_{\mathcal{D}_1}(\alpha, \vec{c})$ depends only on the number of edges $e \in G_n$ having color 1^k in α.

Definition 6.2. *Let $\phi(a, b) = 2^{-a}(2^k - 1)^{-(a-b)}$.*

Lemma 6.1. *For any $(\alpha, \vec{c})$ with $|\mathrm{Err}(\alpha, \vec{c})| = 2$ and $m_1 = |\{e \in E(G_n) \mid \alpha_e = 1^k\}|$,*

$$\mu_{\mathcal{D}_1}(\alpha, \vec{c}) = \frac{\phi(|E(G_n)|, m_1)}{2^{n-1}\binom{n}{2}}.$$

Proof. Let $U = \mathrm{Err}(\alpha, \vec{c})$. The probability under $\mathcal{D}_1$ that U is chosen to be flipped is $1/\binom{n}{2}$, and, given U, the probability that the charge vector $\vec{c}$ is produced by the experiment is simply the probability that $\vec{c}^U$ is generated by the uniform distribution over all 2^{n-1} many even charge vectors, that is, $2^{-(n-1)}$. Conditioned on the event that U is chosen to be flipped, and that the charge vector is $\vec{c}$, the chance that α labels the edges for the randomly selected element of $Sol(H_n, \vec{c})$ is $2^{-|E(G_n)|}(2^k-1)^{-(|E(G_n)|-m_1)} = \phi(|E(G_n)|, m_1)$. □

Definition 6.3. *For $U \subset V$, with $|U| = 2$, let $P_l^{(m)}(U)$ be the set of all elements of $P_l^{(m)}$ that have a path whose end points are U.*

Now consider the measure $\mu_{\mathcal{R}(\vec{x})}(\alpha, \vec{c})$. Let $\{i\} = \cap \vec{x} \subseteq [n]$, $U = \text{Err}(\alpha, \vec{c})$, with $|U| = 2$, and $m_1 = |\{e \in E(G_n) \mid \alpha_e = 1^k\}|$. By the definition of $\mathcal{R}(\vec{x})$,

$$\begin{aligned}\mu_{\mathcal{R}(\vec{x})}(\alpha, \vec{c}) = &\Pr_{\vec{p} \in P_l^{(m)}}[\text{Ends}(p_i) = \text{Err}(\alpha, \vec{c}) \wedge \chi(\alpha_{\vec{p}}) = \vec{x}] \\ &\cdot \Pr_{\substack{\vec{c'} \in \{0,1\}^n \\ \alpha' \in \mathcal{L}_k(Sol(H_n - \vec{p}, \vec{c'}))}}[\alpha' = \alpha_{G_n - \vec{p}} \text{ and } \vec{c'} = \vec{c}] \\ = &\Pr_{\vec{p} \in P_l^{(m)}}[\text{Ends}(p_i) = \text{Err}(U)] \cdot \Pr_{\vec{p} \in P_l^{(m)}(U)}[\chi(\alpha_{\vec{p}}) = \vec{x}] \\ &\cdot \phi(|E(G_n)| - ml, m_1 - l)/2^{n-1}.\end{aligned}$$

Observe that p_i is a uniformly chosen element of P_l, and we can analyze the first term using the following property of random paths on Lubotzky–Phillips–Sarnak (LPS) expanders proved as part of Lemma 6.9 in section 6.2.2.

Lemma 6.2. *For $u \neq v \in V(G_n)$ and $l \geq c_1 \log n / \log \log n$, $\Pr_{p \in P_l}[\text{Ends}(p) = \{u, v\}] = (1 \pm o(1))/\binom{n}{2}$.*

Thus

$$\begin{aligned}\mu_{\mathcal{R}(\vec{x})}(\alpha, \vec{c}) &= (1 \pm o(1)) \frac{\phi(|E(G_n)| - ml, m_1 - l)}{\binom{n}{2} 2^{n-1}} \cdot \Pr_{\vec{p} \in P_l^{(m)}(U)}[\chi(\alpha_{\vec{p}}) = \vec{x}] \\ &= (1 \pm o(1)) \frac{\mu_{\mathcal{D}_1}(\alpha, \vec{c})}{\phi(ml, l)} \cdot \Pr_{\vec{p} \in P_l^{(m)}(U)}[\chi(\alpha_{\vec{p}}) = \vec{x}].\end{aligned}$$

It follows that we will obtain the desired result if we can show that, for all but a $o(1)$ measure of $(\alpha, \vec{c})$ under $\mu_{\mathcal{D}_1}$,

$$\Pr_{\vec{p} \in P_l^{(m)}(U)}[\chi(\alpha_{\vec{p}}) = \vec{x}] = (1 \pm o(1))\phi(ml, l) = (1 \pm o(1))2^{-ml}(2^k - 1)^{-(m-1)l},$$

where $U = \text{Err}(\alpha, \vec{c})$. In the case that this happens, we say that $(\alpha, \vec{c})$ is *well-distributed for $\vec{x}$*.

Using the second moment method we prove the following lemma, which shows that, for all but a $o(1)$ measure of $(\alpha, \vec{c})$ under $\mu_{\mathcal{D}_1}$, $(\alpha, \vec{c})$ is indeed well-distributed for $\vec{x}$. The detailed proof is given in section 6.1.

Lemma 6.3. *Let $m \leq n^{1/3}/\log n$, $l = 2\lceil c_1 \log n / \log \log n \rceil$, and $\vec{x} \in (\{0,1\}^m)^k$, with $|\cap \vec{x}| = 1$. For almost all $U \subset [n]$, with $|U| = 2$,*

$$\Pr_{(\alpha, \vec{c}) \in \mathcal{D}_1}[(\alpha, \vec{c}) \textit{ is well-distributed for } \vec{x} \mid \text{Err}(\alpha, \vec{c}) = U] = 1 - o(1).$$

Lemma 5.3 follows from this almost immediately.

Proof of Lemma 5.3. Let $\vec{x} \in (\{0,1\}^m)^k$ and $|\cap \vec{x}| = 1$. By Lemma 6.3 and the preceding argument, for all $U \in \binom{[n]}{2}$ except for a set B that forms an $o(1)$ fraction of $\binom{[n]}{2}$,

$$\Pr_{(\alpha,\vec{c})\in\mathcal{D}_1}[\mu_{\mathcal{R}(\vec{x})}(\alpha,\vec{c}) = (1\pm o(1))\mu_{\mathcal{D}_1}(\alpha,\vec{c}) \mid \mathrm{Err}(\alpha,\vec{c}) = U] = 1 - o(1).$$

By Lemma 6.2, $\Pr_{(\alpha,\vec{c})\in\mathcal{D}_1}[\mathrm{Err}(\alpha,\vec{c}) \in B] = o(1)$. Therefore by summing over distinct choices of U, we obtain that, with probability $1-o(1)$ over $(\alpha,\vec{c}) \in \mathcal{D}_1$, $\mu_{\mathcal{R}(\vec{x})}(\alpha,\vec{c}) = (1\pm o(1))\mu_{\mathcal{D}_1}(\alpha,\vec{c})$. This is equivalent to the desired conclusion that $||\mathcal{D}_1 - \mathcal{R}(\vec{x})||_1$ is $o(1)$. □

6.1. Most $(\alpha,\vec{c})$ are well-distributed. In this section we use the second moment method to prove Lemma 6.3. For this purpose we will need the following property of the LPS expander graphs G_n, proved in section 6.2, which will allow us to show that the correlations considered in the second moment method are low.

Definition 6.4. *For $\vec{p},\vec{q} \in P_l^{(m)}$ we write $\vec{p} \sim_s \vec{q}$ when $\vec{p}$ and $\vec{q}$ share exactly s edges. Let $\gamma > 0$ be a positive real number. We say that $U \subset V(G_n)$ is* γ-nice *if, for all $s \geq 0$, $\Pr_{\vec{p},\vec{q}\in P_l^{(m)}(U)}[\vec{p} \sim_s \vec{q}] \leq \gamma^s$.*

Theorem 6.4 (proved in section 6.2). *Suppose that $m \leq n^{1/3}/\log n$ and $l = 2\lceil c_1 \log n/\log\log n\rceil$. There are constants $c > 0$ and c' such that, for all but a $o(1)$ fraction of sets $U = \{u,v\} \subset V(G_n)$, for all $\vec{q} \in P_l^{(m)}(U)$ and every integer $s \geq 0$,*

$$\Pr_{\vec{p}\in P_l^{(m)}(U)}[\vec{p} \sim_s \vec{q}] \leq (c'/(\log\log n)^{1/4} + (\log n)^{-c})^s;$$

i.e., almost every $U \in V^{(2)}$ is $(c'/(\log\log n)^{1/4} + 1/\log^c n)$-nice.

We now use this in our application of the second moment method to prove that most $(\alpha,\vec{c})$ pairs are well-distributed:

Lemma 6.5. *Let $m \leq n^{1/3}/\log n$, $l = 2\lceil c_1 \log n/\log\log n\rceil$, $\vec{x} \in (\{0,1\}^m)^k$, with $|\cap \vec{x}| = 1$, and $|U| = 2$. If U is γ-nice, with $\gamma = o(2^{-k})$, then*

$$\Pr_{(\alpha,\vec{c})\in\mathcal{D}_1}[(\alpha,\vec{c}) \textit{ is well-distributed for } \vec{x} \mid \mathrm{Err}(\alpha,\vec{c}) = U] = 1 - o(1)$$

Proof. For each $\vec{p} \in P_l^{(m)}(U)$, let $X_{\vec{p}}$ denote the indicator variable for the event that $\chi(\alpha_{\vec{p}}) = \vec{x}$.

We now calculate $E_{(\alpha,\vec{c})\in\mathcal{D}_1}[X_{\vec{p}}]$. For $(\alpha,\vec{c})$ chosen according to $\mathcal{D}_1$, the assignment $\alpha_{\vec{p}}$ is distributed according to $(\mathcal{L}_k)^{ml}$; therefore, since for $\chi(\alpha_{\vec{p}})$ to equal $\vec{x}$, $\alpha_{\vec{p}}$ must have precisely l edges whose color is 1^k and $l(m-1)$ edges whose color is a lift of label 0,

$$E_{(\alpha,\vec{c})\in\mathcal{D}_1}[X_{\vec{p}}] = \Pr_{(\alpha,\vec{c})\in\mathcal{D}_1}[X_{\vec{p}} = 1] = \phi(ml,l) = 2^{-ml}(2^k-1)^{-(m-1)l}.$$

Let $X = \sum_{\vec{p}\in P_l^{(m)}(U)} X_{\vec{p}}$. X is the random variable denoting the number of sequences $\vec{p} \in P_l^{(m)}(U)$ for which $\chi(\alpha_{\vec{p}}) = \vec{x}$. By the linearity of expectation, $E_{(\alpha,\vec{c})}[X] = \phi(ml,l)\cdot|P_l^{(m)}(U)|$.

We use the second moment method to show that X is concentrated near its expectation. For $\vec{p},\vec{q} \in P_l^{(m)}(U)$, the random variables $X_{\vec{p}}$ and $X_{\vec{q}}$ are correlated if and only if $\vec{p}$ and $\vec{q}$ share an edge. Because U is γ-nice, $\Pr_{\vec{p},\vec{q}\in P_l^{(m)}(U)}[\vec{p} \sim_s \vec{q}] \leq \gamma^s$.

When $X_{\vec{p}} = 1$, the colors of all edges of $\vec{p}$ are determined. Therefore given $X_{\vec{p}} = 1$, if $\vec{p} \sim \vec{q}$, either some edge that $\vec{p}$ and $\vec{q}$ share ensures that $X_{\vec{q}} = 0$ or the probability that $X_{\vec{p}} = X_{\vec{q}} = 1$ is nonzero. In the latter case consider $G' = \bigcup_{i=1}^{m}(p_i \cup q_i)$, which contains $2ml - s$ edges. Because the marginal distribution of α to the edges of G' independently assigns each e of G' a label using the distribution $\mathcal{L}_k$ (per Definition 5.1), we have that the probability that $\chi(\alpha_{\vec{p}}) = \chi(\alpha_{\vec{q}}) = \vec{x}$ is larger than $[\phi(ml,l)]^2$ by a factor of either 2 or $2(2^k - 1)$ per shared edge depending on whether that edge has a label 1 or 0.

Let $D = \sum_{\vec{p} \sim_s \vec{q}} \Pr_{(\alpha,\vec{c})}[X_{\vec{p}} = X_{\vec{q}} = 1]$.

$$\begin{aligned}
D &= \sum_{s=1}^{ml} \sum_{\vec{p}\sim_s \vec{q}} \Pr_{(\alpha,\vec{c})}[X_{\vec{p}} = X_{\vec{q}} = 1] \\
&\le \sum_{s=1}^{ml} \sum_{\vec{p}\sim_s \vec{q}} (2(2^k-1))^s \Pr_{(\alpha,\vec{c})\in\mathcal{D}_1}[X_{\vec{p}} = 1] \Pr_{(\alpha,\vec{c})\in\mathcal{D}_1}[X_{\vec{q}} = 1] \\
&= \sum_{s=1}^{ml} \sum_{\vec{p}\sim_s \vec{q}} (2(2^k-1))^s (\phi(ml,l))^2 \\
&= \sum_{s=1}^{ml} |P_l^{(m)}(U)|^2 \Pr_{\vec{p},\vec{q}\in P_l^{(m)}(U)}[\vec{p} \sim_s \vec{q}](2(2^k-1))^s(\phi(ml,l))^2 \\
&= (|P_l^{(m)}(U)| \cdot \phi(ml,l))^2 \sum_{s=1}^{lm} \Pr_{\vec{p},\vec{q}\in P_l^{(m)}(U)}[\vec{p} \sim_s \vec{q}](2(2^k-1))^s \\
&= (E_{(\alpha,\vec{c})\in\mathcal{D}_1}[X])^2 \sum_{s=1}^{ml} \Pr_{\vec{p},\vec{q}\in P_l^{(m)}(U)}[\vec{p} \sim_s \vec{q}](2(2^k-1))^s \\
&\le (E_{(\alpha,\vec{c})\in\mathcal{D}_1}[X])^2 \sum_{s=1}^{ml} \gamma^s (2(2^k-1))^s.
\end{aligned}$$

Since $\gamma = o(2^{-k})$ by hypothesis, $\sum_{s=1}^{\infty} \gamma^s(2(2^k-1))^s$ is $o(1)$, and thus D is $o((E_{(\alpha,\vec{c})\in\mathcal{D}_1}[X])^2)$. Therefore, $E_{(\alpha,\vec{c})}(X^2) = D + E_{(\alpha,\vec{c})}(X) = o((E_{(\alpha,\vec{c})}[X])^2) + E_{(\alpha,\vec{c})}(X)$, and by the second moment method,

$$\Pr_{(\alpha,\vec{c})\in\mathcal{D}_1}[|X - E_{(\alpha,\vec{c})\in\mathcal{D}_1}(X)| \ge \epsilon E_{(\alpha,\vec{c})\in\mathcal{D}_1}(X)] \le \frac{D + E_{(\alpha,\vec{c})\in\mathcal{D}_1}[X]}{\epsilon^2 E_{(\alpha,\vec{c})}(X)^2} = o(1).$$

By choosing ϵ as an appropriate function that is $o(1)$, we obtain that, with probability $1 - o(1)$ in the choice of $(\alpha,\vec{c}) \in \mathcal{D}_1$, $X = (1 \pm o(1))\phi(ml,l) \cdot |P_l^{(m)}(U)|$ and therefore, with probability $1 - o(1)$ in $(\alpha,\vec{c})$, $\Pr_{\vec{p}\in P_l^{(m)}(U)}[\chi(\alpha_{\vec{p}}) = \vec{x}] = (1 \pm o(1))\phi(ml,l)$ and thus $(\alpha,\vec{c})$ is well-distributed for $\vec{x}$. □

Proof of Lemma 6.3. Let $\vec{x} \in (\{0,1\}^m)^k$ and $|\cap \vec{x}| = 1$. By Theorem 6.4 there is a $\delta > 0$ so that for all but a $o(1)$ fraction of sets $U \subset V(G_n)$, with $|U| = 2$, U is γ-nice for $\gamma = c''/(\log\log n)^{1/4}$ for some constant c'' and γ is $o(2^{-k})$. Therefore, $\Pr_{(\alpha,\vec{c})\in\mathcal{D}_1}[\mathrm{Err}(\alpha,\vec{c}) \text{ is } \gamma\text{-nice}] = 1 - o(1)$, and by Lemma 6.5, $\Pr_{(\alpha,\vec{c})\in\mathcal{D}_1}[(\alpha,\vec{c})$ is well-distributed for $\vec{x} \mid \mathrm{Err}(\alpha,\vec{c}) = U] = 1 - o(1)$. □

6.2. Graph theoretic properties of LPS expanders.

6.2.1. The LPS expanders. In the analysis of $R(\vec{x})$, we want the end points of the random paths in G_n to be almost uniformly distributed. We base our proof of this upon the fact that the end points of random walks in expander graphs are almost uniformly distributed (Proposition 6.6). Since a walk is allowed to repeat vertices but a path does not repeat vertices, the length of the walk is too large with respect to the degree, and it is very likely that a random walk will not be a path. To transfer Proposition 6.6 from walks to paths, we use a graph in which random walks of length l will have their end points almost uniformly distributed, but the walks are short enough with respect to the degree so that a random walk is very likely to be a path. We use the LPS expanders. The crucial properties of the expander graphs G_n constructed in [22] that we need are as follows:

1. G_n is regular of degree $\Delta = \Theta(\log n)$.
2. G_n is connected and nonbipartite.
3. The second eigenvalue of G_n is $O(\sqrt{\log n})$.
4. The girth of G_n is $\Omega(\log n/\log\log n)$.

A walk in G_n is chosen by selecting a start node and repeatedly following one of the Δ edges adjacent to the current node.

Proposition 6.6. *There exists $c_1 > 0$ so that for every $u, v \in V(G_n)$ a random walk in G_n of length $l \geq c_1 \log n/\log\log n$ starting at u ends at vertex v with a probability at least $1/n - 1/n^2$ and at most $1/n + 1/n^2$.*

We consider random walks and random paths in the G_n graphs of a fixed length $l = l(n) = 2\lceil c_1 \log n/\log\log n\rceil$ that is twice the minimum length specified in Proposition 6.6 so that their midpoints are nearly uniformly distributed.

6.2.2. Approximating paths by walks.

Remark 1. In principle one might replace disjoint paths in the definition of Π_{01disj} by disjoint walks of the same length, conditioned on each having distinct end points. However, in that case it would be overwhelmingly likely that many walks will repeat edges, and therefore, as graphs, they would contain different numbers of edges. This would significantly complicate the second moment argument of Lemma 6.5.

We show that, because G_n is expanding and has high girth, random walks in G_n not only mix well but they are paths almost surely as well. We state some folklore properties of random walks and observe how they translate into properties of random paths.

For $v \in V(G_n)$, let $W_l(v)$ be the set of all Δ^l walks of length l in G_n starting at v, and let $P_l(v)$ be the set of all paths of length l in G_n with one end point v. Let $\mu_{W_l(v)}$ be the measure given by a uniform distribution over $W_l(v)$, and let $\mu_{P_l(v)}$ be the measure given by a uniform distribution over $P_l(v)$.

Lemma 6.7. *There exists a universal constant c_3 so that, for every $v \in V(G_n)$ and for each path $p \in P_l(v)$, $(1 - c_3/\log\log n)\mu_{P_l(v)}(p) \leq \mu_{W_l(v)}(p) \leq \mu_{P_l(v)}(p)$. Moreover, for w uniformly chosen from $W_l(v)$ the probability that w is not a path is at most $c_3/\log\log n$.*

Proof. Observe that every $p \in P_l(v)$ has equal measure under $\mu_{W_l(v)}$ so $\mu_{W_l(v)}(p) \leq \mu_{P_l(v)}(p)$, and, moreover, $\mu_{W_l(v)}(p) = \mu_{P_l(v)}(p)\mu_{W_l(v)}(P_l(v))$.

Set $g = girth(G_n)$. By the properties of G_n, $g \geq c_0 \log n/\log\log n$ for some constant $c_0 > 0$ and its degree $\Delta \geq c_2 \log n$ for some constant $c_2 \geq 0$. Notice that for any walk w of length l each vertex in w can have at most $l/(g-3)$ many neighbors also in w. (If u is a vertex in w that has two neighbors u' and u'' in G_n within distance $g-3$ on w, then there is a cycle of length $g-1$ in $w \cup \{(u,u'),(u,u'')\}$ which is a subgraph of G_n.) Therefore

$$\mu_{W_l(v)}(P_l(v)) \geq \left(\frac{\Delta - l/(g-3)}{\Delta}\right)^l \ \geq\ 1 - \frac{l^2}{\Delta(g-3)}$$
$$\geq 1 - \frac{2\lceil c_1 \log n/\log\log n\rceil^2}{c_2 \log n \cdot (c_0 \log n/\log\log n - 3)} \ \geq\ 1 - c_3/\log\log n$$

for some constant c_3. □

The following are folklore properties of random walks in G_n.

PROPOSITION 6.8. *Let W_l be the set of all walks of length l in G_n.*

1. *For each $v \in V(G_n)$, $\Pr_{w\in W_l}[v \in V(w)] \leq (l+1)/n$.*
2. *For each $u \neq v \in V(G_n)$, $\Pr_{w\in W_l}[\text{Ends}(w) = \{u, v\}] = (1 \pm 2/n)/\binom{n}{2}$.*

Proof. There is a sequence of $l+1$ vertices (not necessarily distinct) on each walk w in W_l and precisely Δ^l walks in which v is the ith vertex in w. Therefore, in total there are at most $(l+1)\Delta^l$ walks with $v \in V(w)$. (This is an overcount since v may appear more than once in w.) Since there are precisely $n\Delta^l$ random walks in G_n of length l, $\Pr_{w\in W_l}[v \in V(w)] \leq (l+1)/n$.

By Proposition 6.6 the chance that a particular pair of distinct vertices $\{u, v\}$ appear as end points of w is $\frac{2}{n}(1/n \pm 1/n^2)$, which is $(1 \pm 2/n)/\binom{n}{2}$. □

We obtain the following easy corollary which includes a proof of Lemma 6.2.

LEMMA 6.9. *Let P_l be the set of all paths in G_n of length l.*

1. *Let $V' \subseteq V(G_n)$. There exists a constant c so that*

$$\Pr_{p\in P_l}[V(p) \cap V' \neq \emptyset] \leq (1 + c/\log\log n)\frac{|V'|(l+1)}{n}.$$

2. *Let $u \neq v \in V(G_n)$. Then $\Pr_{p\in P_l}[\text{Ends}(p) = \{u, v\}] = (1 \pm o(1))/\binom{n}{2}$.*

Proof. By Proposition 6.8, for w a randomly chosen walk of length l in G_n,

$$\Pr_{w\in W_l}[V(w) \cap V' \neq \emptyset] \leq \frac{|V'|(l+1)}{n},$$

and by Lemma 6.7, $\Pr_{w\in W_l}[w \text{ is a path}] \geq 1 - c_3/\log\log n$. The random distribution of paths p of length l in G_n is the same as the random distribution of walks w of length l in G_n conditioned on w being a path. Therefore

$$\begin{aligned}\Pr_{p\in P_l}[V \cap V(p) \neq \emptyset] &= \Pr_{w\in W_l}[V \cap V(w) \neq \emptyset \mid w \text{ is a path}]\\ &\leq \frac{|V|(l+1)}{(1 - c_3/\log\log n)n}\\ &\leq (1 + c/\log\log n)\frac{|V|(l+1)}{n}\end{aligned}$$

for some constant c.

For $u \neq v \in V(G_n)$, by Lemma 6.7 $\Pr_{p\in P_l}[\text{Ends}(p) = \{u, v\}]$ is within a $1 \pm o(1)$ factor of $\Pr_{w\in W_l}[\text{Ends}(w) = \{u, v\}]$, and by Proposition 6.8 the latter is $(1 \pm o(1))/\binom{n}{2}$, which yields the desired property. □

6.2.3. The proof of Theorem 6.4. In this subsection we prove Theorem 6.4. We will actually prove a slightly stronger result in which $\vec{q} \in P_l^{(m)}(U)$ is replaced by any subgraph of G_n with at most $m(l+1)$ vertices and a maximum degree at most 2.

It will be convenient to consider sequences of length l paths P_l^m that are not necessarily vertex-disjoint. Let $\mu_{P_l^{(m)}}$ be the uniform measure on $P_l^{(m)}$ and $\mu_{P_l^m}$ be the uniform distribution on P_l^m.

Lemma 6.10. *Suppose that $m \le n^{1/3}/\log n$ and $l = 2\lceil c_1 \log n/\log\log n\rceil$. For any $\vec{p} \in P_l^{(m)}$, $(1-o(1))\mu_{P_l^{(m)}}(\vec{p}) \le \mu_{P_l^m}(\vec{p}) \le \mu_{P_l^{(m)}}(\vec{p})$.*

Proof. Conditioned on the paths in $\vec{p} \in P_l^m$ being vertex-disjoint, $\mu_{P_l^m}$ is uniform over $P_l^{(m)}$. By Lemma 6.9, the probability that the ith path shares a vertex with paths $p_1, \ldots, p_{i-1}$ is at most $(1 + c/\log\log n)(l+1)^2(m-1)/n \le 2l^2m/n$, and the probability that the paths in P_l^m are not vertex-disjoint is at most $2l^2m^2/n \le 1/n^{1/3}$. □

We first observe that if we required only that $\vec{p} \in P_l^{(m)}$ rather than $\vec{p} \in P_l^{(m)}(U)$—i.e., we had no requirement that one path in $\vec{p}$ have its end points in U—then the exponentially decaying bound on intersection size of Theorem 6.4 would be relatively easy.

Lemma 6.11. *Suppose that $m \le n^{1/3}/\log n$ and $l = 2\lceil c_1 \log n/\log\log n\rceil$. There is some constant $c \ge 0$ such that, for all subgraphs G' of G_n with at most $m(l+1)$ vertices and every integer $s \ge 0$,*

$$\Pr_{\vec{p} \in P_l^{(m)}}[|E(\cup\vec{p}) \cap E(G')| \ge s] \le (\log n)^{-cs}.$$

Proof. For $\vec{p} \in P_l^{(m)}$, because each component of $\vec{p}$ is a path of length l, if $|E(\cup\vec{p}) \cap E(G')| \ge s$, then there are at least $\lceil s/l\rceil$ paths p_i in $\vec{p}$ that share an edge (and therefore a vertex) with G'. By Lemma 6.9, the probability that a random p_i from P_l shares a vertex with G' is at most $(1 + c/\log\log n)(l+1)^2m/n < 2l^2m/n$. Therefore for elements of P_l^m, the probability that there are least $r = \lceil s/l\rceil$ such paths is at most $\binom{m}{r}(2l^2m/n)^r < (2l^2m^2/n)^r/2$. By Lemma 6.10, the probability that this happens for elements of $P_l^{(m)}$ is at most $(2l^2m^2/n)^r \le n^{s/(3l)} = (\log n)^{-cs}$ for some constant $c > 0$. □

The major complication of the proof of Theorem 6.4 is the assumption that $\vec{p}$ contains a path with end points u and v for $U = \{u, v\}$, $u \ne v$. We base the analysis of paths with end points U on the analysis of walks with end points U. For some sets U, for example, if u and v are adjacent in G_n, the distributions of random walks and random paths with end points U may not be close to each other.[3] We will see that, for most choices of U, the probabilities under the two distributions are close to each other, and this will be enough to obtain the bound required by Theorem 6.4.

Definition 6.5. *For $U = \{u, v\} \in V(G_n)$ let $W_l(U)$ be the set of all walks in G_n of length n that have end points U.*

Lemma 6.12. *There is a constant c_4 such that, for all but at most a $c_4/\log\log n$ fraction of pairs $u \ne v \in V(G_n)$,*

$$\Pr_{w \in W_l(\{u,v\})}[w \textit{ is a path}] \ge 2/3.$$

Proof. By Lemma 6.7,

$$\Pr_{w \in W_l}[w \text{ is not a path}] \le c_3/\log\log n.$$

[3] Even in these cases the distributions may be sufficiently close, but we do not need to analyze them.

Therefore by definition,

$$\sum_{u\neq v\in V(G_n)} \Pr_{w\in W_l}[\mathrm{Ends}(w)=\{u,v\}] \Pr_{w\in W_l(\{u,v\})}[w \text{ is not a path}] \leq c_3/\log\log n.$$

By Proposition 6.8, $\Pr_{w\in W_l}[\mathrm{Ends}(w)=\{u,v\}] \geq (1-2/n)\binom{n}{2}^{-1}$, and thus

$$(1-2/n)\binom{n}{2}^{-1} \sum_{u\neq v\in V(G_n)} \Pr_{w\in W_l(\{u,v\})}[w \text{ is not a path}] \leq c_3/\log\log n,$$

which says that the expected value

$$E_{u\neq v\in V(G_n)}(\Pr_{w\in W_l(\{u,v\})}[w \text{ is not a path}]) \leq \frac{c_3/\log\log n}{(1-2/n)}.$$

We now apply Markov's inequality to obtain that the fraction of pairs $u \neq v \in V(G_n)$ for which $\Pr_{w\in W_l(\{u,v\})}[w$ is not a path$] \geq 1/3$, is at most $\frac{c_3/\log\log n}{(1-2/n)/3} \leq c_4/\log\log n$ for some constant c_4. ∎

Bounding intersection size of random walks. Lemma 6.12 will allow us to use the following analysis involving a random walk with end points in U rather than a random path.

LEMMA 6.13. *Let G' be a subgraph of G_n with the property that every vertex has a degree at most d in G'. For fixed $v \in V(G_n)$,*

$$\Pr_{w\in W_l(v)}[|E(w)\cap E(G')| \geq s] \leq \binom{l}{s}\left(\frac{d}{\Delta}\right)^s.$$

Proof. There are at most $\binom{l}{s}$ many choices of steps in the random walk in which the first s shared edges can occur. Fix some such set of steps $S \subseteq [l]$. For each $i \in S$ a necessary condition for the ith edge in the walk to lie in $E(G')$ is that the end point u after step $i-1$ must lie in $V(G')$. Since $\deg_{G'}(u) \leq d$, given that $u \in V(G')$, the probability that the ith edge lies in $E(G')$ is then at most d/Δ. That is, conditioned on a shared edge in each of the first j elements in S, the chance of a shared edge in the $j+1$st element in S is at most d/Δ because every vertex has a degree at most d in G'. This yields a total probability at most $\binom{l}{s}(d/\Delta)^s$ as required. ∎

In order to analyze the random walks in $W_l(U)$ we need more than the result of Lemma 6.13, since it constrains only one end point of the random walk rather than both end points. We can view each half of a random walk in which both end points are constrained as two random walks of half the length with only one end point constrained. (Obviously, these two half-length walks are highly correlated.)

LEMMA 6.14. *Let $l = 2\lceil c_1 \log n/\log\log n\rceil$. Let G' be a subgraph of G_n in which every vertex has a degree at most d. For $u \neq v \in V(G_n)$,*

$$\Pr_{w\in W_l(\{u,v\})}[|E(w)\cap E(G')| \geq s] < \left(\frac{2dl}{\Delta}\right)^{s/2}.$$

Proof. Without loss of generality, walk $w \in W_l(\{u,v\})$ starts at u and ends at v. Let $l' = l/2$. Let $w = (w_u, w_v)$, where w_u and w_v each have length l'. We first observe that w_u is nearly uniformly distributed in $W_{l'}(u)$:

Let $w^* \in W_{l'}(u)$, and let v^* be the end of w^*.

$$\begin{aligned}&\Pr_{w \in W_l(\{u,v\})}[w_u = w^* \mid w \text{ starts at } u]\\ &= \frac{\Pr_{w \in W_l(u)}[w_u = w^* \text{ and } w_v, \text{ starts at } v^*, \text{ ends at } v]}{\Pr_{w \in W_l(u)}[w \text{ ends at } v]}\\ &= \frac{\Pr_{w_u \in W_{l'}(u)}[w_u = w^*] \cdot \Pr_{w_v \in W_{l'}(v^*)}[w_v \text{ ends at } v]}{\Pr_{w \in W_l(u)}[w \text{ ends at } v]}.\end{aligned}$$

Clearly $\Pr_{w_u \in W_{l'}(u)}[w_u = w^*] = \Delta^{-l'} = \Delta^{-l/2}$, and since $l > l' \geq c_1 \log n / \log\log n$ by Proposition 6.6, both $\Pr_{w_v \in W_{l'}(v^*)}[w_v \text{ ends at } v]$ and $\Pr_{w \in W_l(u)}[w \text{ ends at } v]$ are $1/n \pm 1/n^2$; thus

$$\Pr_{w \in W_l(\{u,v\})}[w_u = w^* \mid w \text{ starts at } u] = (1 \pm O(1/n))\Delta^{-l/2}.$$

Since G_n is a *regular* undirected graph, a length l random walk from u to v has the same distribution as a length l random walk from v to u. Thus by symmetry with the above argument, within a $1 \pm O(1/n)$ factor, w_v is distributed as a (nearly) uniform random walk of length l' starting at v.

Now if there are a total of s edges in common between w and G', then at least $\lceil s/2 \rceil$ must be shared between G' and one of the two halves of w, w_u and w_v. By Lemma 6.13 and the above argument each of these probabilities is at most $(1 + O(1/n))(\frac{dl'}{\Delta})^{\lceil s/2 \rceil}$, and the total probability is at most $2(1 + O(1/n))(\frac{dl}{2\Delta})^{\lceil s/2 \rceil} \leq (2\frac{dl}{\Delta})^{\lceil s/2 \rceil}$. □

Deriving the bound.

LEMMA 6.15. *Let $l = 2\lceil c_1 \log n / \log\log n \rceil$ and $m \leq n^{1/3}/\log n$. For any fixed subgraph G' of G_n with at most $m(l+1)$ vertices and a maximum degree at most 2, and any set $U = \{u,v\} \subset V(G_n)$,*

$$\Pr_{(w,\vec{p}))\in W_l(U) \times P_l^{m-1}}[|(E(w) \cup E(\vec{p})) \cap E(G')| \geq s] \leq (c''/\log\log n)^{s/4} + (\log n)^{-cs/2}.$$

Proof. If there are s edge intersections between $E(w) \cup E(\vec{p})$ and G', then at least $s/2$ of them occur in either w or $\vec{p}$. Lemma 6.14 implies that $\Pr_{w \in W_l(U)}[|E(w) \cap E(G')| \geq s/2] \leq \left(\frac{4l}{\Delta}\right)^{s/4} \leq (c''/\log\log n)^{s/4}$.

By Lemma 6.11, $\Pr_{\vec{p} \in P_l^{m-1}}[|E(\vec{p}) \cap E(G')| \geq s/2] \leq \Pr_{\vec{p} \in P_l^m}[|E(\vec{p}) \cap E(G')| \geq s/2] \leq (\log n)^{-cs/2}$. □

We now obtain Theorem 6.4.

LEMMA 6.16. *Suppose that $m \leq n^{1/3}/\log n$ and $l = 2\lceil c_1 \log n/\log\log n \rceil$. For all but a $c_4/\log\log n$ fraction of all $U = \{u,v\}$, $u \neq v \in V(G_n)$, there are constants $c, c' > 0$ such that, for all subgraphs G' of G_n with at most $m(l+1)$ vertices and a maximum degree 2 and for every integer $s \geq 0$,*

$$\Pr_{\vec{p} \in P_l^{(m)}(U)}[|E(\cup\vec{p}) \cap E(G')| \geq s] \leq ((c'/\log\log n)^{1/4} + (\log n)^{-c})^s.$$

Proof. By Lemma 6.12, for all but a $c_4/\log\log n$ fraction of U, $\Pr_{w \in W_l(U)}[w \text{ is a path}] \geq 2/3$. For any such U, since the distribution of $w \in W_l(U)$ conditional on w being a path is uniform over $P_l(U)$, the measure of any event on $P_l(U) \times P_l^{m-1}$ is at most $3/2$ times that on $W_l(U) \times P_l^{m-1}$. Further, by the same argument as Lemma 6.10, the probability that the paths in $\vec{p}$ chosen from $P_l(U) \times P_l^{m-1}$ are vertex-disjoint is at least $1-o(1)$ conditioned on being vertex-disjoint the distribution of $\vec{p}$ is uniform over $P_l^{(m)}(U)$. Therefore the measure of any event on $P_l^{(m)}(U)$ is at most $(1+o(1))3/2 \leq 2$ times that on $W_l(U) \times P_l^{m-1}$. Applying Lemma 6.15 and adjusting constants c and c' yields the bound. $\square$

7. Discussion. There are a couple of interesting open problems related to our work beyond the natural problem of the communication complexity of DISJ_k.

The first regards *automatizability* and the existence of *separation oracles*. In [21] it was shown that if a system of 0/1 inequalities has a rank $\leq d$ LS refutation, then the system of inequalities possesses a separation oracle that runs in time $n^{O(d)}$. (A separation oracle is a procedure that takes a polytope P and a point $\vec{x}$ and returns either "true" if $\vec{x} \in P$ or a hyperplane separating $\vec{x}$ and P.) Does semantic $\mathbf{Th(k)}$ have an efficiently computable separation oracle as LS does? A refutation system $\mathcal{R}$ is said to be *automatizable* ([6], cf. [2]) if there is an algorithm that, given unsatisifable CNF Ψ, the algorithm finds a refutation of Ψ in time $S^{O(1)}$, where S is the minimum size of an $\mathcal{R}$ refutation of ψ. The question of the existence of a separation oracle for $\mathbf{Th(k)}$ is closely related to whether or not $\mathbf{Th(k)}$ is automatizable, and we conjecture that the answer to both questions is negative.

The second question is whether or not it is possible to extend our lower bounds to other tautologies that would imply inapproximability results for polynomial-time $\mathbf{Th(k)}$-based algorithms. For example, if we could prove superpolynomial lower bounds for treelike $\mathbf{Th(k)}$ proofs of random 3CNF formulas, this would imply inapproximability results for $\mathbf{Th(k)}$-based linear programming algorithms for MaxSAT [7]. Of course, lower bounds for random 3CNF formulas are open for the $\mathbf{Th(1)}$ systems and even for treelike cutting planes with unary coefficients. A first step towards analyzing random 3CNFs in the $\mathbf{Th(k)}$ systems would be to improve the analysis of this paper to apply to a graph of degree 3 rather than one of degree $\Theta(\log n)$.

Acknowledgment. We are indebted to Avi Wigderson for helpful discussions and insights.

REFERENCES

[1] S. Arora, B. Bollobás, L. Lovász, and I. Tourlakis, *Proving integrality gaps without knowing the linear program*, Theory of Computing, 2 (2006), pp. 19–51.

[2] P. Beame and T. Pitassi, *Current Trends in Theoretical Computer Science: Entering the* 21*st Century*, in Propositional Proof Complexity: Past, Present, and Future, G. Paun, G. Rozenberg, and A. Salomaa, eds., World Scientific Publishing, 2001, pp. 42–70.

[3] P. Beame, T. Pitassi, N. Segerlind, and A. Wigderson, *A strong direct product theorem for corruption and the multiparty communication complexity of disjointness*, Computational Complexity, 15 (2006), pp. 391–432.

[4] A. Bockmayr, F. Eisenbrand, M. Hartmann, and A. Schulz, *On the Chvatal rank of polytopes in the* 0/1 *cube*, Discrete Appl. Math., 98 (1999), pp. 21–27.

[5] M. L. Bonet, T. Pitassi, and R. Raz, *Lower bounds for cutting planes proofs with small coefficients*, J. Symbolic Logic, 62 (1997), pp. 708–728.

[6] M. L. Bonet, T. Pitassi, and R. Raz, *On interpolation and automatization for Frege systems*, SIAM J. Sci. Comput., 29 (2000), pp. 1939–1967.

[7] J. Buresh-Oppenheim, N. Galesi, S. Hoory, A. Magen, and T. Pitassi, *A strong direct product theorem for corruption and the multiparty communication complexity of disjointness*, Theory of Computing, 2 (2006), pp. 65–90.

[8] A. K. Chandra, M. L. Furst, and R. J. Lipton, *Multi-party protocols*, in Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing, Boston, MA, 1983, pp. 94–99.

[9] V. Chvátal, W. Cook, and M. Hartmann, *On cutting-plane proofs in combinatorial optimization*, Linear Algebra Appl., 114/115 (1989), pp. 455–499.

[10] V. Chvátal, *Edmonds polytopes and a hierarchy of combinatorial problems*, Discrete Math., 4 (1973), pp. 305–337.

[11] W. Cook, C. R. Coullard, and G. Turan, *On the complexity of cutting plane proofs*, Discrete Appl. Math., 18 (1987), pp. 25–38.

[12] S. Dash, *An exponential lower bound on the length of some classes of branch-and-cut proofs*, Math. Oper. Res., 30 (2005), pp. 678–700.

[13] R. Diestel, *Graph Theory*, Springer-Verlag, Berlin, 1997.

[14] F. Eisenbrand and A. S. Schulz, *Bounds on the Chvatal rank of polytopes in the* 0/1*-cube*, Combinatorica, 23 (2003), pp. 245–261.

[15] D. Grigoriev, E. A. Hirsch, and D. V. Pasechnik, *Complexity of semi-algebraic proofs*, in (STACS) 2002: 19th Annual Symposium on Theoretical Aspects of Computer Science, Antibes, France, 2002, Lecture Notes in Comput. Sci. 2285, Springer-Verlag, Berlin, pp. 419–430.

[16] R. Impagliazzo, T. Pitassi, and A. Urquhart, *Upper and lower bounds on tree-like cutting planes proofs*, in Proceedings of the 9th Annual IEEE Symposium on Logic in Computer Science, Paris, France, 1994, pp. 220–228.

[17] D. Itsykson and A. Kojevnikov, *Lower bounds of static Lovasz-Schrijver calculus proofs for Tseitin tautologies*, Zapiski Nauchny Seminarov POMI, 340 (2006), pp. 10–32 (in Russian).

[18] B. Kalyanasundaram and G. Schnitger, *The probabilistic communication complexity of set intersection*, SIAM J. Discrete Math., 5 (1992), pp. 545–557.

[19] L. G. Khachian, *A polynomial time algorithm for linear programming*, Dokl. Akad. Nauk SSSR, n.s., 244 (1979), pp. 1093–1096 (in Russian); Soviet Math. Dokl. 20 (1979), pp. 191–194 (in English).

[20] E. Kushilevitz and N. Nisan, *Communication Complexity*, Cambridge University Press, Cambridge, England, 1997.

[21] L. Lovász and A. Schrijver, *Cones of matrices and set-functions and* 0-1 *optimization*, SIAM J. Optim., 1 (1991), pp. 166–190.

[22] A. Lubotzky, R. Phillips, and P. Sarnak, *Ramanujan graphs*, Combinatorica, 8 (1988), pp. 261–277.

[23] N. Nisan, *The communication complexity of threshold gates*, in Combinatorics: Paul Erdös is Eighty, Volume I, V. Mikl'os and T. Szonyi, eds., Bolyai Society, 1993, pp. 301–315.

[24] P. Pudlák, *Lower bounds for resolution and cutting plane proofs and monotone computations*, J. Symbolic Logic, 62 (1997), pp. 981–998.

[25] P. Pudlák, *On the complexity of propositional calculus*, in Sets and Proofs, Invited Papers from Logic Colloquium '97, Cambridge, 1999, pp. 197–218.

[26] R. Raz and A. Wigderson, *Monotone circuits for matching require linear depth*, J. ACM, 39 (1992), pp. 736–744.

[27] H. D. Sherali and W. P. Adams, *A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems*, SIAM J. Discrete Math., 3 (1990), pp. 411–430.

[28] L. Valiant and V. Vazirani, *NP is as easy as detecting unique solutions*, Theoret. Comput. Sci., (1986), pp. 85–93.

SIAM J. COMPUT.
Vol. 37, No. 3, pp. 870–894

IMPROVED BOUNDS ON NONBLOCKING 3-STAGE CLOS NETWORKS*

JOSÉ R. CORREA† AND MICHEL X. GOEMANS‡

Abstract. We consider a generalization of edge coloring bipartite graphs in which every edge has a weight in $[0, 1]$ and the coloring of the edges must satisfy that the sum of the weights of the edges incident to a vertex v of any color must be at most 1. For unit weights, König's theorem says that the number of colors needed is exactly the maximum degree. For this generalization, we show that $2.557n + o(n)$ colors are sufficient, where n is the maximum total weight adjacent to any vertex, improving the previously best bound of $2.833n + O(1)$ due to Du et al. Our analysis is interesting on its own and involves a novel decomposition result for bipartite graphs and the introduction of an associated continuous one-dimensional bin packing instance which we can prove allows perfect packing. This question is motivated by the question of the rearrangeability of 3-stage Clos networks. In that context, the corresponding parameter n of interest in the edge coloring problem is the maximum over all vertices of the number of unit-sized bins needed to pack the weights of the incident edges. In that setting, we are able to improve the bound to $2.5480n + o(n)$, also improving a bound of $2.5625n + O(1)$ of Du et al. We also consider the online version of this problem in which edges have to be colored as soon as they are revealed. In this context, we can show that $5n$ colors are enough. This contrasts with the best known lower bound of $3n - 2$ by Tsai, Wang, and Hwang but improves upon the previous best upper bound of $5.75n$ obtained by Gao and Hwang. Additionally, we show several improved bounds for more restricted versions of the problem. These online bounds are achieved by simple and easy-to-implement algorithms, inspired by the first fit heuristic for bin packing.

Key words. bipartite edge coloring, rearrangeability of 3-stage Clos networks, bin packing

AMS subject classifications. 68W40, 68R10, 90C27, 90C59

DOI. 10.1137/060656413

1. Introduction.

1.1. Clos networks.

Suppose we need to connect a set of inlets $i_1, \dots, i_k$—which may represent telephone calls, parallel machines, or any kind of connection request—to a set of outlets $j_1, \dots, j_k$, through an *interconnection network*, in such a way that any request permutation (i.e., a permutation of $\{1, \dots, k\}$) can be routed simultaneously. More precisely, let us define a connection request as a pair, (i, j), where i is an inlet and j an outlet, and a *request frame* as any collection of requests such that every inlet and every outlet are associated with at most one request. The goal is to design a network that can route any request frame; such a network is called *nonblocking*.

Naturally, the simplest way to achieve this is to directly connect every inlet to every outlet by a different link. This solution, called *crossbar*, was already developed and implemented for telephone communications in the late 1930's by Western Electric (the Bell System). Despite the simplicity and nice properties of crossbar networks,

*Received by the editors April 5, 2006; accepted for publication (in revised form) February 16, 2007; published electronically August 15, 2007.
http://www.siam.org/journals/sicomp/37-3/65641.html

†School of Business, Universidad Adolfo Ibáñez, Avenida Presidente Errázuriz 3485, Santiago, Chile (correa@uai.cl). The research of this author was partially supported by grant FONDECYT 1060035.

‡Department of Mathematics, Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, MA 02139 (goemans@math.mit.edu). The research of this author was partially supported by NSF contract CCF-0515221 and ONR grant N00014-05-1-0148.

their main drawback is that they require too many links to achieve their goal: If we have k inlets and k outlets, they require k^2 links. In 1953, Clos [9] introduced a new type of interconnection network with the same property but that requires only $O(k^{3/2})$ links. These networks have been widely used for data communications and parallel computing systems (see, e.g., [3, 16]).

Formally, a 3-stage Clos network $C(n_1, r_1, m, n_2, r_2)$ is an interconnection network where the first stage consists of r_1 crossbars of size $n_1 \times m$, the last stage has r_2 crossbars of size $m \times n_2$, and the middle stage has m crossbars of size $r_1 \times r_2$. Moreover, each of the r_1 input switches is connected to each of the m middle switches. Similarly, the middle stage and the last stage are fully connected. We focus on the case in which $n_1 = n_2 = n$; i.e., the number of *inlets* or inputs of the input stage switches is equal to the number of *outlets* or outputs of the output stage switches. We also assume that $r_1 = r_2 = r$, even though all our results hold independently of what r_1 and r_2 are. The resulting Clos network is denoted by $C(n, m, r)$.

In a Clos network, a request frame is said to be *routable* if all requests can be routed through a middle switch so that no two requests share a link. The main question related to 3-stage Clos networks is to determine the number m of middle switches (crossbars) needed to route any request frame, i.e., for the network to be nonblocking. The answer, however, depends on the model we consider. Essentially there are three settings in which this question has been studied and used:

- *Rearrangeably nonblocking:* An interconnection network is *rearrangeably nonblocking* (or just rearrangeable) if every request frame is routable. This is the relevant question in an offline setting.
- *Strictly nonblocking:* An interconnection network is *strictly nonblocking* if any new connection request, compatible with a request frame, can be routed, independent of how the rest of the request frame is routed (i.e., independent of the state of the network). This is a relevant question in an online setting.
- *Wide-sense nonblocking:* If connection requests are revealed over time, an interconnection network is *wide-sense nonblocking* if any new connection request, compatible with a request frame, can be routed, provided that the rest of the request frame was routed according to a given routing algorithm. This question is the most important in practice, since it is motivated by the online environment, but it is less restrictive than the strictly nonblocking requirement. It is important to mention that several authors consider the more restrictive definition of wide-sense nonblocking in which the algorithm has to be able to route new connection requests even if previous connections terminate (see, e.g., Benes' original book [5]).

Clos himself noted that $C(n, 2n-1, r)$ is strictly nonblocking (which implies that it is wide-sense nonblocking as well), while, shortly after, Slepian [29] (see also [5]) proved that $C(n, n, r)$ is rearrangeable. Moreover, both results are best possible. It is then clear that if the total number of inlets is k and we choose $n = r = \sqrt{k}$, the number of links required in a 3-stage Clos network is $5 \times k^{3/2}$ or $8 \times k^{3/2}$, depending on whether we need it to be rearrangeably or strictly nonblocking.

Although our focus here will be the study of 3-stage Clos networks, let us briefly mention a few results for general interconnection networks. Shannon [27] showed that $\Omega(k \log k)$ links are needed for an interconnection network to even be rearrangeable. Surprisingly, this lower bound was matched by Benes [6] and Beizer [4], who designed rearrangeable networks of size $O(k \log k)$. Later, Bassalygo and Pinsker [2] constructively showed the existence of strictly nonblocking networks of size $O(k \log k)$.

The multirate environment. We just described the *classic* switching environment, in which connection requests fully use a link and have all the same bandwidth. However, in modern communications, different requests may have different bandwidths and may be combined in a given link if the "link capacity" is large enough to carry both requests. This setting is usually called the *multirate* environment. In such a setting, a connection request is a triple (i, j, w), where i is an inlet, j an outlet, and w the weight (thus, the classic environment corresponds to the special case in which all weights are 1). A *request frame* is a collection of requests such that the total weight of all requests in the frame involving a fixed inlet or outlet does not exceed 1. In a Clos network, all $r \times m$ links between the input switches and middle switches and all $m \times r$ links between the middle switches and the output switches also have capacity 1. A request frame is said to be *routable* if all requests can be routed through a middle switch so that none of the link capacities is violated. For a recent survey on multirate Clos networks, we refer the reader to the excellent survey by Turner and Melen [30], who also initiated the research on multirate switching networks [23]. As in the classic environment, the question is to determine the minimum value of m of middle switches such that any request frame can be routed; the network is then said to be multirate nonblocking. Again, the answer depends on whether the problem is considered online or offline. However, the questions are still wide open and need further investigations.

- *Rearrangeably nonblocking (offline):* An interconnection network is said to be *multirate rearrangeably nonblocking* (or just rearrangeable) if every request frame is routable. The question is thus to determine the minimum value of m of middle switches such that $C(n, m, r)$ is multirate rearrangeable, and this minimum value is denoted by $m(n, r)$. It is particularly interesting to obtain bounds that are independent of r.
- *Wide-sense nonblocking (online):* If connection requests are revealed over time (both the inlet-outlet pair and its weight), an interconnection network is said to be *wide-sense nonblocking* if any new connection request, compatible with a request frame, can be routed, provided that all the rest of the request frame was routed according to a given routing algorithm. Thus, the question is again to determine the minimum value m of middle switches such that $C(n, m, r)$ is wide-sense nonblocking, and this value is denoted by $m_W(n, r)$. Let us emphasize again that we assume that the requests never terminate, i.e., that we have no deletions during the execution; this is the same weaker setting as in [13], for example.
- *Strictly nonblocking:* In the multirate environment, we say that an interconnection network is *strictly nonblocking* if any new connection request, compatible with a request frame, can be routed, independent of how the rest of the request frame is routed (i.e., independent of the state of the network).

1.2. Problem definition.

The question of rearrangeability and nonblocking properties of a 3-stage Clos network can be translated in graph-theoretic terms in the following way. We are given a bipartite (multi)graph $G = (V, E)$ with bipartition A, B (say with $|A| = |B| = r$); in what follows, all our graphs will be multigraphs. A and B represent the input and output switches, respectively. Edge $e = (i, j)$ represents a request between input switch i and output switch j and carries a weight $0 \leq w(e) \leq 1$. The assumption of the requests being a request frame can be translated into the assumption that the weights on the edges incident to $v \in V$ can be packed into n unit-sized bins. That is, for all $v \in V$, the set $\delta(v)$ of edges incident to v can be partitioned into n groups C_i^v, $i = 1, \dots, n$, satisfying

$$\sum_{e \in C_i^v} w(e) \leq 1 \qquad \text{for all } i = 1 \dots, n. \tag{1}$$

Following the notation in [24], let $\mathcal{B}_r^n$ be the collection of such edge-weighted bipartite multigraphs.

A Clos network $C(n, m, r)$ is then (multirate) rearrangeable if, for every graph in $\mathcal{B}_r^n$, the edges can be colored with m colors so that the total weight of all edges of the same color incident to a vertex v is at most 1. The question is thus to determine the minimum number $m(n, r)$ of colors needed to properly color every weighted bipartite graph in $\mathcal{B}_r^n$. In the online setting, we know only a priori that the graph belongs to $\mathcal{B}_r^n$, but the edges and their weight are revealed over time. Similar to the rearrangeable case, a Clos network $C(n, m, r)$ is (multirate) wide-sense nonblocking if there exists an online algorithm A such that, for every graph in $\mathcal{B}_r^n$, the edges can be colored with m colors so that the total weight of all edges of the same color incident to a vertex v is at most 1. The question is thus to determine the minimum number $m_W(n, r)$ for which there is an online algorithm that properly colors every weighted bipartite graph in $\mathcal{B}_r^n$ using no more than $m_W(n, r)$ colors. In the same manner, $C(n, m, r)$ is (multirate) strictly nonblocking if, for any $G = (V, E) \in \mathcal{B}_r^n$ and any proper m-coloring of $(V, E \setminus \{e\})$, for any $e \in E$, edge e can be colored without changing the color of any already colored edge and using any extra color. Now $m_S(n, r)$ is the minimum number of colors such that $C(n, m_S(n, r), r)$ is strictly nonblocking.

If all weights are forced to belong to a subset $I \subset [0, 1]$, let $\mathcal{B}_r^n(I)$ denote the natural extension of $\mathcal{B}_r^n$. In this case, $m_I(n, r)$ is the smallest integer such that every graph in $\mathcal{B}_r^n(I)$ admits a proper coloring with $m_I(n, r)$ colors. The quantities $m_{W\,I}(n, r)$ and $m_{S\,I}(n, r)$ are the natural counterparts of $m_I(n, r)$ in the wide-sense and strictly nonblocking setting.

Another special case that has attracted attention is when all edge weights can take only k different values (known, in Clos network terminology, as the bounded rate environment, or k-rate environment). We denote by $m^k(n, r)$ the minimum number of middle switches so that $C(m^k(n, r), n, r)$ is multirate rearrangeable when all request frames have weights with only k different values. Similarly, $m_W^k(n, r)$ is the corresponding counterpart of $m^k(n, r)$.

In section 3, we focus on a generalized bipartite edge-coloring problem, very similar to the one just described, except that we require only the weights incident to any vertex to add up to at most n. That is, condition (1) is replaced by the following weaker condition:

$$\sum_{e \in \delta(v)} w(e) \leq n \qquad \text{for all } v \in V. \tag{2}$$

Here $\mathcal{D}_r^n$ denotes the natural counterpart of $\mathcal{B}_r^n$. As $\mathcal{B}_r^n \subseteq \mathcal{D}_r^n$, the required number of colors in this case, denoted by $M(n, r)$, is clearly greater than or equal to $m(n, r)$. If all weights are forced to belong to a subset $I \subset [0, 1]$, $\mathcal{D}_r^n(I)$ and $M_I(n, r)$ denote the natural counterparts of $\mathcal{B}_r^n(\mathrm{I})$ and $m_I(n, r)$.

1.3. Discussion of previous work.

Let us review some existing results on this problem. We start by giving the most relevant results on rearrangeability, and later we focus on wide-sense and strictly nonblocking properties.

Rearrangeability. The first important result was proved shortly after the introduction of 3-stage Clos networks and is due to Slepian [29] (see also [5]). He used König's edge-coloring theorem [19] (see also [11]) to prove that $m_{[1,1]}(n,r) = n$. Melen and Turner [23] initiated the research on multirate switching networks and proved that $m_{[0,1/2]}(n,r) \leq M_{[0,1/2]}(n,r) \leq 2n-1$. More generally, they proved that

$$m_{[0,B]}(n,r) \leq M_{[0,B]}(n,r) \leq \frac{n}{1-B}.$$

On the other hand, it is easy to prove that $m_{[b,1]}(n,r) \leq n\lfloor \frac{1}{b} \rfloor$ and that $M_{[b,1]}(n,r) \leq \frac{n}{b}$.

Previous to this work, the best bounds known on $m(n,r)$ in the general setting are $\frac{5n}{4} \leq m(n,r) \leq \frac{41n}{16} + O(1)$ and were obtained by Ngo and Vu [24] (lower bound) and Du et al. [12] (upper bound). The latter authors also obtained the previously best bounds for $M(n,r)$, namely $2n-1 \leq M(n,r) \leq \frac{17n}{6} + O(1)$.

In the k-rate environment, better bounds have been proved. For $k=2$, one can actually verify the Chung–Ross conjecture, namely, that the $2n-1$ bound holds in this case [8]. Moreover, Lin et al. [21] proved that

$$m^3(n,r) \leq \frac{9n}{4} + O(1) \quad \text{ and } \quad m^3_{(\frac{1}{5},1]}(n,r) \leq 2n.$$

The first bound is an improvement over the $\frac{7n}{3}$ bound obtained by Lin et al. [20]. Unfortunately, the proofs of all bounds for the finite rate environment rely on rather tedious case analysis.

Wide-sense and strictly nonblocking. Let us now survey some of the most relevant results concerning nonblocking properties of 3-stage Clos networks. In the classical environment, Clos [9] proved that $C(n, 2n-1, r)$ is strictly nonblocking. Unfortunately, as first noted in [23], in the multirate environment, $C(n,m,r)$ cannot be strictly nonblocking unless m is infinity. Indeed, consider the network $C(n,m,1)$ and assume that there is a connection request of weight 1 and $(n-1)/\varepsilon$ connection requests of weight $\varepsilon = (n-1)/m$ between the only input and output switch pair in the network. A possible current state for the network is that each small connection request is routed along a different middle switch, and thus the large request cannot be routed, implying that the network is in a blocking state. However, if connection requests are restricted to have weights within some interval, finite bounds can be obtained. Indeed, Melen and Turner [23] proved that $m_{S[b,1]}(n,r) \leq 2\lfloor (n-1)/b \rfloor + 3$, which was further improved by Chung and Ross [8] to $m_{S[b,1]}(n,r) \leq 2\lfloor 1/b \rfloor (n-1) + 1$. The latter authors also proved that $m_{S(0,B]} \leq 2\lceil \frac{n-B}{1-B} \rceil + 1$.

The bad example above motivated the algorithmic concept of wide-sense nonblocking. Indeed, already in [23] it was noted that $8n$ middle switches are enough to ensure the wide-sense nonblocking condition, i.e., $m_W(n,r) \leq 8n$. Later, Chung and Ross [8] used their bounds on $m_{S[b,1]}(n,r)$ and $m_{S(0,B]}(n,r)$ to improve the bound. Indeed, their algorithm would split connection requests according to their weight: the smaller than or equal to 1/2 and those strictly larger than 1/2. The bound is therefore

$$m_W(n,r) \leq m_{S(1/2,1]}(n,r) + m_{S(0,1/2]}(n,r) \leq 2n-2+1+4n+1 = 6n.$$

The best known bound previous to our result was obtained by Gao and Hwang [13]. They used a *quota scheme*, which consists of reserving some middle switches for large connections while letting the rest carry any connection request. This approach led them to the bound $m_{W[0,1/2]} \leq 3.75n$, implying, in the same manner as above, that

$$m_W(n,r) \leq 5.75n.$$

The study of lower bounds for wide-sense nonblocking properties has been much more recent. Bar-Noy, Motwani, and Naor [1] were the first to prove that in the classical setting $m_{W[1,1]}(n,r) \geq 2n-1$ for exponentially large r. This surprising result essentially says that in the classical single-rate environment the strictly nonblocking and wide-sense nonblocking conditions are the same. Moreover, recent work by Haxell et al. [14] shows that this lower bound holds even for $r = \Omega(n^2)$. In the multirate environment, there is only a recent improvement on the previous bound. Tsai, Wang, and Hwang [31] proved that $m_W(n,r) \geq 3n-2$, and their proof also works in the more restricted 2-rate environment.

1.4. Overview and main results in the paper. The main goal of this paper is to present bounds on $M(n,r)$, $m(n,r)$, and $m_W(n,r)$. Indeed, we will show that $2.557n$, $2.548n$, and $5n$ are, respectively, upper bounds on these numbers.

We start in section 2 by showing a result on balanced decomposition of bipartite graphs into matchings. In the context of Clos networks, this result becomes useful only in section 3; however, we believe it is interesting on its own, and so we have decided to present it in a separate section. The question that is addressed is as follows: Given a bipartite graph G and nonnegative numbers $\gamma_1, \ldots, \gamma_l$ summing to 1, decompose the graph into $F_1, \ldots, F_l$ such that the degree of any vertex v in F_i is approximately γ_i times the degree of v in G. We show that the decomposition can be done such that for all i and all vertices, the degree of v in F_i differs from its required value by an additive constant less than 3. The question whether this constant can be decreased to 1 is to the best of our knowledge open.

Our main contribution in this paper, proved in section 3, is the following result.

THEOREM 1. *The number of colors required to properly color every weighted bipartite graph in $\mathcal{D}_r^n$ is at most $2.557n + o(n)$. In other words,*

$$M(n,r) \leq 2.557n + o(n).$$

Observe that this does not improve only upon Du et al.'s bound of $\frac{17}{6}n + O(1)$ on $M(n,r)$ but even slightly upon their bound of $\frac{41}{16}n+O(1) = 2.5625n+O(1)$ on $m(n,r)$. In fact, our approach can also be applied to bounding $m(n,r)$ directly, and this gives us a slightly improved bound of $m(n,r) \leq 2.5480n + o(n)$. The latter improvement is sketched in section 3.7.

For most of section 3, we consider the generalized bipartite edge coloring problem in which the weights on edges incident to any vertex sum to at most n, i.e., graphs in $\mathcal{D}_r^n$. The approach we consider to attack this problem associates a bin packing instance with every such generalized edge coloring instance. For this purpose, we first decompose the edge weighted bipartite graph $G = (V,E)$ into a union of matchings. We then create a bin packing instance in which all bins have size 1. We create an item of our bin packing instance for each matching in our decomposition, and we set its size to be the maximum weight of any edge in the matching. A packing with k bins immediately leads to a valid k-coloring by simply coloring the edges of all matchings (items) placed in the same bin with the same unique color. As we shall see in section 3.1, this approach needs that we first discard all edges whose weight is less than some parameter α (to be determined). This can be done using the following result implicit in Du et al. [12].

LEMMA 2. *Consider $G = (V,E) \in \mathcal{D}_r^n$ with bipartition $V = A \cup B$ and assume that we have used at least $\frac{2n}{1-\alpha}$ colors to color all edges except some edges e with $w(e) \leq \alpha$. Then we can greedily color these remaining edges without using any additional color. In particular, if $M_{(\alpha,1]}(n,r) \leq \lceil 2n/(1-\alpha) \rceil$, then $M(n,r) \leq \lceil 2n/(1-\alpha) \rceil$.*

Proof. If $e = (u, v) \in E$ with $w(e) \leq \alpha$ cannot be colored then the total weight of edges of a given color i incident to either u or v is greater than $1 - \alpha$. Summing over all $\frac{2n}{1-\alpha}$ colors, we get a contradiction with sum of conditions (2) for u and v. $\square$

Therefore, we can focus on instances in which all weights are in $[\alpha, 1]$, provided that we are willing to use $\lceil \frac{2n}{1-\alpha} \rceil$ colors. As the proof of this last result used only any greedy algorithm, the result also holds in the wide-sense nonblocking setting.

Our main contribution is to show that, for *any* generalized edge coloring problem with weights in $[\alpha, 1]$, we can decompose the bipartite graph into matchings in such a way that the corresponding bin packing instance can be packed into at most $n + o(n)$ bins plus the number of bins required to pack a continuous bin packing instance with density $\frac{n}{x^2}$ for $x \in [\alpha, 1]$ (i.e., the number of items with size in the interval $(x, x + dx)$ is $\frac{n}{x^2} dx$). We should emphasize that our bin packing instance is independent of the given bipartite graph G; it is based only on the fact that $G \in \mathcal{D}_r^n$. Although it is easier to refer in the statements here to the continuous bin packing instance, we actually deal only with an arbitrarily fine discretization of it and consider discrete bin packing instances. Our decomposition of the graph into matchings relies on the result of section 2 and is described in section 3.2, while the construction of our bin packing instance is detailed in section 3.3.

Once the continuous bin packing instance with density $\frac{n}{x^2}$ for $x \in [\alpha, 1]$ is constructed, in sections 3.4, 3.5, and 3.6, we turn to compute the number of bins it requires. First, we observe that all items of size greater than $1 - \alpha$ need to be placed alone in bins; they therefore require $\int_{1-\alpha}^{1} \frac{n}{x^2} dx = \frac{\alpha}{1-\alpha} n$ bins. For the remaining items with density $\frac{n}{x^2}$ for $x \in [\alpha, 1 - \alpha]$, we prove that they can be *perfectly* packed. This means that the number of bins they require is simply their total size, up to lower-order terms (accounting for the discretization). This means that they require $\int_{\alpha}^{1-\alpha} x \frac{n}{x^2} dx = n \ln \frac{1-\alpha}{\alpha}$ additional bins. This relies on a result of Rhee and Talagrand [26]. The total number of bins used is thus $(1 + \frac{\alpha}{1-\alpha} + \ln \frac{1-\alpha}{\alpha})n$, and we choose α so that this equals $\frac{2}{1-\alpha} n$ in order to be able to greedily color the edges with weight lower than α. For $\alpha = 0.217811\ldots$, we obtain that the number of colors needed is less than $2.557n$.

It is worth mentioning that our main result can be done algorithmically. Indeed, the continuous bin packing instance is independent of the input; therefore, a discretization of it can be solved optimally a priori by exhaustive search (or by using any good algorithm for bin packing). The matching decomposition, for edges with weight in $[\alpha, 1]$, can be efficiently done using network flows techniques (see Lemma 5). Finally, the edges with weight in $(0, \alpha)$ can be greedily colored as in Lemma 2.

In section 4, we will use simple adaptations of the first fit (FF) heuristic for the classical bin packing problem to obtain improved bounds on the wide-sense nonblocking properties of 3-stage Clos networks. In the bin packing setting, FF places a new item in the first bin that has space available for it; in our online setting, it will simply color an edge with the smallest possible color (under some arbitrary order on the colors) as long as it does not violate condition (1). Our main result, proved in section 4.1, is to show that $C(n, 5n, r)$ is wide-sense nonblocking, i.e., $m_W(n, r) \leq 5n$. Later, in section 4.2, we show that $m_{W(0,1/2]} \leq 3.601n + 3$. Both bounds improve upon the bounds obtained by Gao and Hwang [13] of $5.75n$ and $3.75n$, respectively.

Additionally, in section 4.3 we are able to show that in the 2-rate environment, there is an online algorithm that uses no more than $3n$ middle switches to schedule any request frame. This not only improves the previous best known bound of $4n$ [13] but also almost matches the lower bound on $m_W(n, r)$ of $3n - 2$ obtained by

Tsai, Wang, and Hwang [31] which is also valid in the 2-rate case. We can therefore conclude that

$$3n-2 \leq m_W^2(n,r) \leq 3n.$$

Finally, in section 5 we prove that using an analogue of the FF decreasing heuristic for bin packing, no more than $\frac{8n}{3}$ middle switches are needed to route any request frame. As sorting is needed, this bound holds only in the offline setting, and it does not improve upon the bound of $m(n,r) \leq 2.548n + o(n)$ given in section 3. However, it has the following advantages: (i) it is a nonasymptotic result; (ii) it is a very simple to implement algorithm; and (iii) it can be implemented to run in time $O(n \log n)$.

2. Balanced decompositions of bipartite graphs. Given a subset of edges F of a graph G and a vertex v, we let $\deg_F(v)$ denote the degree of vertex v in F, that is, $|\delta(v) \cap F|$, where $\delta(v)$ is the set of edges incident to v in the graph. The following result follows easily from network flow theory.

LEMMA 3 (Hoffman [15]). *Consider a bipartite graph $G = (V, E)$ and let $0 \leq \mu_1, \mu_2$ with $\mu_1 + \mu_2 = 1$. Then there exists a partition of E into E_1 and E_2 such that*

$$\lfloor \mu_i \deg_E(v) \rfloor \leq \deg_{E_i}(v) \leq \lceil \mu_i \deg_E(v) \rceil$$

for $i = 1, 2$ and all $v \in V$.

Proof. Let A, B be the bipartition of the bipartite graph G. Orient all edges from A to B. Add a source with arcs to all vertices in A and a sink with arcs from all vertices in B. Set the capacity of all the arcs in E to be 1, and set upper and lower capacities on the arcs adjacent to the source and sink to be $\lceil \mu_1 \deg_E(v) \rceil$ and $\lfloor \mu_1 \deg_E(v) \rfloor$, where v is the corresponding adjacent vertex. As a feasible flow can be obtained by setting the flow on every arc in E to be μ_1, there exists an integer feasible flow, and this flow corresponds to the edge set E_1. The remaining edges E_2 also satisfy the required property. □

The next theorem is an extension of Hoffman's result.

THEOREM 4. *Consider a bipartite graph $G = (V, E)$ and let $\gamma_1, \ldots, \gamma_l \in (0, 1)$ such that $\sum_{i=1}^{l} \gamma_i = 1$. Then there exists a partition $E_1, \ldots, E_l$ of E such that for all $v \in V$ and all $i = 1, \ldots, l$,*

$$\gamma_i \deg_E(v) - e_i(v) < \deg_{E_i}(v) < \gamma_i \deg_E(v) + e_i(v).$$

Here $e_i(v) < 3$, and $\sum_{i=1}^{l} e_i(v) \leq 2(l-1)$.

Proof. Let $L = \{1, \ldots, l\}$. We construct a binary tree T with $l-1$ internal nodes and l leaves, each node being labelled by a subset of L. The root is labelled with L, and the l leaves are labelled by a distinct singleton subset of L. If an internal node is labelled with N, then its two children are labelled with I and $N \setminus I$, where $I, N \setminus I$ is the most balanced number partition of N; i.e., I is such that $\max\{\gamma(I), \gamma(N \setminus I)\}$ is minimized (for a set S, $\gamma(S)$ denotes $\sum_{i \in S} \gamma_i$).

With every node with label I, we also associate an edge set $E(I)$. We first set $E(L) = E$. Given $E(N)$ for an internal node N, we obtain $E(I)$ and $E(N \setminus I)$ for its children by applying Lemma 3 to the graph with edge set $E(N)$ and with $\mu_1 = \gamma(I)/\gamma(N)$ and $\mu_2 = 1 - \mu_1$. The leaves are thus associated with subgraphs $E(\{i\})$ which make a partition of E. We claim that $E(\{i\})$ satisfies the required properties for E_i.

Fix a vertex $v \in V$ (for simplicity, we just drop v when writing $\deg_*(v)$) and an index $i \in L$. Let $\{i\} = A_0 \subset A_1 \subset \cdots \subset A_k = L$ be the labels on the path from the leaf $\{i\}$ to the root. We now derive an upper bound on $\deg_{E_i}(v)$ (and we could proceed similarly for the lower bound). From Lemma 3, we have that

$$\begin{aligned}\deg_{E_i}(v) = \deg_{E(A_0)} &< \frac{\gamma(A_0)}{\gamma(A_1)} \deg_{E(A_1)} +1 \\ &< \frac{\gamma(A_0)}{\gamma(A_1)} \left(\frac{\gamma(A_1)}{\gamma(A_2)} \deg_{E(A_2)} +1 \right) + 1 \\ &< \frac{\gamma(A_0)}{\gamma(A_1)} \left(\frac{\gamma(A_1)}{\gamma(A_2)} \left(\cdots \left(\frac{\gamma(A_{k-1})}{\gamma(A_k)} \deg_E +1 \right) \cdots \right) + 1 \right) + 1 \\ &= \frac{\gamma(A_0)}{\gamma(A_k)} \deg_E +1 + \frac{\gamma(A_0)}{\gamma(A_1)} + \frac{\gamma(A_0)}{\gamma(A_2)} + \cdots + \frac{\gamma(A_0)}{\gamma(A_{k-1})} \\ &= \gamma_i \deg_E + e_i(v),\end{aligned}$$

where $e_i(v) = 1 + \frac{\gamma(A_0)}{\gamma(A_1)} + \frac{\gamma(A_0)}{\gamma(A_2)} + \cdots + \frac{\gamma(A_0)}{\gamma(A_{k-1})}$. Let $\eta = \min_{i \in A_1} \gamma_i$ and let j be the arg min. Let $a = \gamma(A_0) \geq \eta$. Thus we have $\gamma(A_1) \geq a + \eta$. In general, when considering A_k, we split it into A_{k-1} and $A_k \setminus A_{k-1}$, while we could have split it into $A_{k-1} \setminus \{j\}$ and the rest. This implies that $\gamma(A_{k-1}) - \eta \leq \gamma(A_k) - \gamma(A_{k-1})$, i.e., $\gamma(A_k) \geq 2\gamma(A_{k-1}) - \eta$. Using this repeatedly, we get $\gamma(A_2) \geq 2a + \eta$, $\gamma(A_3) \geq 4a + \eta$, and, generally, $\gamma(A_{k-1}) \geq 2^k a + \eta$. Thus the bound becomes

$$\begin{aligned}e_i(v) &\leq 1 + \frac{a}{a+\eta} + \frac{a}{2a+\eta} + \frac{a}{4a+\eta} + \frac{a}{8a+\eta} + \cdots \\ &\leq 1 + \frac{a}{a} + \frac{a}{2a} + \frac{a}{4a} + \frac{a}{8a} + \cdots < 3.\end{aligned}$$

Finally, in order to get a bound on $\sum_i e_i(v)$, observe that

$$\sum_{i=1}^{l} e_i(v) = \sum_{(\text{all labels } N \text{ except the root})} \sum_{i \in N} \frac{\gamma_i}{\gamma(N)} = 2(l-1),$$

since there are $2l-1$ nodes in the binary tree. A proof of the lower bound on $\deg_{E_i}(v)$ is identical. □

We suspect that the bound can be further improved. If $\gamma_i = 1/l$ for every i, de Werra [10] has shown that we can impose $\lfloor \gamma_i \deg_E(v) \rfloor \leq \deg_{E_i}(v) \leq \lceil \gamma_i \deg_E(v) \rceil$ for every i, while Theorem 4 implies $\lfloor \gamma_i \deg_E(v) \rfloor - 2 \leq \deg_{E_i}(v) \leq \lceil \gamma_i \deg_E(v) \rceil + 2$ for every i (without making assumptions on the γ_i's). We do not know whether the tighter condition (without the $+2$) can be imposed in the general case. The proof technique used here, however, cannot even improve the $+2$ term into a $+1$ term. Indeed, for $\gamma_i = 1/13$ for $i = 1, \ldots, 13$, one can see that no partitioning scheme would give a bound on $e_i(v)$ (using the analysis in the proof of Theorem 4) better than $1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{5} + \frac{1}{13} = 2 + \frac{7}{260}$ (and this can be shown to be the worst when all γ_i's are equal).

As stated, the proof of Theorem 4 is not algorithmic, since we need to solve number partition as a subroutine. However, we used in the proof only the fact that the partitioning of N used is locally optimum in the sense that no item can be moved to the other side of the partition while making it more balanced. A locally optimum number partition can be obtained in polynomial time in several ways. Brucker, Hurink, and Werne [7] show (in the context of scheduling parallel machines) that iteratively improving the partition until a local optimum is reached takes $O(|N|^2)$ iterations. Schuurman and Vredeveld [28] noted that iteratively finding the best local improvement requires $O(|N|)$ iterations, which implies an overall running time of $O(|N| \log |N|)$. One can also use the differencing method of Karmarkar and Karp [18]. This differencing method, which also runs in $O(|N| \log |N|)$ time, consists of repeatedly replacing the largest two items by one new item whose size (i.e., γ value) equals the difference in sizes of these largest two items until only one item of size say Δ remains. By inverting the process, one can easily obtain a partition $(I, N \setminus I)$ with $\gamma(I) = \gamma(N \setminus I) + \Delta$. A simple inductive argument shows that all items in I have $\gamma_i \geq \Delta$, and therefore the partition obtained is locally optimum. Using any of these algorithms to find a local optimum, a partition of the edge set satisfying the conditions of Theorem 4 can be obtained in polynomial time.

3. Rearrangeably nonblocking Clos networks.

In this section, we consider the generalized bipartite edge coloring problem in which the weights on edges incident to any vertex sum to at most n, i.e., graphs in $\mathcal{D}_r^n$. As described earlier, we associate a bin packing instance with every such generalized edge coloring instance by decomposing the edge weighted bipartite graph $G = (V, E)$ into a union of matchings. We then create a bin packing instance in which all bins have size 1. We create an item of our bin packing instance for each matching in our decomposition, and we set its size to be the maximum weight of any edge in the matching. A packing with k bins immediately leads to a valid k-coloring by simply coloring the edges of all matchings (items) placed in the same bin with the same unique color.

3.1. Limitations.

Consider the following trivial instance of our generalized edge coloring problem. Let X be a finite subset of $(0, 1]$ and create a vertex in A and in B for each element $x \in X$ and $\lfloor \frac{n}{x} \rfloor$ edges between them. In this case, $2n - 1$ colors are sufficient (and needed if $\frac{1}{2} + \epsilon \in X$ for some small ϵ). No matter what decomposition into matchings we consider, our bin packing instance has at least $\lfloor \frac{n}{x} \rfloor$ items (matchings) of size at least x for every $x \in X$. If $X = \{x_0, x_1, \ldots, x_l\}$ with $x_0 > x_1 > \cdots > x_l$, this bin packing instance requires no fewer bins than another bin packing instance with $\lfloor \frac{n}{x_i} \rfloor - \lfloor \frac{n}{x_{i-1}} \rfloor$ items of size x_i for every $i \geq 1$ and $\lfloor \frac{n}{x_i} \rfloor$ items of size x_0. As X gets denser in $(0, 1]$, this bin packing instance tends to a continuous bin packing instance with density $\frac{n}{x^2}$ (i.e., the number of items of size in $(x, x + dx)$ is $\frac{n}{x^2} dx$) after having removed the n items of size 1. Now the number of bins required is at least the total size of all items $n + \int_0^1 x \frac{n}{x^2} dx$, which is unbounded!

To overcome this problem, we first discard all edges whose weight is less than some parameter α (to be determined) by using Lemma 2. This said, we can turn to proving the graph partitioning result in which our work is based.

3.2. Partitioning the graph.

From now on we fix a parameter $0 < \alpha < 1$ and work with graphs in $\mathcal{D}_r^n(\alpha, 1)$. The decomposition we need to construct our bin packing instance is given below.

LEMMA 5. *Consider the sequence $\alpha_0 = 1 > \alpha_1 > \alpha_2 > \cdots > \alpha_p = \alpha \geq 0$. Let $G = (V, E) \in \mathcal{D}_r^n(\alpha, 1)$. Then there exist sets $F_1, \ldots, F_p$ partitioning E such that the following hold:*

(i) $\max_{e \in F_k} w(e) \leq \alpha_{k-1}$.

(ii) *For all vertices* $v \in V$,

$$\deg_{F_k}(v) \le \left(\frac{1}{\alpha_k} - \frac{1}{\alpha_{k-1}}\right) n + a_k(v) \quad \text{for all } 2 \le k \le p,$$

$$\deg_{F_1}(v) \le \frac{n}{\alpha_1} + a_1(v),$$

where $a_k(v) \le 3(p-k+1)$.

Proof. Consider an instance $G = (L, R, E)$ with weight function w and let

$$D_i = \{e \in E \ : \ w(e) \in (\alpha_i, \alpha_{i-1}]\}$$

for $i = 1, \dots, p$. From inequality (2) we can easily deduce that for all $v \in L \cup R$,

$$\sum_{i=k}^{p} \alpha_i \deg_{D_i}(v) \le n \qquad \text{for all } k = 1 \dots, p. \tag{3}$$

If we divide the inequality (3) corresponding to $k = 1$ by α_1 and multiply the kth inequality (3) by $(\frac{1}{\alpha_k} - \frac{1}{\alpha_{k-1}})$, we obtain the following set of inequalities:

$$\left(\frac{1}{\alpha_1}\right)\alpha_1 \deg_{D_1}(v) + \left(\frac{1}{\alpha_1}\right)\alpha_2 \deg_{D_2}(v) + \cdots + \left(\frac{1}{\alpha_1}\right)\alpha_p \deg_{D_p}(v) \le \left(\frac{1}{\alpha_1}\right) n$$

$$\left(\frac{1}{\alpha_2} - \frac{1}{\alpha_1}\right)\alpha_2 \deg_{D_2}(v) + \left(\frac{1}{\alpha_2} - \frac{1}{\alpha_1}\right)\alpha_3 \deg_{D_3}(v) + \cdots + \left(\frac{1}{\alpha_2} - \frac{1}{\alpha_1}\right)\alpha_p \deg_{D_p}(v)$$
$$\le \left(\frac{1}{\alpha_2} - \frac{1}{\alpha_1}\right) n$$

$$\vdots$$

$$\left(\frac{1}{\alpha_p} - \frac{1}{\alpha_{p-1}}\right)\alpha_p \deg_{D_p}(v) \le \left(\frac{1}{\alpha_p} - \frac{1}{\alpha_{p-1}}\right) n.$$

Note that, for all $i = 1, \dots, p$, the coefficients in front of $\deg_{D_i}(v)$ over the above inequalities sum to 1. Therefore, for each D_i we can apply Theorem 4 with

$$\gamma_1^i = \frac{1}{\alpha_1}\alpha_i, \gamma_2^i = \left(\frac{1}{\alpha_2} - \frac{1}{\alpha_1}\right)\alpha_i, \dots, \gamma_i^i = \left(\frac{1}{\alpha_i} - \frac{1}{\alpha_{i-1}}\right)\alpha_i$$

to partition D_i into sets $D_i^1, \dots, D_i^i$ such that for all $k = 1, \dots, i$ and all $v \in V$,

$$\gamma_k^i \deg_{D_i}(v) - e_k^i(v) < \deg_{D_i^k}(v) < \gamma_k^i \deg_{D_i}(v) + e_k^i(v),$$

where $e_k^i(v) \le 3$ and $\sum_{k=1}^{i} e_k^i(v) \le 2(i-1)$.

We are now ready to finish the proof. Define $F_k = D_k^k \cup D_{k+1}^k \cup \cdots \cup D_p^k$ for all

$k = 1, \dots, p$. Thus letting $a_k(v) = \sum_{i=k}^{p} e_k^i(v) \le 3(p-k+1)$, we have the following:

$$\begin{aligned}\deg_{F_1}(v) &\le \sum_{i=1}^{p} \left(\gamma_1^i \deg_{D_i}(v) + e_1^i(v)\right)\\ &\le \left(\frac{1}{\alpha_1}\right) n + a_1(v);\\ \deg_{F_k}(v) &\le \sum_{i=k}^{p} \left(\gamma_k^i \deg_{D_i}(v) + e_k^i(v)\right)\\ &\le \left(\frac{1}{\alpha_k} - \frac{1}{\alpha_{k-1}}\right) n + a_k(v), \qquad 2 \le k \le p. \qquad \square\end{aligned}$$

3.3. The associated bin packing problem. Let us now consider a bipartite graph $G = (V, E) \in \mathcal{D}_r^n(\alpha, 1)$ and $F_1, \dots, F_p$ as in Lemma 5. By König's theorem, F_k can be decomposed into no more than

$$\left(\frac{1}{\alpha_k} - \frac{1}{\alpha_{k-1}}\right) n + \max_{v \in V} a_k(v)$$

matchings, for all $k = 2, \dots, p$ and F_1 can be decomposed into $\frac{n}{\alpha_1} + \max_{v \in V} a_1(v)$ matchings. We now construct an instance of the one-dimensional bin packing problem with unit-sized bins. Arbitrarily select $(\frac{1}{\alpha_k} - \frac{1}{\alpha_{k-1}})n$ matchings (or, more formally, the floor of this quantity) in the decomposition of F_k for $k = 2, \dots, p$ and assign each of them an item of size α_{k-1}. Similarly, arbitrarily select $\frac{n}{\alpha_1}$ matchings in the decomposition of F_1 and assign each of them an item of size α_0. Let $\mathcal{M}$ be those matchings selected in $F_1, \dots, F_p$, and, by construction, we have an item for each element of $\mathcal{M}$. Our bin packing instance is thus the following:

Input: $\frac{n}{\alpha_1}$ items of size 1 and $(\frac{1}{\alpha_k} - \frac{1}{\alpha_{k-1}})n$ items of size α_{k-1} for $k = 2, \dots, p$.

Output: A packing of the items into the minimum number of bins.

Observe that this bin packing instance is *independent* of $G = (V, E) \in \mathcal{D}_r^n(\alpha, 1)$ and depends only on n and the values of α_i selected.

Given any solution to this bin packing instance, say with k opened bins, we can easily obtain a coloring of all the edges in the union of the matchings in $\mathcal{M}$ using just k colors. Indeed, we can simply color an edge belonging to a matching by a color representing the bin in which the corresponding item is packed. In constructing the bin packing instance, we have discarded at most

$$\sum_{k=1}^{p} \max_{v \in V} a_k(v) \le 3 \sum_{k=1}^{p} (p-k+1) = \frac{3}{2} p(p+1)$$

matchings, and they can be colored with a new color for each of them. In summary, the number of colors we need is at most the optimal number of bins of our bin packing instance plus $\frac{3}{2}p(p+1)$. An interesting feature of the results on the previous section is that they do not assume any conditions on p. We will see later that the optimal value for p is $\Theta(n^{1/3})$, which implies that the number of additional colors we need to accommodate the matchings not in $\mathcal{M}$ is $\frac{3}{2}p(p+1) = O(n^{2/3}) = o(n)$ and hence negligible.

As an example of the associated bin packing instance, consider the case with $p = 3$ and $\alpha_1 = \frac{1}{2}$, $\alpha_2 = \frac{1}{3}$, and $\alpha_3 = \alpha = \frac{1}{4}$. The bin packing instance then consists of $2n$ items of size 1, n items of size $\frac{1}{2}$, and n items of size $\frac{1}{3}$, and these items can be packed into $2n + \frac{n}{2} + \frac{n}{3} = \frac{17}{6}n$ bins (plus $O(1)$ bins for fractionally opened bins). The argument above regarding discarded items shows that we need $O(p^2) = O(1)$ additional bins. Using Lemma 2, we then obtain that $M(n, r) \leq \frac{17}{6}n + O(1)$. This derivation is essentially identical to the result of Du et al. [12], and the approach taken here can be viewed as an extension of it.

Our goal now is to focus on our general bin packing instance and analyze the number of bins it requires. Since all items in the bin packing instance have size at least $\alpha = \alpha_p$, it is clear that items whose size is more that $1-\alpha$ are forced to use a full bin in any feasible packing. Hence, without loss of generality, we can let $\alpha_1 = 1 - \alpha$. With this, an optimal packing always needs $n/(1 - \alpha)$ bins to pack items of size 1 plus a certain number of bins to pack the remaining items (of size $\alpha_1, \ldots, \alpha_p$).

3.4. A lower bound. A trivial lower bound on the number of unit bins required to pack our discrete instance is $n/(1-\alpha)$ bins (for the items of size greater than $1-\alpha$) plus the total size of the remaining items:

$$\frac{n}{1-\alpha} + \sum_{k=2}^{p} \alpha_{k-1} \left(\frac{1}{\alpha_k} - \frac{1}{\alpha_{k-1}} \right) n.$$

This can be lower bounded in the following way. Let $g : [\alpha, 1 - \alpha] \to \mathbb{R}$ be defined by $g(x) = 1/x^2$. As $n \int_{\alpha_k}^{\alpha_{k-1}} g(x)dx = (\frac{1}{\alpha_k} - \frac{1}{\alpha_{k-1}})n$ is the number of items of size α_{k-1} and $\alpha_{k-1} \geq x$ for any $x \in [\alpha_k, \alpha_{k-1}]$, we have that

$$\sum_{k=2}^{p} \alpha_{k-1} \left(\frac{1}{\alpha_k} - \frac{1}{\alpha_{k-1}} \right) n \geq n \int_{\alpha}^{1-\alpha} x g(x) dx$$

$$= \int_{\alpha}^{1-\alpha} \left(\frac{n}{x} \right) dx = n \ln \frac{1-\alpha}{\alpha}.$$

Therefore, from Lemma 2, we derive that our analysis cannot give an upper bound on $M_{[\alpha,1]}(n, r)$ better than

$$\min_{\alpha \in (0,1]} \max \left\{ \frac{2n}{1-\alpha}, \frac{n}{1-\alpha} + n \ln \frac{1-\alpha}{\alpha} \right\} = M \cdot n,$$

with $2.5569 \leq M \leq 2.5570$. The term $\frac{2n}{1-\alpha}$ comes from Lemma 2, while the other term is the bound just obtained. The value of α for which the minimum is attained is $\alpha \approx 0.2178117$. From now on, we fix α to be the argmin of the above expression. In what follows, we show that this lower bound is actually achievable by relating the number of bins required by our bin packing instance to a continuous bin packing instance and analyzing it. For this purpose, we assume that the α_i's in the definition of our bin packing instance are equally spaced in $[\alpha, 1-\alpha]$, i.e., $\alpha_{k-1} - \alpha_k = \Delta = \frac{1-2\alpha}{p-1}$ with $\alpha_1 = 1 - \alpha$ and $\alpha_p = \alpha$.

3.5. The continuous packing problem. We round our bin packing instance to a continuous bin packing problem for which packing strategies with sublinear waste exist. We first define what we mean by a continuous bin packing instance. Consider a finite positive measure μ with density g defined over $[a, b]$ (with $0 \leq a \leq b \leq 1$) and, for any integer q, consider a uniform discretization $a = x_1 < \cdots < x_q = b$ of the interval $[a, b]$. Let Q_n^q be the optimal number of bins needed to pack the bin packing instance in which, for all $1 \leq i < q$, there are $\lceil n\mu([x_i, x_{i+1}))\rceil$ items of size x_{i+1}. The *value* of our bin packing instance is then defined as $\lim_{q\to\infty} \lim_{n\to\infty} \frac{Q_n^q}{n}$. By simply considering the total size of the items, we see that the value of a continuous instance is never smaller than

$$\int_a^b x d\mu(x) = \int_a^b xg(x)dx.$$

We say that μ admits a perfect packing if we have equality

$$\lim_{q\to\infty} \lim_{n\to\infty} \frac{Q_n^q}{n} = \int_a^b x d\mu(x) = \int_a^b xg(x)dx.$$

The lower bound in the previous section suggests that we consider the continuous bin packing instance with the continuous density $g(x) = \frac{1}{x^2}$ over $x \in [\alpha, 1-\alpha]$. In the next section, we show that a result of Rhee and Talagrand [26] can be applied to prove that g actually admits a perfect packing. What we show now is that the difference between the number of bins we need in our discrete instance and the value of this continuous instance times n is $O(\frac{n}{p})$ and hence sublinear whenever p grows with n. For this purpose, we show that we can discard $O(n/p)$ items in our discrete instance and obtain an instance which is dominated by discrete realizations of our continuous instance. Indeed, as $\int_{\alpha_{k-1}}^{\alpha_{k-2}} g(x)dx = \frac{1}{\alpha_{k-1}} - \frac{1}{\alpha_{k-2}}$, the continuous instance would dominate the discrete instance if we had only $(\frac{1}{\alpha_{k-1}} - \frac{1}{\alpha_{k-2}})n$ items of size α_{k-1}. We therefore need to discard a number of items of size α_{k-1} equal to

$$\begin{aligned}&\left(\frac{1}{\alpha_k} - \frac{1}{\alpha_{k-1}}\right)n - \left(\frac{1}{\alpha_{k-1}} - \frac{1}{\alpha_{k-2}}\right)n \\ &= \frac{2\Delta^2}{\alpha_k\alpha_{k-1}\alpha_{k-2}}n \leq \frac{2\Delta^2}{\alpha^3}n.\end{aligned}$$

Over all values of k, this amounts to discarding $p\frac{2\Delta^2}{\alpha^3}n = \Theta(\frac{n}{p})$ items, and they can each be packed in a separate bin.

As announced, we show in the next section that g admits a perfect packing. This implies that the total number of colors needed to color any graph $G \in \mathcal{D}_r^n(\alpha, 1)$ is at most $M \cdot n + O(p^2) + O(n/p)$, which is optimized choosing $p = \Theta(n^{1/3})$. For the optimal choice of α, which is approximately 0.2178117, the previous quantity becomes

$$M \cdot n + O(n^{\frac{2}{3}}) < 2.557 \cdot n + O(n^{\frac{2}{3}}),$$

concluding the proof of Theorem 1.

3.6. Perfect packing.

Consider the positive measure μ defined over the interval $[\alpha, 1-\alpha]$ with density $g(x) = 1/x^2$ for the optimal parameter α just obtained. To show that a perfect packing exists, we decompose g as the sum of three other positive functions, f_1, f_2 and f_3, all of which allow perfect packing. Furthermore, all bins used for the items corresponding to f_i will contain exactly $i+1$ items. With this, μ is a mixture of the corresponding measures μ_1, μ_2 and μ_3. The decomposition is depicted in Figure 1.

Consider the following functions:

1. $f_1(x) = \begin{cases} g(1-x) & \text{if } x \in [\alpha, 1/2), \\ g(x) & \text{if } x \in [1/2, 1-\alpha], \\ 0 & \text{otherwise,} \end{cases}$
2. $f_2(x) = \begin{cases} g(x) - f_1(x) - c & \text{if } x \in [1/4, \beta), \\ d & \text{if } x \in [\beta, \delta), \\ g(x) - f_1(x) & \text{if } x \in [\delta, 1/2), \\ 0 & \text{otherwise,} \end{cases}$
3. $f_3(x) = \begin{cases} g(x) - f_1(x) & \text{if } x \in [\alpha, 1/4), \\ c & \text{if } x \in [1/4, \beta), \\ g(x) - f_1(x) - d & \text{if } x \in [\beta, \delta), \\ 0 & \text{otherwise.} \end{cases}$

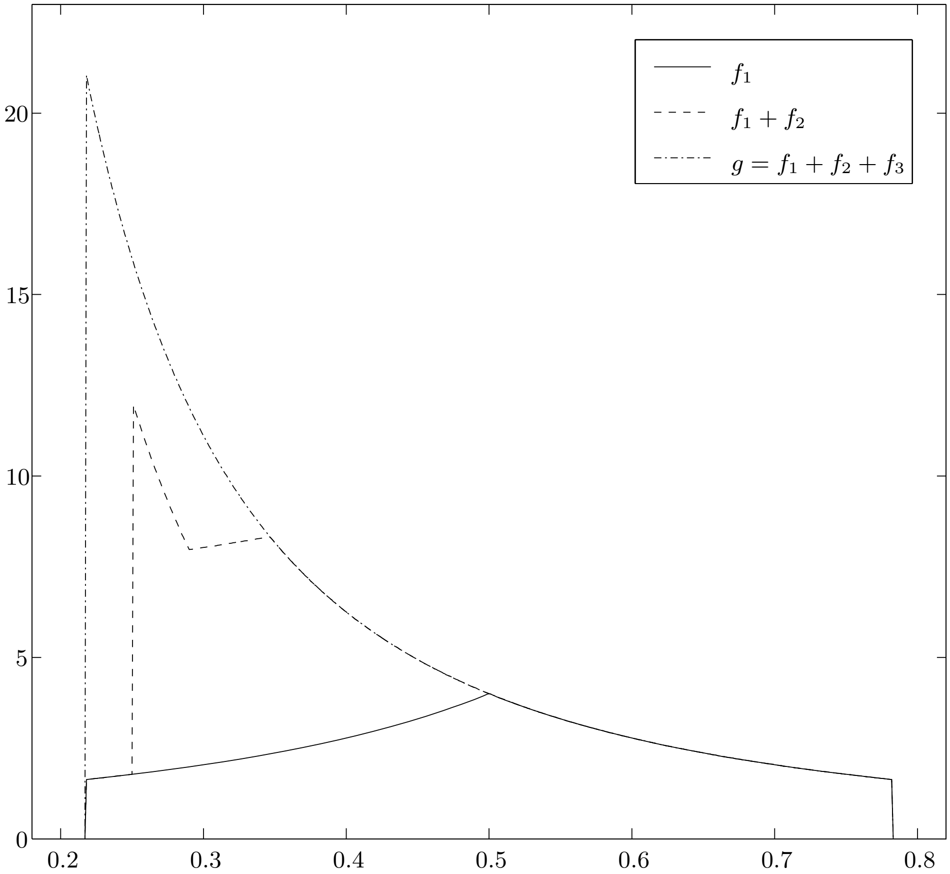


FIG. 1. *Decomposition of g into f_1, f_2, and f_3.*

Here $c = g(\beta) - f_1(\beta) - d$ and $d = g(\delta) - f_1(\delta)$ (so that f_2 is continuous). Clearly, for all $x \in [\alpha, 1-\alpha]$, $g(x) = f_1(x) + f_2(x) + f_3(x)$. The values of β and δ are uniquely determined by imposing that the average value of f_2 is $1/3$ and that of f_3 is $1/4$.

Namely, if $\beta \approx 0.2900708$ and $\delta \approx 0.3465256$, then

$$\frac{\int_{\alpha}^{1-\alpha} x f_1(x)dx}{\int_{\alpha}^{1-\alpha} f_1(x)dx} = \frac{1}{2},$$

$$\frac{\int_{1/4}^{1/2} x f_2(x)dx}{\int_{1/4}^{1/2} f_2(x)dx} = \frac{1}{3},$$

$$\frac{\int_{\alpha}^{\delta} x f_3(x)dx}{\int_{\alpha}^{\delta} f_3(x)dx} = \frac{1}{4}.$$

To prove that all f_1, f_2, and f_3 allow perfect packing, we use a perfect packing result proved by Karmarkar [17] and by Loulou [22] and a powerful theorem by Rhee and Talagrand [26]. The former result says that measures that are symmetric around $1/2^k$ for some integer k allow perfect packing. The latter can be stated as follows.

Theorem 6 (Rhee and Talagrand [26]). *Consider a decreasing measure μ defined over $[a, b]$ (with $0 \leq a \leq b \leq 1$) and an integer $p \geq 3$ such that $1/p \in [a, b]$. Then μ allows perfect packing if the following are satisfied:*

(i) $(p-1)a + b \leq 1$.

(ii) $\int_a^b x d\mu(x) = \frac{1}{p} \int_a^b d\mu(x)$.

In what follows, we briefly outline this result. Let $0 \leq a \leq b \leq c \leq 1$ be such that $(p-1)a + c \leq 1$ and $a + b < 2/p < a + c$. The *L-shaped function*, denoted by $L(a, b, c)$, is the unique (up to a multiplicative constant) nondecreasing real function defined over $[a, c]$, which is constant on $[a, b]$ and constant on $(b, c]$, and whose average value is $1/p$, i.e.,

$$\frac{\int_a^c xL(a, b, c)(x)dx}{\int_a^c L(a, b, c)(x)dx} = \frac{1}{p}.$$

In order to prove Theorem 6, Rhee and Talagrand first showed how to decompose a density satisfying the assumptions of the theorem as the limit of sum of L-shaped functions with the above properties. Then the central part of their work was to show that all such L-shaped functions do allow perfect packing. Unfortunately, they did not find a simple perfect packing strategy, and so they overcame the problem using a perfect packing characterization by Rhee [25], together with a complicated (and implicit) "exhaustion method," that decomposes an L-shaped function into possibly uncountably many perfectly packable functions.

Let us mention, however, that although the previous result was proved in a probabilistic setting (namely, under the following definition: μ allows perfect packing if and only if the expected number of bins needed to pack n independent and identically distributed random variables drawn according to μ divided by n approaches the expected size of an item), the proof also applies to our setting here.

Lemma 7. *The measure μ with density function $g : [\alpha, 1-\alpha] \to \mathbb{R}$ with $g(x) = 1/x^2$ allows perfect packing.*

Proof. As $g = f_1 + f_2 + f_3$, we need only show that each f_i, $i = 1, 2, 3$, allows perfect packing. The result follows immediately for f_1. Indeed, f_1 is symmetric around $1/2$. It remains to prove that both f_2 and f_3 satisfy the conditions of the previous theorem.

(1) The density f_2 is clearly decreasing in $[1/4, 1/2]$. Moreover,

$$\int_{1/4}^{1/2} x f_2(x) dx = \frac{1}{3} \int_{1/4}^{1/2} f_2(x) dx.$$

Finally, $(3-1)\frac{1}{4} + \frac{1}{2} = 1$. Thus all conditions are satisfied.

(2) Again, the density f_3 is decreasing in $[\alpha, \delta]$. In this case,

$$\int_{\alpha}^{\delta} x f_3(x) dx = \frac{1}{4} \int_{\alpha}^{\delta} f_3(x) dx,$$

and $(4-1)\alpha + \delta < 1$ (indeed, $(4-1)\alpha + \delta \approx 0.9999607$). □

3.7. Improved analysis for the rearrangeability of 3-stage Clos networks. In this section, we briefly discuss how a slight improvement of the $2.557n$ bound can be achieved when considering graphs belonging to $\mathcal{B}_r^n$. Specifically, we establish that $m(n, r) \leq 2.5480n + o(n)$. The analysis is essentially the same as the one for the bound on $M(n, r)$; therefore, we give only the main differences.

Let $G = (V, E) \in \mathcal{B}_r^n$. Since the weights satisfy condition (1), we can strengthen the main inequality used in Lemma 5 to be $\sum_{i=k}^{p} \deg_{D_i}(v) \leq 4n$ whenever $\alpha_p > 1/5$ (this is a strengthening only for $\alpha_k \leq 1/4$). This inequality, combined with the ideas in Lemma 5, can be used to prove the following result.

Lemma 8. *Let $G = (V, E) \in \mathcal{B}_r^n(\alpha, 1)$ and consider a sequence $\alpha_0 = 1 > \alpha_1 > \cdots > \alpha_l = 1/4 > \cdots > \alpha_p = \alpha > 1/5$. Then there exist sets $F_1, \dots, F_p$ partitioning E such that the following hold:*

(i) $\max_{e \in F_k} w(e) \leq \alpha_{k-1}$.

(ii) *For all vertices $v \in V$,*

$$\deg_{F_1}(v) \leq \frac{n}{\alpha_1} + a_1(v),$$

$$\deg_{F_k}(v) \leq \left(\frac{1}{\alpha_k} - \frac{1}{\alpha_{k-1}}\right) n + a_k(v), \qquad 2 \leq k \leq l,$$

$$\deg_{F_k}(v) \leq 16\,(\alpha_{k-1} - \alpha_k)\, n + a_k(v), \qquad l+1 \leq k \leq p,$$

where $a_k(v) \leq 3(p - k + 1)$.

By mimicking the analysis in section 3.3, the problem now translates into packing the function $g : [\alpha, 1-\alpha] \to \mathbb{R}$ such that $g(x) = 16$ if $x \in [\alpha, 1/4]$, and $g(x) = 1/x^2$ otherwise. The value of α now has to be taken a bit smaller than it used to be: $\alpha \approx 0.2151$ is the optimal choice. For that value of α, a decomposition of g very similar to that in section 3.6 can be found. Applying again the result in [26], such decomposition amounts to concluding that g allows perfect packing. The total number of colors needed is therefore

$$n \int_{\alpha}^{1-\alpha} x g(x) dx + \frac{n}{1-\alpha} + o(n) = \frac{2n}{1-\alpha} + o(n)$$

$$< 2.5480 \cdot n + o(n),$$

where the inequality comes from the choice of α.

4. Wide-sense nonblocking Clos networks. In what follows, we consider the online coloring formulation of the problem, assuming that edge weights satisfy condition (1). We start by describing two variants of the FF heuristic in the context of wide-sense nonblocking 3-stage Clos networks. Let $G = (V, E)$ be the bipartite graph with bipartition $V = A \cup B$ such that edge weights satisfy (1). Let $\{1, \ldots, M\}$ be the colors with which we attempt to find a valid coloring of G. Assume all edges in $\tilde{E} \subset E$ have been revealed and colored so far, and a new edge $e = (u, v) \notin \tilde{E}$ is revealed. The *first-fit-min (FF-Min)* heuristic assigns $c(e) = j$ (i.e., colors e with color j), where j is the smallest color for which adding e does not violate the valid coloring condition. In other words,

$$c(e) = j = \min \Bigg\{ 1 \leq i \leq M : w(e) + \sum_{f:\, f \in \delta(v) \cap \tilde{E},\, c(f)=i} w(f) \leq 1, \\ w(e) + \sum_{f:\, f \in \delta(u) \cap \tilde{E},\, c(f)=i} w(f) \leq 1 \Bigg\}.$$

In the first-fit-max (FF-Max) heuristic, the above minimization is replaced by a maximization.

The main issue now is to determine the smallest M such that FF-Min can always assign a color to a given edge. In order to establish our main result, we need a preliminary definition regarding the *blocking number* of $\mathcal{B}_r^n([0,1])$ under algorithm FF-Min. For an interval $I \subset [0, 1]$, we define the blocking number of $\mathcal{B}_r^n(I)$ under algorithm A (or simply the blocking number of algorithm A) as the maximum over all vertices $v \in V$ and over any graph $G = (V, E) \in \mathcal{B}_r^n(I)$ of the number of colors whose total weight adjacent to v is more than $1/2$. We denote it by $B_I(A)$:

$$B_I(A) = \max_{G=(V,E) \in \mathcal{B}_r^n(I)} \max_{v \in V} \left\{ \text{number of colors } i : \sum_{f:\, f \in \delta(v)\, c(f)=i} w(f) > \frac{1}{2} \text{ under A} \right\}.$$

By definition of $\mathcal{B}_r^n(I)$, we have that, for any algorithm A and for any $I \subseteq [0, 1]$, $B_I(A) \leq 2n - 1$. This bound, for interval $[0, 1]$, is exactly what we need to establish our main result.

We remark that our results do not hold for the more restrictive definition of wide-sense nonblocking in which the algorithm has to be able to route new connection requests *even if* previous connections terminate. In terms of the graph coloring problem, the more restrictive condition allows not only additions but also deletions of edges over time.

4.1. Wide-sense nonblocking for general connection requests. We now give the main result of this section, namely the bound on $m_W(n, r)$ in the general case.

THEOREM 9. *The number of colors needed to color any graph in $\mathcal{B}_r^n$ using algorithm FF-Min is at most $5n$, i.e.,*

$$m_W(n, r) \leq 5n.$$

Proof. Consider algorithm FF-Min, with $M = 5n$, applied to $G \in \mathcal{B}_r^n$. Let A, B be the bipartition of V, i.e., $V = A \cup B$. Let us say that an edge e is *large* if $w(e) > 1/2$ and *small* if $w(e) \leq 1/2$.

Consider iteration k of the algorithm and assume $e_k = e = (u, v)$ for some $u \in A$, $v \in B$. To see that the algorithm indeed works, we prove that edge e can be colored with some of $5n$ available colors. For this we consider two cases:

- Edge $e = e_k$ is small ($w(e) \leq 1/2$). Since FF-Min is a greedy-type heuristic, by Lemma 2, $\frac{2n}{1-1/2} = 4n$ colors are enough; and thus a color smaller than or equal to $4n$ is assigned to e.
- Edge $e = e_k$ is large ($w(e) > 1/2$). In this case, assume e cannot be colored. Let S^k_{uv} be the set of colors $1 \leq i \leq 5n$ such that there exists an edge e_t with $t < k$ satisfying that
  - e_t is colored with i,
  - $w(e_t) > 1/2$, and
  - e_t is adjacent to either u or v.

  The bound $B_I(A) \leq 2n-1$ for any algorithm implies that $|S^k_{uv}| \leq 2(2n-1) = 4n-2$. Now consider s, the smallest color in S^k_{uv}, such that $i > s$ implies that $i \in S^k_{uv}$. Since $|S^k_{uv}| \leq 4n-2$, we have that $s \geq n+3$. By definition, there is a small edge, say $f = (u, t)$, colored with $s-1$. The fact that f is small amounts to concluding that for all $i < s-1$, either

  $$\sum_{e:\, e\in\delta(u), c(e)=i} w(e) > \frac{1}{2} \quad \text{or} \quad \sum_{e:\, e\in\delta(t), c(e)=i} w(e) > \frac{1}{2}.$$

  The latter can happen for at most $2n-1$ colors (see the bound on the blocking number above), and therefore the former holds for at least $s-2-(2n-1) = s-2n-1$ colors. (As the former can happen only for at most $2n-1$ colors, this actually also implies $s \leq 4n$.) On the other hand, the number of large edges adjacent to u or v which are colored with $j \geq s$ is at least $5n-s+1$. Since, by condition (1), at most n of these can be adjacent to v, at least $5n-s-n+1 = 4n-s+1$ are adjacent to u.

  Overall we have that

  $$\sum_{e:\, e\in\delta(u)} w(e) > \frac{s-2n-1}{2} + \frac{4n-s+1}{2} = n,$$

  which contradicts (1). □

4.2. Improved bounds for the case of small connection requests. We now turn to the case in which all connection requests have weights in $[0, 1/2]$. In terms of our graph coloring problem, this means considering graphs in $\mathcal{B}^n_r([0, 1/2])$. Gao and Hwang [13] have proved that $m_{W[0,1/2]}(n, r) \leq 3.75n$. Let us now see how an improvement of this result can be obtained.

LEMMA 10. *The number of colors needed to color any graph in $\mathcal{B}^n_r([0, 1/2])$ using algorithm FF-Min is at most $3.601n + 3$, i.e.,*

$$m_{W[0,1/2]}(n, r) \leq 3.601n + 3.$$

Proof. Observe first that from Lemma 2, if e is an edge with weight $w(e)$, FF-Min actually assigns to it a color $c(e)$ satisfying $c(e) \leq \frac{2n}{1-w(e)} + 1$, or

$$w(e) \geq 1 - \frac{2n}{c(e)-1}. \tag{4}$$

This immediately implies that edges with weight below $\frac{1}{4}$ are assigned to the first $\lceil \frac{8n}{3} \rceil$ colors.

Let M be the number of colors needed by FF-Min and let $e = (u, v)$ be an edge that could not be assigned to any of the first $M-1$ colors. Consider a color $j \leq M-1$, since e was not assigned to j (and $w(e) \leq 1/2$); then $\sum_{f\colon f\in\delta(u),c(f)=j} w(f) > 1/2$ or $\sum_{f\colon f\in\delta(v),c(f)=j} w(f) > 1/2$. Assume, without loss of generality, that the latter holds. Since all weights are at most $1/2$, at least two edges in $\delta(v)$ are colored j, and thus

$$\sum_{f\colon f\in\delta(v),c(f)=j} w(f) \geq 2\left(1 - \frac{2n}{j-1}\right).$$

We can now compute $\sum_{g\in\delta(v)} w(g) + \sum_{g\in\delta(u)} w(g)$ using the previous equation, the fact that for a color $j \leq \lceil \frac{8n}{3} \rceil$ either $\sum_{f\colon f\in\delta(u),c(f)=j} w(f) > 1/2$ or $\sum_{f\colon f\in\delta(v),c(f)=j}$ $w(f) > 1/2$, and (4):

$$\begin{aligned}
\sum_{g\in\delta(v)} w(g) + \sum_{g\in\delta(u)} w(g) &> 2w(e) + \left\lceil \frac{8n}{3} \right\rceil \frac{1}{2} + \sum_{j=\lceil \frac{8n}{3} \rceil+1}^{M-1} 2\left(1 - \frac{2n}{j-1}\right) \\
&\geq \left\lceil \frac{8n}{3} \right\rceil \frac{1}{2} + \sum_{j=\lceil \frac{8n}{3} \rceil+1}^{M} 2\left(1 - \frac{2n}{j-1}\right) \\
&\geq 2M - \frac{3}{2}\left\lceil \frac{8n}{3} \right\rceil - 4n \int_{\lceil 8n/3 \rceil -1}^{M-1} \frac{1}{x} dx \\
&\geq 2M - 4n - \frac{3}{2} - 4n \int_{8n/3-1}^{M-1} \frac{1}{x} dx \\
&= 2M - 4n - \frac{3}{2} + 4n \ln\left(\frac{8n-3}{3M-3}\right) \\
&= 2M - 4n - \frac{3}{2} + 4n \ln\left(\frac{8n}{3M-9}\right) \\
&\quad + 4n \ln\left(\frac{(3M-9)(8n-3)}{(3M-3)8n}\right).
\end{aligned}$$

However, for $M \geq 3.601n + 3$, the above quantity surpasses $2n$, leading to a contradiction. Indeed, for this choice of M, the last term is greater than -4, and so the previous quantity is greater than $2 \cdot 3.601n - 4n + 4n \ln(8/10.803) > 2n$. □

4.3. The 2-rate environment. We now prove the bound on $m_W(n, r)$ when connection requests can take only two values, which are known beforehand. As mentioned before, this result almost closes the gap with the best known lower bound in this environment. Indeed, the result in this section, together with results in [13, 31], implies that

$$3n - 2 \leq m_W^2(n, r) \leq 3n.$$

In what follows, we denote by b and B the two rates (or edge weights) and assume that $0 < b < B \leq 1$. Gao and Hwang [13] already proved the bound in the case $B \leq 1/2$.

LEMMA 11 (Gao and Hwang [13]). *If $0 < b < B \leq 1/2$ are the two rates, then $m^2_W(n, r) \leq 3n$.*

We complete Gao and Hwang's result by proving a slightly better bound when $B > 1/2$. Of course, we may assume $b \leq 1/2$, for otherwise condition (1) allows us to reason that every vertex has degree at most n, and thus an even stronger bound of $2n$ holds for any online algorithm. Let k be the largest integer such that $B + kb \leq 1$ and ℓ be the largest integer such that $\ell b \leq 1$. Let us associate a *height* of 1 with every edge of weight b and a height of $(\ell - k)$ to every edge of weight B; and denote by $h(e)$ the height of an edge. As at most one item of size B can fit into a bin, (1) implies that the height of edges in any bin is at most l, and thus $\sum_{e\in\delta(v)} h(e) \leq n\ell$.

The algorithm we need to consider is the following.

```
ALGORITHM FF-MIN-MAX

(1) Assume the edges are revealed in the order $\{e_1, \ldots, e_m\}$.
(2) For $p = 1$ to $m$ do:
    (a) If $w(e_p) = B$, assign a color $1 \leq i \leq 3n-1$ to $e_p$ using FF-Max.
    (b) If $w(e_p) = b$:
        * Assign any color $1 \leq i \leq 3n - 1$ to $e_p$ such that at most
          $k - 1$ small edges adjacent to $u$ have been colored $i$, and
          at most $k - 1$ small edges adjacent to $v$ have been colored
          $i$, if one such color exists.
        * Otherwise, assign a color $1 \leq i \leq 3n - 1$ to $e_p$ using FF-
          Min.
```

LEMMA 12. *The number of colors needed to color any graph in $\mathcal{B}^n_r(\{b, B\})$, with $0 < b \leq 1/2 < B \leq 1$, using Algorithm FF-Min-Max is at most $3n - 1$.*

Proof. Consider the graph $G = (V, E) \in \mathcal{B}^n_r(\{b, B\})$ and assume the set of colors is $\{1, \ldots, 3n - 1\}$. For the purpose of this proof, let us say that an edge e is *large* if $w(e) = B$ and *small* if $w(e) = b$.

Consider step (1) of the algorithm and let $e_i = (u, v) \in E$ be the edge currently considered.

Let us first see that in step (2)(b) of the algorithm, FF-Min does not attempt to color e_i using a color larger than $2n - 1$ (assuming thus that no color could be found such that either u or v has at most $k - 1$ edges adjacent to it of that color). Indeed, it is enough to observe that FF-Min is a greedy-type algorithm. With this in mind, assume that FF-Min could not color e_i (which is a small edge) with a color $j \leq 2n-1$. Then, for any color $j \leq 2n - 1$, one of the following is satisfied:

- There are ℓ small edges $f_1 \ldots, f_\ell$ such that $c(f_r) = j$ for all $r = 1, \ldots, \ell$, and either $\{f_1, \ldots, f_\ell\} \subseteq \delta(u)$ or $\{f_1, \ldots, f_\ell\} \subseteq \delta(u)$.
- There are k small edges $f_1, \ldots, f_k$ and one large edge f such that $c(f_r) = c(f) = j$ for all $r = 1 \ldots, k$, and either $\{f_1, \ldots, f_k, f\} \subseteq \delta(u)$ or $\{f_1, \ldots, f_k, f\} \subseteq \delta(u)$.

We can therefore assume that there is a set of n colors $c_1, \ldots, c_n$ such that for each c_r, ℓ small edges or k small and one large edge are colored with c_r and are adjacent to u (otherwise, the property is true with v). Since also e_i is adjacent to u, we obtain that the total height of edges adjacent to u is at least $n\ell + 1 > n\ell$, a contradiction with condition (1).

We conclude that the algorithm will always find a feasible color for a small edge.

Suppose now that the algorithm cannot assign any color to $e_i = (u, v)$ because e_i is a large edge ($w(e_i) = B$). Then Algorithm FF-Min-Max attempted to color e_i in step (2)(a) using FF-Max, but no color was found. We see in what follows that this is impossible.

Let j be the largest color having at least $k + 1$ small edges adjacent to either u or v. From the analysis above, no color i with $i \geq 2n$ has more than k small edges adjacent to it, and thus $j < 2n$. In addition, for any color i with $i > j$, there must already be a large edge colored i adjacent to either u or v, and as this can happen to at most $2n - 2$ edges, we have $j > n$. Thus, $n < j < 2n$. Without loss of generality, assume that at least $k + 1$ small edges of color j are adjacent to u and let $f = (u, t)$ be one of those edges that was colored using FF-min (there is at least one such f, since there are at least $k+1$ edges in total). Since f was colored by FF-min, for every color $1 \leq r \leq 3n - 1$, at least k small edges of color r are adjacent to u or t. Also, by definition of FF-min, for every color $1 \leq r < j$, there is a set of edges colored with r, of total height ℓ, adjacent to u or t (such a set consists of either ℓ small edges or k small and one large edge). Overall we have that

$$\sum_{e \in \delta(u) \cup \delta(t) : w(e) = b} h(e) + \sum_{e \in \delta(u) \cup \delta(t) : w(e) = B \text{ and } c(e) < j} h(e) \geq (3n-1)k + (j-1)(\ell - k) + 1,$$

where the final $+1$ comes from the fact that $k+1$ small edges adjacent to u are colored with j. Thus, since $n\ell$ is the maximum total height of edges adjacent to t,

$$(5) \qquad \sum_{e \in \delta(u) : w(e) = b} h(e) + \sum_{e \in \delta(u) : w(e) = B \text{ and } c(e) < j} h(e) \geq (3n-1)k + (j-1)(\ell - k) + 1 - n\ell.$$

On the other hand, for every color r with $j < r \leq 3n - 1$, there is a large edge of color r adjacent to u or v. Since at most $n - 1$ (already colored) large edges can be adjacent to v, at least $3n - 1 - j - (n - 1) = 2n - j$ large edges of color larger than j are adjacent to u. Thus,

$$\sum_{e \in \delta(u) :\, w(e) = B \text{ and } c(e) > j} h(e) \geq (2n - j)(\ell - k).$$

Combining this with (5) and the fact that e_i is large, we conclude that the total height of edges adjacent to u is at least

$$[\ell - k] + [(3n - 1)k + (j - 1)(\ell - k) + 1 - nl] + [(\ell - k)(2n - j)],$$

where the first term corresponds to the height of e_i, the second to inequality (5), and the third to large edges adjacent to u of color $r > j$. The above quantity equals $(3n - 1)k - n\ell + 2n(\ell - k) + 1 = (n - 1)k + n\ell + 1 > n\ell$, which is impossible. □

5. A simple algorithm for multirate rearrangeability. In this section, we reconsider the offline setting and present a simple algorithm that is multirate rearrangeably nonblocking and that uses no more than $8n/3$ colors. In comparison, the algorithm of section 3 uses $2.548n + o(n)$ colors but is more complex. The algorithm we consider is the following.

```
Algorithm FF-Min-Decreasing

(1) Sort the edges according to their weight such that $w(e_1) \geq w(e_2) \geq \cdots \geq w(e_m)$. Let $k = 1$.
(2) While $k \leq m$ do:
    (a) Assign a color $1 \leq i \leq \lceil 8n/3 \rceil$ to $e_k$ using FF-Min.
    (b) $k = k + 1$.
```

As is the case for FF decreasing for bin packing, Algorithm FF-Min-Decreasing can be implemented in time $O(n \log n)$. As it involves sorting, it is applicable only to the offline setting. We have the following result.

Theorem 13. *The number of colors needed to color any graph in $\mathcal{B}_r^n$ using Algorithm FF-Min-Decreasing is at most $\lceil 8n/3 \rceil$.*

Proof. Consider the graph $G = (V, E) \in \mathcal{B}_r^n$ and let k be the smallest index such that $w(e_k) \leq \frac{1}{4}$. As FF-Min-Decreasing is a greedy algorithm, we know from Lemma 2 that our algorithm will always be able to color $\{e_k, \ldots, e_m\}$. Thus we can assume that $w(e_i) > \frac{1}{4}$ for all $i = 1 \ldots, m$; i.e., we can assume $G = (V, E) \in \mathcal{B}_r^n(]1/4, 1])$.

Let $e = e_\ell = (u, v)$ be the first edge that could not be colored by FF-Min-Decreasing. We distinguish two cases:

(i) $w(e) = \alpha > 1/3$. In this case, we will prove that FF-Min-Decreasing colors e with a color no larger than $2n$. We define the function $g_\alpha : [0, 1] \to [0, 1]$ as

$$g_\alpha(x) = \begin{cases} 1 & \text{if } 1 - \alpha < x, \\ 1/2 & \text{if } \alpha \leq x \leq 1 - \alpha, \\ 0 & \text{if } x < \alpha \end{cases}$$

and consider the modified edge weights $w'(e_i) = g_\alpha(w(e_i))$. We know that $G_\ell = (V, \{e_1, \ldots, e_\ell\})$ together with w satisfies condition (1), and from the sorting step $w(e_i) \geq \alpha$ for all $i = 1, \ldots, \ell$. Thus, G, together with w', also satisfies condition (1). Now, as e could not be colored using the first $2n$ colors, for all $1 \leq i \leq 2n$ either

$$\sum_{e : e \in \delta(u), c(e) = i} w(e) > 1 - \alpha \quad \text{or} \quad \sum_{e : e \in \delta(v), c(e) = i} w(e) > 1 - \alpha.$$

We can then assume that for a set $B \subset \{1, \ldots, 2n\}$ with $|B| \geq n$ the first inequality holds. For $i \in B$, the previous condition implies that the edges in $\delta(u)$ colored with i are either one edge f with $w(f) > 1 - \alpha$ or two edges f, g with $w(f) \geq \alpha$ and $w(g) \geq \alpha$. In both cases,

$$\sum_{e : e \in \delta(u), c(e) = i} w'(e) = 1.$$

It follows that

$$\sum_{e : e \in \delta(u)} w'(e) > \sum_{e : e \in \delta(u), c(e) \in B} w'(e) \geq n,$$

a contradiction with condition (1).

(ii) $1/3 \geq w(e) = \alpha > \frac{1}{4}$. We define the function $f_\alpha : [0,1] \to [0,1]$ as

$$f_\alpha(x) = \begin{cases} 1 & \text{if } 1-\alpha < x, \\ 1/2 & \text{if } \frac{1-\alpha}{2} < x \leq 1-\alpha, \\ 1/4 & \text{if } \alpha \leq x \leq \frac{1-\alpha}{2}, \\ 0 & \text{if } x < \alpha \end{cases}$$

and consider the modified edge weights $w'(e_i) = f_\alpha(w(e_i))$. As in the previous case, $G_\ell = (V, \{e_1, \dots, e_\ell\})$, together with w, satisfies condition (1). Thus, from the sorting step $w(e_i) \geq \alpha$ for all $i = 1, \dots, \ell$, and it is easy to check that G_ℓ, together with w', also satisfies condition (1).
Additionally, as e could not be colored using the $\lceil \frac{8n}{3} \rceil$ colors, for all $1 \leq i \leq \lceil \frac{8n}{3} \rceil$ either

$$\sum_{e: e \in \delta(u), c(e) = i} w(e) > 1 - \alpha \quad \text{or} \quad \sum_{e: e \in \delta(v), c(e) = i} w(e) > 1 - \alpha.$$

We can then assume that for a set $B \subset \{1, \dots, \lceil 8n/3 \rceil\}$ with $|B| \geq 4n/3$ the first inequality holds. For $i \in B$, the previous condition implies that the edges in $\delta(u)$ colored with i are either one edge f with $w(f) > 1 - \alpha$; two edges f, g with at least one of them (the largest), say f, satisfying $w(f) > \frac{1-\alpha}{2}$; or three edges f, g, h (all with weights greater than α). In any case,

$$\sum_{e: e \in \delta(u), c(e) = i} w'(e) \geq \frac{3}{4}.$$

It follows that

$$\sum_{e: e \in \delta(u)} w'(e) > \sum_{e: e \in \delta(u), c(e) \in B} w'(e) \geq |B| \cdot \frac{3}{4} \geq n. \qquad \square$$

REFERENCES

[1] A. Bar-Noy, R. Motwani, and J. Naor, *The greedy algorithm is optimal for online edge coloring*, Inform. Process. Lett., 44 (1992), pp. 251–253.

[2] L. A. Bassalygo and M. S. Pinsker, *Complexity of optimum nonblocking switching networks without reconnections*, Problems Inform. Transmission, 9 (1974), pp. 64-66.

[3] J. Beetem, M. Denneau, and D. Weingarten, *The GF11 supercomputer*, in Proceedings of the 12th ACM Annual International Symposium on Computer Architecture, 1985, pp. 108–115.

[4] B. Beizer, *The analysis and synthesis of signal switching networks*, in Proceedings of the Symposium on Mathematical Theory of Automata, 1962, pp. 563–576.

[5] V. E. Benes, *Mathematical Theory of Connecting Networks and Telephone Traffic*, Academic Press, New York, 1965.

[6] V. E. Benes, *Optimal rearrangeable multistage connecting networks*, Bell System Tech. J., 43 (1964), pp. 1641–1656.

[7] P. Brucker, J. Hurink, and F. Werner, *Improving local search heuristics for some scheduling problems*—I, Discrete Appl. Math., 65 (1996), pp. 97–122.

[8] S.-P. Chung and K. W. Ross, *On nonblocking multirate interconnection networks*, SIAM J. Comput., 20 (1991), pp. 726–736.

[9] C. Clos, *A study of nonblocking switching networks*, Bell Systems Tech. J., 32 (1953), pp. 406–424.

[10] D. de Werra, *On some combinatorial problems arising in scheduling*, CORS. J., 8 (1970), pp. 165–175.

[11] R. Diestel, *Graph Theory*, Grad. Texts in Math. 173, Springer-Verlag, New York, 1997.

[12] D. Z. Du, B. Gao, F. K. Hwang, and J. H. Kim, *On multirate rearrangeable Clos networks*, SIAM J. Comput., 28 (1998), pp. 463–470.

[13] B. Gao and F. Hwang, *Wide-sense nonblocking for multirate* 3*-stage Clos networks*, Theoret. Comput. Sci., 182 (1997), pp. 171–182.

[14] P. Haxell, A. Rasala, G. Wilfong, and P. Winkler, *Wide-sense nonblocking WDM cross-connects*, in Proceedings of the 10th European Symposium on Algorithms (ESA 2002), Lecture Notes in Comput. Sci. 2461, Springer-Verlag, Berlin, 2002, pp. 538–550.

[15] A. J. Hoffman, *Generalization of a theorem of König*, J. Washington Acad. Sci., 46 (1956), pp. 211–212.

[16] A. Itoh, W. Takahashi, H. Nagano, M. Kurisaka, and S. Iwasaki, *Practical implementation and packaging technologies for a large-scale ATM switching system*, IEEE J. Selected Areas in Comm., 9 (1991), pp. 1280–1288.

[17] N. Karmarkar, *Probabilistic analysis of some bin-packing algorithms*, in Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science, 1982, pp. 107–111.

[18] N. Karmarkar and R. M. Karp, *The Differencing Method of Set Partitioning*, Technical report UCB/CSD 82/113, University of California, Berkeley, CA, 1982.

[19] D. König, *Graphok és alkalmazásuk a determinánsok és a halmazok elméletére*, Mathematikai és Természettudományi Értesitö, 34 (1916), pp. 104–119.

[20] G.-H. Lin, D.-Z. Du, X.-D. Hu, and G. Xue, *On rearrangeability of multirate Clos networks*, SIAM J. Comput., 28 (1999), pp. 1225–1231.

[21] G. Lin, D. Du, W. Wu, and K. Yoo, *On* 3*-rate rearrangeability of Clos networks*, in Advances in Switching Networks, DIMACS Ser. Discrete Math. Theoret. Comput. Sci. 42. AMS, Providence, RI, 1998, pp. 315–333.

[22] R. Loulou, *Probabilistic behavior of optimal bin-packing solutions*, Oper. Res. Lett., 3 (1984), pp. 129–135.

[23] R. Melen and J. S. Turner, *Nonblocking multirate networks*, SIAM J. Comput., 18 (1989), pp. 301–313.

[24] H. Q. Ngo and V. H. Vu, *Multirate rearrangeable Clos networks and a generalized edge coloring problem on bipartite graphs*, in Proceedings of the Fourteenth ACM-SIAM Symposium on Discrete Algorithms (SODA 2003), 2003, pp. 834–840.

[25] W. Rhee, *Optimal bin packing with items of random sizes*, Math. Oper. Res., 13 (1988), pp. 140–151.

[26] W. Rhee and M. Talagrand, *Some distributions that allow perfect packing*, J. Assoc. Comput. Math., 35 (1988), pp. 564–578.

[27] C. E. Shannon, *Memory requirements in a telephone exchange*, Bell Systems Tech. J., 29 (1950), pp. 343–349.

[28] P. Schuurman and T. Vredeveld, *Performance guarantees of local search for multiprocessor scheduling*, in Proceedings of the 8th Conference on Integer Programming and Combinatorial Optimization (IPCO 2001), Springer-Verlag, Berlin, 2001, pp. 370–382.

[29] D. Slepian, *Two Theorems on a Particular Crossbar Switching*, manuscript, 1958.

[30] J. S. Turner and R. Melen, *Multirate Clos networks*, IEEE Comm. Mag., 41 (2003), pp. 38–44.

[31] K.-H. Tsai, D.-W. Wang, and F. Hwang, *Lower bound for wide-sense nonblocking Clos network*, Theoret. Comput. Sci., 261 (2001), pp. 323–328.

SIAM J. COMPUT.
Vol. 37, No. 3, pp. 895–922

THE RESOLUTION COMPLEXITY OF RANDOM CONSTRAINT SATISFACTION PROBLEMS[*]

MICHAEL MOLLOY[†] AND MOHAMMAD R. SALAVATIPOUR[‡]

Abstract. We consider random instances of constraint satisfaction problems where each variable has domain size d and each constraint contains t restrictions on k variables. For each (d, k, t) we determine whether the resolution complexity is a.s. constant, polynomial, or exponential in the number of variables. For a particular range of (d, k, t), we determine a sharp threshold for resolution complexity where the resolution complexity drops from a.s. exponential to a.s. polynomial when the clause density passes a specific value.

Key words. random constraint satisfaction problems, resolution complexity, random k-SAT, phase transition, sharp threshold

AMS subject classifications. 03F20, 60C05, 68R05, 68Q25

DOI. 10.1137/S0097539703436485

1. Introduction.

A constraint satisfaction problem (CSP) is a generalized form of satisfiability which is widely studied in the artificial intelligence community. For example, the journal *Constraints* is devoted to these problems. Roughly speaking, a CSP generalizes SAT[1] in the sense that variables can draw their values from a more general domain than simply $\{T, F\}$, and each clause (also known as a constraint) forms a set of restrictions on the values that the variables in the clause may jointly take.

Random instances of k-SAT have been extremely well studied over the past few decades (see [1] for many references). More recently, the interest in this area has expanded into random instances of various generalizations of k-SAT, such as NAE-SAT [3], XOR-SAT [14, 18, 19], $(2+p)$-SAT [35, 37, 38, 4, 2], and many others. All of these can be expressed as CSPs. It was natural for this interest to eventually spread to random instances of CSPs, rigorously in [5, 15, 33, 20, 31, 32, 40] and experimentally even earlier (see [24] for a good survey).

One of the most important results regarding random k-SAT is that of Chvátal and Szemerédi [13], who showed that for any $k \geq 3$ and $c > 0$ a random instance of k-SAT with n variables and cn clauses will almost surely (a.s.)[2] have no resolution proof of unsatisfiabilty of length less than $2^{\Theta(n)}$. It is easy to show that, for large values of c, such random instances are a.s. unsatisfiable. This immediately implied that, for sufficiently large values of c, any Davis–Putnam-style algorithm will take exponential time on such an input. Furthermore, it provided an astoundingly vast and rich class to the, beforehand rather sparse [26, 39], list of unsatisfiable instances of k-SAT for which there is no polytime resolution proof of unsatisfiability. Such instances are

[*] Received by the editors October 16, 2003; accepted for publication (in revised form) May 11, 2006; published electronically August 22, 2007. This work was supported by NSERC, a Sloan Research Fellowship, and a Premier's Research Excellence Award.
http://www.siam.org/journals/sicomp/37-3/43648.html

[†] Department of Computer Science, University of Toronto, Toronto, ON, M4E 3G1 Canada (molloy@cs.toronto.edu).

[‡] Department of Computing Science, University of Alberta, Edmonton, AB, T6G 2E8, Canada (mreza@cs.ualberta.ca).

[1] Instances of SAT are boolean formulas in conjunctive normal form. For k-SAT, all clauses contain k literals.

[2] Formal definitions of these and other terms will appear in the next section.

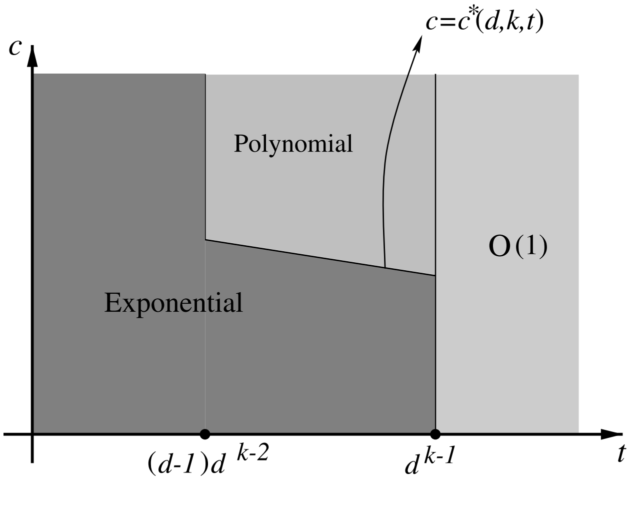


FIG. 1. *Resolution complexities.*

of great interest, since their existence can be viewed as a step toward proving that there are some unsatisfiable instances with no polytime proof of unsatisfiability of any kind, i.e., that $NP \neq co-NP$. Chvátal and Szemerédi's paper spawned numerous extensions and generalizations, e.g., [7, 8, 2, 6], including a general framework for proving lower bounds on resolution complexity by Ben-Sasson and Wigderson [9].

Mitchell [31, 32] extended the framework of Ben-Sasson and Wigderson to the setting of CSPs. He then used this framework to prove exponential lower bounds on the resolution complexity of a very natural class of random CSPs—one where the number of restrictions per constraint is fixed. Specifically, he considered random CSPs with domain size $d \geq 2$ and every constraint containing precisely t restrictions on $k \geq 2$ variables.[3] Note that these CSPs are trivial if either d or k is equal to 1 and that they are the well-studied 2-SAT when $d = k = 2$ and $t = 1$. Mitchell showed that, for $t \leq (d-1)/2, k = 2$, for $t \leq d-1, k \geq 3$, and for any constant $c > 0$, such a random instance with cn constraints will a.s. have no subexponential proof of unsatisfiability.[4] Again, it is easy to see that, for sufficiently large c, these instances are a.s. unsatisfiable (for $t > 0$). In contrast, Achlioptas et al. [5] showed that for $t \geq d^{k-1}$, and any $c > 0$, such a random instance will a.s. have an unsatisfiable subproblem of size $O(1)$ and thus will have a $O(1)$-length resolution proof of unsatisfiability. In this paper, we fill in the gap between $d-1$ and d^{k-1}. Using $\mathcal{F}_{n,M}^{d,k,t}$ to denote such a random CSP with M constraints, we prove the following theorems which are summarized in Figure 1.

THEOREM 1. *For any constants $d, k \geq 2$ and $1 \leq t < (d-1)d^{k-2}$, and for every constant $c > 0$, $\mathcal{F}_{n,M=cn}^{d,k,t}$ a.s. has resolution complexity at least $2^{\Theta(n)}$.*

For $d, k \geq 2$ and $t \geq (d-1)d^{k-2}$, we define

$$c^*(d,k,t) = \tfrac{1}{dk(k-1)}\binom{d^k}{t} \Bigg/ \binom{d^k-(d-1)d^{k-2}}{t-(d-1)d^{k-2}}.$$

[3] This natural model was, historically, one of the first two random models of a random CSP to be studied; the other turned out to be problematic and a.s. has $O(1)$ length resolution proofs of unsatisfiablity for any nontrivial number of constraints. See [33] or [24] for more details; the latter reference contains more than 30 references to the study of the model considered here.

[4] In [31], Mitchell claims to prove that this holds for $t \leq (d-1)(k-1)$ so long as d, k are not both 2. But there is an unfortunate error in his Lemmas 8 and 10, and his proof holds only for $t \leq (d-1)/2, k = 2$ and $t \leq d-1, k \geq 3$.

(An explanation of the derivation of this expression will have to wait until the end of the next section—see Lemma 9.)

THEOREM 2. *For any constants $d, k \geq 2$ and $(d-1)d^{k-2} \leq t < d^{k-1}$, and for every $c < c^*(d,k,t)$, $\mathcal{F}^{d,k,t}_{n,M=cn}$ a.s. has resolution complexity at least $2^{\Theta(n)}$.*

THEOREM 3. *For any constants $d, k \geq 2$ and $(d-1)d^{k-2} \leq t < d^{k-1}$, and for every constant $c > c^*(d,k,t)$, $\mathcal{F}^{d,k,t}_{n,M=cn}$ a.s. has resolution complexity* poly(n).

Trivially, the resolution complexity of a satisfiable CSP is infinite, so Theorem 2 is of no interest if, for all $c < c^*(d,k,t)$, $\mathcal{F}^{d,k,t}_{n,M}$ is a.s. satisfiable. This is, in fact, well known to be the case for $d = 2, k = 2, t = 1$ (i.e., 2-SAT), and we prove here that it is also the case for $d = 2, k = 3, t = 3$ (see Theorem 5 below). We prove that it is not the case for $d = 2, k = 3, t = 2$ and for all other d, k (note that when $d = 2, k = 3$, and t is as in Theorem 2, then $t \in \{2, 3\}$).

THEOREM 4. (a) *For any constants $d, k \geq 2$ and for every $c > \ln d / \ln[d^k/(d^k - t)]$, $\mathcal{F}^{d,k,t}_{n,M=cn}$ is a.s. unsatisfiable.*

(b) *For any constants $d, k \geq 2$, $(d,k) \notin \{(2,2),(2,3)\}$, and $(d-1)d^{k-2} \leq t < d^{k-1}$,*

$$\frac{\ln d}{\ln[d^k/(d^k - t)]} < c^*(d,k,t).$$

(c) *For every $c > 2.114$, $\mathcal{F}^{2,3,2}_{n,M=cn}$ is a.s. unsatisfiable.*

Parts (a) and (b) of Theorem 4 prove that $\mathcal{F}^{d,k,t}_{n,M}$ is a.s. unsatisfiable for some values of $c < c^*(d,k,t)$ for $d, k \geq 2$, $(d,k) \notin \{(2,2),(2,3)\}$, and $(d-1)d^{k-2} \leq t < d^{k-1}$. Part (c) proves the same for the case $d = 2, k = 3, t = 2$ since $c^*(2,3,2) = 7/3 > 2.114$. The next theorem shows that this is not the case for $d = 2, k = 3, t = 3$, and that here, just as in 2-SAT, there are short resolution proofs of unsatisfiability for every value of c above the threshold of satisfiability.

THEOREM 5. *For every $c < 7/9 = c^*(2,3,3)$, $\mathcal{F}^{2,3,3}_{n,M=cn}$ is a.s. satisfiable. Thus, $7/9$ is the (sharp) threshold of satisfiability for $\mathcal{F}^{2,3,3}_{n,M=cn}$.*

As mentioned above, when $t \geq d^{k-1}$, the resolution complexity of $\mathcal{F}^{d,k,t}_{n,M=cn}$ is a.s. $O(1)$. So these theorems completely characterize the resolution complexity of $\mathcal{F}^{d,k,t}_{n,M=cn}$ for every constant d, k, t, c except for $(d-1)d^{k-2} \leq t < d^{k-1}$ and $c = c^*(d,k,t)$. Here we have a sharp threshold for resolution complexity, similar to that found in [2], where the main technical result was the following.

THEOREM 6. *For any $\Delta, \epsilon > 0$, consider a random conjunctive normal form (CNF) formula F on n variables with Δn 3-clauses and $(1-\epsilon)n$ 2-clauses where every such formula is equally likely. F a.s. has resolution complexity at least $2^{\Theta(n)}$.*

(The other side of the "sharp threshold," i.e., that if the number of 2-clauses is $(1+\epsilon)n$ for some $\epsilon > 0$, then a.s. the resolution complexity of F is poly(n), was previously known to follow from the work in [12, 21, 25].)

Theorems 2 and 3 are at heart very similar to Theorem 6. For $(d-1)d^{k-2} \leq t < d^{k-1}$, a certain type of constraint called a *forcer* arises. Forcers play, essentially, the same role that 2-clauses play in random CNF formulas. We show that if $c > c^*$, then the forcers alone provide an unsatisfiable CSP with low resolution complexity, while if $c < c^*$, then, even along with the additional nonforcer constraints, the CSP has high resolution complexity.

Independently, Gao and Culberson [23] proved Theorem 3 for the special case $d = 2$. Essentially, they showed that in this case the forcers imply 2-clauses and for $c > c^*$ these 2-clauses form a random instance of 2-SAT which is above the satisfiability threshold. It is well known that such an instance will have low resolution complexity. We remark more on this at the end of subsection 2.3.

At first, we tried to adapt the lengthy proof of Theorem 6 to the setting of Theorem 2, but we were unsuccessful. Fortunately, we found an alternate proof technique, and to our pleasant surprise, it produced a proof of Theorem 2 which was dramatically shorter than the proof from [2] of Theorem 6. In fact, our technique yields a short proof of Theorem 6 which we provide in Appendix A. This technique looks like it will be of value to those who wish to prove future similar theorems.

We close this section by mentioning that the class of models of random CSPs considered here is a subset of the more general class introduced in [15, 33]. That class contains a much wider range of problems, including XOR-SAT and d-colorability. It would be very nice to characterize which models from that larger class exhibit high resolution complexity, but thus far we are unable to do so.

2. Preliminaries.

Here we give formal definitions of some of the concepts discussed in the introduction, along with other concepts required for the remainder of the paper.

2.1. The random model.

In our setting, the variables of our problem all have the same domain of permissible values, $\mathcal{D} = \{1, \dots, d\}$, and all constraints will be on k variables, for some fixed integers $d, k \geq 2$. Given a k-tuple of variables, $(x_1, \dots, x_k)$, a *restriction* on $(x_1, \dots, x_k)$ is a k-tuple of values $R = (\delta_1, \dots \delta_k)$, where each $\delta_i \in \mathcal{D}$. A set of restrictions on a k-tuple $(x_1, \dots, x_k)$ is called a *constraint* (or a clause). An assignment of values to the variables of a constraint C *satisfies* C if that assignment is not one of the restrictions in C. A CSP consists of a set of variables and a collection of constraints on subsets of those variables. An assignment of values to all variables in a CSP satisfies that CSP if every constraint is simultaneously satisfied. A CSP is *satisfiable* if there is at least one such satisfying assignment. The *degree* of a variable is the number of constraints in which it lies.

A *subproblem* of a CSP $\mathcal{I}$ is a CSP which is obtained by removing some of the variables and some of the constraints from $\mathcal{I}$, where, of course, if a variable x is removed, then every constraint containing x is also removed. When there is no possibility of confusion we often use, for example, $\mathcal{I} - \{C_1, C_2\}$ to denote the subproblem obtained by deleting the constraints C_1 and C_2 from $\mathcal{I}$ and $\mathcal{I} - \{x_1, x_2\}$ to denote the subproblem obtained by deleting the variables x_1 and x_2 from $\mathcal{I}$, along with any constraints containing them.

Recall that a k-uniform hypergraph is a generalization of a graph, where each edge contains k vertices. The *constraint hypergraph* of a CSP is the k-uniform hypergraph whose vertices correspond to the variables and whose edges correspond to the k-tuples of variables which have (nonempty) constraints. Of course, when $k = 2$, the constraint hypergraph is simply a graph, and so we often call it the *constraint graph.*

We define $\Omega^{d,k,t}$ to be the set of CSPs in which every variable has domain $\{1, \dots, d\}$, every constraint has k variables and t restrictions, and no two constraints use the same k-tuple of variables.

The random model: Specify c, n, d, k, t, and let $M = cn$. First choose a random constraint hypergraph with n vertices and M edges of size k, where each such hypergraph is equally likely. Next, for each edge e, we choose a random constraint on the k variables of e, with domains $\mathcal{D} = \{1, \dots, d\}$, uniformly from among all constraints with exactly t restrictions.

Note that every member of $\Omega^{d,k,t}$ with n variables and M clauses is equally likely to be chosen. We use $\mathcal{F}^{d,k,t}_{n,M}$ to denote a random CSP drawn from this model. We say that a property holds a.s. if the probability that it holds tends to 1 as n tends to infinity.

Remark. Alternatively, we could have chosen the constraint hypergraph by making an independent choice for each potential edge and deciding to put it in the hypergraph with probability $p = \frac{c \times k!}{n^{k-1}}$. We denote the resulting random CSP by $\mathcal{F}_{n,p}^{d,k,t}$. This model is, in many senses, equivalent to $\mathcal{F}_{n,M}^{d,k,t}$, as we describe in Appendix B. In particular, Lemma 25 implies that all of the theorems in this paper translate to $\mathcal{F}_{n,p}^{d,k,t}$. We will make use of the equivalence of these models in the proofs of Lemmas 15 and 16.

2.2. Resolution complexity. For a Boolean CNF formula F, a *resolution refutation* of F with length r is a sequence of clauses $C_1, \ldots, C_r = \emptyset$ such that each C_i is either a clause of F or is derived from two earlier clauses $C_j, C_{j'}$ for $j, j' < i$ by the following rule: $C_j = (A \vee x), C_{j'} = (B \vee \overline{x})$, and $C_i = (A \vee B)$, for some variable x. The *resolution complexity* of F, denoted $\mathbf{RES}(F)$, is the length of the shortest resolution refutation of F. (If F is satisfiable, then $\mathbf{RES}(F) = \infty$.)

Mitchell [32] discusses two natural ways to extend the notion of resolution complexity to the setting of a CSP. These two measures of resolution complexity are denoted $\mathbf{C-RES}$ and $\mathbf{NG-RES}$. The latter appears on the surface to be the most natural extension in that it extends resolution rules to the setting of a CSP and then carries them out. $\mathbf{C-RES}$, on the other hand, converts a CSP to a Boolean CNF formula and then carries out CNF resolution on that formula. Mitchell shows that, for every CSP instance $\mathcal{I}$, $\mathbf{C-RES}(\mathcal{I}) \leq \text{poly}(\mathbf{NG-RES}(\mathcal{I}))$, whereas there are many choices for $\mathcal{I}$ for which the converse is not true. Furthermore, all commonly used resolution-type CSP algorithms correspond nicely to the $\mathbf{C-RES}$ complexity of the input, but there are some that do not correspond to the $\mathbf{NG-RES}$. For that reason, we focus in this paper on the $\mathbf{C-RES}$ complexity, as did Mitchell in [31].

Given an instance $\mathcal{I}$ of a CSP in which every variable has domain $\{1, \ldots, d\}$, we construct a Boolean CNF formula CNF($\mathcal{I}$) as follows. For each variable x of $\mathcal{I}$, there are d variables in CNF($\mathcal{I}$), denoted $x : 1, x : 2, \ldots, x : d$, and there is a *domain clause* $(x : 1 \vee \cdots \vee x : d)$. For each restriction $(\delta_1, \ldots, \delta_k)$ on variables $(x_1, \ldots, x_k)$ in any constraint of $\mathcal{I}$, CNF($\mathcal{I}$) has a *conflict clause* $(\overline{x_1 : \delta_1} \vee \cdots \vee \overline{x_k : \delta_k})$. It is easy to see that CNF($\mathcal{I}$) has a satisfying assignment iff $\mathcal{I}$ does—if $\mathcal{I}$ has a satisfying assignment, then we produce one for CNF($\mathcal{I}$) by setting $x : \delta$ to True iff $x = \delta$; if CNF($\mathcal{I}$) has a satisfying assignment, then we produce one for $\mathcal{I}$ by setting $x = \delta$, where δ is any one of the values for which $x : \delta$ is True.

Remark. It is natural to consider adding an extra set of constraints for each variable x which specifies that $x : \delta$ can be true for at most one value of δ. But it is easily verified that each of the results in this paper (in particular, Lemma 7) holds regardless of whether we include these clauses; to be specific, we do not include them.

We define the resolution complexity of $\mathcal{I}$, denoted $\mathbf{C-RES}(\mathcal{I})$, to be equal to $\mathbf{RES}(\text{CNF}(\mathcal{I}))$.

In most previous papers bounding the resolution complexity of random instances of SAT or CSP for $k \geq 3$, a key lemma has been to establish that the following two conditions hold a.s. for some constants $\alpha, \zeta > 0$:

(A) Every subproblem on at most αn variables is satisfiable.

(B) Every subproblem on v variables, where $\frac{1}{2}\alpha n \leq v \leq \alpha n$, has at least ζn variables of degree 1.

For SAT, these two facts imply that a.s. the resolution complexity is exponential in n using principles introduced in [13] and refined to easily applied tools in [8, 9]. For more general instances of CSP, one needs to establish an additional fact (which is trivially true for SAT):

(C) If x is a variable of degree 1 in a CSP f, then, letting f' be the subproblem obtained by removing x and its constraint, any satisfying assignment of f' can be extended to a satisfying assignment of f by assigning some value to x.

In our setting, (C) holds if $t < d$, but for $t \geq d$, it is easy to see that it fails: Suppose that the constraint x lies in contains the d restrictions: $(1, 1, 1, \dots, 1, 1)$, $(1, 1, 1, \dots, 1, 2), \dots, (1, 1, 1, \dots, 1, d)$, where x is the last variable in the constraint. Then any satisfying assignment for f' in which all of the other variables of the clause receive 1 cannot be extended to f. For $k \geq 3$ Mitchell's proof [31] applies precisely to the range of t for which (C) holds. For $k = 2$, Mitchell modifies the conditions, replacing "degree 1" by "degree 2" in (B) and (C); this revised condition (C) holds precisely when $t \leq (d-1)/2$.

For higher values of t, we need to replace "degree 1" in condition (B) by a more complicated notion and then prove that something similar to condition (C) still holds. We describe how to do this in the next section, after presenting several necessary definitions.

2.3. Some new boundaries. A constraint C on variables $x_1, \dots, x_k$ *forbids* $x_i : \delta$ if each of the d^{k-1} possible k-tuples $(\delta_1, \dots, \delta_k)$, with $\delta_i = \delta$, is a restriction of C. Such a C is called a *forbidder.* As explained in [5] (and expanded on in [33]), it is the presence of forbidders that causes $\mathbf{C} - \mathbf{RES}(\mathcal{F}^{d,k,t}_{n,M=cn}) = O(1)$ a.s. for all c when $t \geq d^{k-1}$. C *permits* $(x_i : \delta, x_j : \gamma)$ if at least one of the d^{k-2} possible k-tuples $(\delta_1, \dots, \delta_k)$, with $\delta_i = \delta$ and $\delta_j = \gamma$, is not a restriction of C. C is a $(x_i : \delta) \rightarrow (x_j : \gamma)$ *forcer* if C does not permit $(x_i : \delta, x_j : \gamma')$ for any $\gamma' \neq \gamma$, i.e., if each of the $(d-1)d^{k-2}$ possible k-tuples $(\delta_1, \dots, \delta_k)$, with $\delta_i = \delta$ and $\delta_j \neq \gamma$, is a restriction of C. Thus C implies "if $x_i = \delta$, then $x_j = \gamma$." In this case, we say that the forcer C *starts* at x_i (or, more specifically, $x_i : \delta$) and *finishes* at x_j (or $x_j : \gamma$). As predicted by Mitchell [32], it is the presence of forcers that causes $\mathbf{C} - \mathbf{RES}(\mathcal{F}^{d,k,t}_{n,M=cn}) = \text{poly}(n)$ a.s. for large c when $(d-1)d^{k-2} \leq t < d^{k-1}$.

A *path* of length r in a k-uniform hypergraph H is a sequence of r edges $e_1, e_2, \dots, e_r$ such that:

- for $1 \leq i \leq r-1$, $e_i \cap e_{i+1} = x_i$—this is called a *connecting vertex*;
- for all $1 \leq i \leq r-2$ and $j > i+1$, $e_i \cap e_j = \emptyset$;
- there are specified vertices $x_0 \in e_1$ and $x_r \in e_r$, called the *endpoints* of the path.

If $e_1, \dots, e_r$ is a path and there is an edge $e_0 \in H$ whose intersection with the vertices of $e_1, \dots, e_r$ is only $\{x_0, x_r\}$, then $e_0, \dots, e_r$ form a *cycle* in H.

A *pendant path* is a path in which no vertices other than the endpoints lie in any edges of H off the path. In other words, there is no restriction on the degrees of the endpoints, each connecting vertex has degree 2 in H, and every other vertex in the path has degree 1 in H.

A (*pendant*) *path* of length r in a CSP is a sequence of r constraints whose underlying edges form a (pendant) path of length r in the underlying hypergraph. If there are values $\delta_0, \dots, \delta_r$ such that the constraint on each e_i is a $(x_{i-1} : \delta_{i-1}) \rightarrow (x_i : \delta_i)$ forcer, then we say that the (pendant) path is a $(x_0 : \delta_0) \rightarrow (x_r : \delta_r)$ *forcing (pendant) path.* It will be convenient to consider a single variable x to be a forcing path of length zero (note that we trivially have $(x : \delta) \rightarrow (x : \delta)$ for every δ); in this case, both endpoints of the forcing path are considered to be x.

A constraint on the edge e_i of a path is a P *forcer* if it is a $(x_{i-1} : \delta) \rightarrow (x_i, \gamma)$ forcer or a $(x_i : \gamma) \rightarrow (x_{i-1}, \delta)$ forcer for some δ, γ.

For any $\mathcal{I} \in \Omega^{d,k,t}$:

- the *first boundary* of $\mathcal{I}$, denoted by $\mathcal{B}^1(\mathcal{I})$, is the set of nonforbidding constraints of $\mathcal{I}$ which contain at most one variable of degree greater than 1;
- the *second boundary* of $\mathcal{I}$, denoted by $\mathcal{B}^2(\mathcal{I})$, is the set of pendant paths of length 4 in $\mathcal{I}$ which have no P forcers;
- the *third boundary* of $\mathcal{I}$, denoted by $\mathcal{B}^3(\mathcal{I})$, is the set of pendant paths of length 2 where one of the two constraints is a P forcer that starts at the connecting vertex, and the other is not a P forcer that finishes at the connecting vertex.

The *boundary* of $\mathcal{I}$ is $\mathcal{B}(\mathcal{I}) = \mathcal{B}^1(\mathcal{I}) \cup \mathcal{B}^2(\mathcal{I}) \cup \mathcal{B}^3(\mathcal{I})$.

Our main lemma corresponds to conditions (A) and (B) from section 2.2.

LEMMA 7. *Consider any $\mathcal{I} \in \Omega^{d,k,t}$ on n variables, where $t < d^{k-1}$. If, for some $\alpha, \zeta > 0$, we have the following:*

(a) *every subproblem on at most αn variables is satisfiable, and*

(b) *every subproblem $\mathcal{I}'$ on v variables, where $\frac{1}{2}\alpha n \leq v \leq \alpha n$, has $|\mathcal{B}(\mathcal{I}')| \geq \zeta n$, then $\mathbf{C} - \mathbf{RES}(\mathcal{I}) \geq 2^{\Theta(n)}$.*

To prove Lemma 7, we require the following lemma, which corresponds to condition (C) from section 2.2.

LEMMA 8. *Consider any $\mathcal{I} \in \Omega^{d,k,t}$, where $t < d^{k-1}$, and any $X \in \mathcal{B}(\mathcal{I})$. Any satisfying assignment of $\mathcal{I} - X$ can be extended to a satisfying assignment of $\mathcal{I}$.*

Proof. Suppose $X \in \mathcal{B}^1(\mathcal{I})$. Then the lemma follows from the fact that, since $t < d^{k-1}$, X cannot be a forbidding constraint.

Suppose $X \in \mathcal{B}^2(\mathcal{I})$, and consider any satisfying assignment of $\mathcal{I} - X$ where x_0, x_4 are assigned δ_0, δ_4. ($\mathcal{I} - X$ is the subproblem obtained by removing all clauses of X and all variables of X other than the endpoints.) Since e_1 is not an (x_0, x_1) forcer, there are at least two choices for δ_1 such that e_1 permits $(x_0 : \delta_0, x_1 : \delta_1)$. Similarly, there are at least two choices for δ_3 which can be assigned to x_3 so e_4 permits $(x_3 : \delta_3, x_4 : \delta_4)$. We will show that, for at least one of these four choices for the pair (δ_1, δ_3), there is a value δ_2 such that e_2 permits $(x_1 : \delta_1, x_2 : \delta_2)$ and e_3 permits $(x_2 : \delta_2, x_1 : \delta_3)$. If this were not the case, then for every $\delta_2 \in \{1, \dots, d\}$ either (i) e_2 does not permit $(x_1 : \delta_1, x_2 : \delta_2)$ for either of the two choices of δ_1 (this requires $2d^{k-2}$ restrictions) or (ii) e_3 does not permit $(x_2 : \delta_2, x_3 : \delta_3)$ for either of the two choices of δ_3 (this also requires $2d^{k-2}$ restrictions). Thus e_2, e_3 would have a total of at least $2d^{k-1}$ restrictions, which is not possible by hypothesis.

Suppose $X \in \mathcal{B}^3(H)$. Let the endpoints of X be x_0, x_2, the connecting variable of X be x_1, and the constraints of X be C_1, C_2, where C_1 is a forcer starting at x_1 and ending at x_0 and C_2 is not a forcer that starts at x_2 and ends at x_1. Consider any satisfying assignment of $\mathcal{I} - X$ where x_0, x_2 are assigned δ_0, δ_2. There are at least $d - 1$ choices for δ_1 such that C_1 permits $(x_0 : \delta_0, x_1 : \delta_1)$, and there are at least two choices for δ_1 such that C_2 permits $(x_1 : \delta_1, x_2 : \delta_2)$. At least one choice for δ_1 lies in the intersection of these sets, and so the lemma follows. □

With Lemma 8 in hand, the proof of Lemma 7 is straightforward, following Mitchell's framework [31].

Proof of Lemma 7. Consider any resolution refutation of CNF($\mathcal{I}$). Mitchell [31, Lemma 1], proves that hypothesis (a) implies there must be a clause C in the refutation and a subproblem $\mathcal{J}$ of $\mathcal{I}$ on between $\frac{1}{2}\alpha n$ and αn variables such that $\mathcal{J}$ minimally implies C in the following sense: (i) Every satisfying assignment of $\mathcal{J}$ satisfies C, and (ii) for subproblem $\mathcal{J}'$ of $\mathcal{J}$, there is a satisfying assignment of $\mathcal{J}'$ that does not satisfy C.

We will next prove that C must have at least $\zeta n/4$ variables.

Consider any clause $X \in \mathcal{B}^1(\mathcal{J})$; we will show that some variable of X appears in C. To see this, consider any assignment α which satisfies $\mathcal{J} - X$ but does not satisfy C. By Lemma 8, it is possible to extend α to a satisfying assignment α' of $\mathcal{J}$, and, since $\mathcal{J}$ implies C, α' satisfies C. Thus, there is some variable of C that is assigned a value in α' but not in α; that variable must be in X.

Nearly identical arguments show that C must contain a non-end-point variable of every member of $\mathcal{B}^2(\mathcal{J})$ and that C must contain the connecting variable from every member of $\mathcal{B}^3(\mathcal{J})$. No variable can be a non-end-point variable of more than four members of $\mathcal{B}^2(\mathcal{J})$. So C contains at least $|\mathcal{B}^1(\mathcal{J})| + |\mathcal{B}^2(\mathcal{J})|/4 + |\mathcal{B}^3(\mathcal{J})|$ variables. Since by hypothesis (b), $|\mathcal{B}(\mathcal{J})| \geq \zeta n$, C must contain at least $\zeta n/4$ variables, as required.

This allows us to apply the now standard "width lemma" of Ben-Sasson and Wigderson [9, Corollary 3.6], to prove our lemma. In particular, if we let $w_1 = k$ be the maximum clause size in CNF($\mathcal{I}$), and $w_2 \geq \zeta n/4$ be the minimum over all resolution refutations of CNF($\mathcal{I}$) of the maximum clause size in the refutation, then the width lemma states that

$$\mathbf{C} - \mathbf{RES}(\mathcal{I}) = e^{\Omega\left((w_2 - w_1)^2/n\right)} \geq 2^{\Theta(n)}. \qquad \square$$

We close this section with a lemma explaining the significance of $c^*(d, k, t)$.

LEMMA 9. *For $d, k \geq 2$ and $t \geq (d-1)d^{k-2}$, let $c = c^*(d, k, t) = \frac{1}{dk(k-1)}\binom{d^k}{t}/\binom{d^k-(d-1)d^{k-2}}{t-(d-1)d^{k-2}}$. Specify any variable x and value $\delta \in \{1, \dots, d\}$. The expected number of forcers in $\mathcal{F}^{d,k,t}_{n,M=cn}$ starting with $x : \delta$ is 1.*

Proof. Let $L = (d-1)d^{k-2}$. The expected number of constraints containing x is ck. For each of the $d(k-1)$ choices of $x' \neq x, \delta' \in \{1, \dots, d\}$ for a particular constraint containing x, the probability that the constraint forms a $(x\!:\!\delta) \rightarrow (x'\!:\!\delta')$ forcer is $\binom{d^k-L}{t-L}/\binom{d^k}{t}$, as there are exactly $\binom{d^k-L}{t-L}$ choices for such a forcer. Therefore, the expected number of forcers from $x : \delta$ is

$$cdk(k-1) \times \frac{\binom{d^k-L}{t-L}}{\binom{d^k}{t}} = \frac{\binom{d^k}{t}}{dk(k-1)\binom{d^k-L}{t-L}} \times dk(k-1) \times \frac{\binom{d^k-L}{t-L}}{\binom{d^k}{t}} = 1. \qquad \square$$

It is instructive to consider the case $k = d = 2$ and $t = (d-1)d^{k-2} = 1$, which is the more familiar random 2-SAT. Here every 2-clause can be viewed as the union of two forcers, e.g., $(x_1 \vee x_2)$ is equivalent to the conjunction of the two forcers $(x_1 : F) \rightarrow (x_2 : T)$ and $(x_2 : F) \rightarrow (x_1 : T)$. (Of course, we are considering the domain to be {T(rue),F(alse)} rather than $\{1, 2\}$. Note that $c^* = 1$, which is the satisfiability threshold for 2-SAT. The reader who is familiar with random 2-SAT will recognize that the property guaranteed by Lemma 9 corresponds very closely to what happens to cause the random 2-SAT to be unsatisfiable. Thus, it is not surprising that, for general d, k, at $c > c^*$ the forcers alone produce an unsatisfiable formula and that it, like random 2-SAT, has small resolution complexity.

For $d = 2$ and general k, it is easy to see that an $(x : F) \rightarrow (y : T)$ forcer is also an $(y : F) \rightarrow (x : T)$ forcer. Thus, such a forcer implies the 2-clause $(x \vee y)$. Extending this reasoning shows that for $c > c^*$ the forcers alone will contain a random instance of 2-SAT where the number of 2-clauses is above the satisfiability threshold. As mentioned earlier, this was discovered independently by Gao and Culberson [23].

3. Proof of Theorem 1.

We begin with a lemma of a type that has become standard in papers on the resolution complexity of random formulas. It says that a.s.

every subproblem of $\mathcal{F}_{n,M}^{d,k,t}$ with at most αn variables has a very low clause-vertex ratio. Thus, to prove that the conditions of Lemma 7 hold, it suffices to prove that certain types of subproblems must have a high clause-vertex ratio.

LEMMA 10. *Let $c > 0$ and $k \geq 2$, and let H be the random k-uniform hypergraph with n vertices and $m = cn$ edges. Then for any $\delta > 0$ there exists $\alpha = \alpha(c, k, \delta) > 0$ such that a.s. H has no subgraph with $0 < h \leq \lfloor \alpha n \rfloor$ vertices and at least $(\frac{1+\delta}{k-1})h$ edges.*

Proof. This proof follows a straightforward first moment calculation of a type that has been carried out many times in similar settings, starting with Łuczak [30].

Let $\mu = \frac{1+\delta}{k-1}$, $\mathcal{S}_h$ be the number of subgraphs of H with h vertices and *exactly* $\lceil \mu h \rceil$ edges, and set $\mathcal{S} = \sum_{h=1}^{\lfloor \alpha n \rfloor} \mathcal{S}_h$. Note that if $\mathcal{S}_h = 0$, then there are no subgraphs of H with h vertices and *at least* μh edges. To count $\mathbf{E}(S_h)$, we multiply the number of choices of h vertices by the number of choices for $\lceil \mu h \rceil$ of the cn random edges and the probability that each of those random edges lies entirely in that set of h vertices:

$$\mathbf{E}(\mathcal{S}_h) \leq \binom{n}{h} \binom{cn}{\lceil \mu h \rceil} \left(\frac{h}{n}\right)^{k\lceil \mu h \rceil}.$$

This yields:

$$\begin{aligned}
\mathbf{E}(\mathcal{S}) &= \sum_{h=1}^{\lfloor \alpha n \rfloor} \mathbf{E}(\mathcal{S}_h) \\
&\leq \sum_{h=1}^{\lfloor \alpha n \rfloor} \binom{n}{h} \binom{cn}{\lceil \mu h \rceil} \left(\frac{h}{n}\right)^{k\lceil \mu h \rceil} \\
&\leq \sum_{h=1}^{\lfloor \alpha n \rfloor} \left(\frac{en}{h}\right)^{h} \left(\frac{ecn}{\lceil \mu h \rceil}\right)^{\lceil \mu h \rceil} \left(\frac{h}{n}\right)^{k\lceil \mu h \rceil} \\
&\leq \frac{ec}{\mu} \sum_{h=1}^{\lfloor \alpha n \rfloor} \left[\left(\frac{h}{n}\right)^{(k-1)\mu - 1} e^{\mu+1} (c/\mu)^{\mu} \right]^{h} \\
&= \frac{ec}{\mu} \sum_{h=1}^{\lfloor \alpha n \rfloor} \left[\left(\frac{h}{n}\right)^{\delta} c' \right]^{h} \qquad \text{for } c' = e^{\mu+1}(c/\mu)^{\mu} \\
&\leq \frac{ec}{\mu} \left(\sum_{h=1}^{\lfloor \log n \rfloor} \left[\left(\frac{\lfloor \log n \rfloor}{n}\right)^{\delta} c' \right] + \sum_{h=\lfloor \log n \rfloor + 1}^{\lfloor \alpha n \rfloor} \left[\left(\frac{\lfloor \alpha n \rfloor}{n}\right)^{\delta} c' \right]^{\lfloor \log n \rfloor} \right) \\
&\leq O\left(\frac{\log^{1+\delta} n}{n^{\delta}}\right) + \sum_{h=\lfloor \log n \rfloor + 1}^{\lfloor \alpha n \rfloor} O\left(\frac{1}{n^2}\right) \qquad \text{for sufficiently small } \alpha \\
&= o(1). \qquad \square
\end{aligned}$$

The next lemma will allow us to show that subproblems with small boundaries must have a high clause-variable ratio and hence, by the previous lemma, must be large.

LEMMA 11. *Let $r \geq 2$ be a constant and H be a k-uniform hypergraph on n vertices and m edges that does not have any component which is a cycle. Let B_1 be the set of edges which have at most one vertex of degree greater than 1 and B_2 be the set of pendant paths of length r. If $|B_1| + |B_2| \leq n/(72r^2k^3)$, then for $\delta = \frac{1}{3rk^2}$: $m \geq n(\frac{1+\delta}{k-1})$.*

Intuitively, the lemma is clear: If $|B_1| = 0$ and if no vertex has degree greater than 2, then it is easy to see that we would have $m = n/(k-1)$. So, in order for m to not be much bigger, either B_1 must be large or there must be very few vertices of degree greater than 2. If the latter is true and B_1 is small, then H must contain long pendant paths, and so B_2 will be big. Our formal proof is a bit lengthy, and so we defer it to the end of the section.

These two lemmas are enough to prove the first of our main theorems.

Proof of Theorem 1. It suffices to prove that a.s. conditions (a) and (b) of Lemma 7 hold for $\mathcal{F}^{d,k,t}_{n,M=cn}$, where $\alpha = \alpha(c, k, \delta = 1/(12k^2))$ from Lemma 10 and $\zeta = \min(1/(72 \times 16k^3), \alpha/10k)$.

Since $t < (d-1)d^{k-2}$, $\mathcal{F}^{d,k,t}_{n,M}$ has no forcer constraints. Using this fact, our proof would follow immediately from Lemmas 10 and 11, if it were not for the fact that Lemma 11 applies only to hypergraphs with no cycle components.

We begin with condition (a). Suppose that $\mathcal{J}$ is a minimally unsatisfiable subproblem of $\mathcal{F}^{d,k,t}_{n,M}$. Thus, the underlying hypergraph of $\mathcal{J}$ is connected. Furthermore, since $t < (d-1)d^{k-2}$, it is easily verified that the underlying hypergraph of $\mathcal{J}$ cannot be a single cycle. Finally, Lemma 8 implies that $|\mathcal{B}^1(\mathcal{J})| = |\mathcal{B}^2(\mathcal{J})| = 0$. Therefore, since $\mathcal{J}$ has no forcers, Lemma 11 with $r = 4$ applies to the underlying hypergraph of $\mathcal{J}$, and so $\mathcal{J}$ has a clause-variable ratio of at least $(1+\delta)/(k+1)$. Thus Lemma 10 implies that a.s. $\mathcal{F}^{d,k,t}_{n,M}$ has no minimally unsatisfiable subproblems of size at most αn. Therefore a.s. $\mathcal{F}^{d,k,t}_{n,M}$ has no unsatisfiable subproblems of size at most αn.

Next is condition (b). We will use the easy fact, provided as Lemma 24 in Appendix B, that a.s. the underlying random hypergraph of $\mathcal{F}^{d,k,t}_{n,M}$ has fewer than $\log n$ cycles of length at most 4. Suppose, by contradiction, that $\mathcal{J}$ is a subproblem of $\mathcal{F}^{d,k,t}_{n,M}$ with v variables, where $\frac{1}{2}\alpha n \le v \le \alpha n$, and with $|\mathcal{B}^1(\mathcal{J})| + |\mathcal{B}^2(\mathcal{J})| \le \zeta n$. (Since there are no forcers, $|B^3(\mathcal{J})| = 0$.) Let H' be the subhypergraph obtained by removing all of the cycle components from the underlying hypergraph of $\mathcal{J}$. By Lemma 11 (with $r = 4$), H' has at least $|H'|\frac{1+\delta}{k-1}$ edges, and note also that $|H'| \le |\mathcal{J}| \le \zeta n < \alpha n$. By Lemma 10, a.s. every such H' is empty. Thus a.s. for every such subproblem $\mathcal{J}$, every component in the underlying hypergraph of $\mathcal{J}$ is a cycle. Every vertex in such a cycle of length at least 5 must lie in a member of $\mathcal{B}^2(\mathcal{J})$, and every member of $\mathcal{B}^2(\mathcal{J})$ contains fewer than $4k$ vertices; so there are at most $4k\zeta n$ vertices in those cycles. As mentioned above, a.s. there are at most $4 \log n$ vertices which lie in cycles of length at most 4 in $\mathcal{F}^{d,k,t}_{n,M}$. Since $4k\zeta n + 4\log n < \frac{1}{2}\alpha n \le |\mathcal{J}|$, we have a contradiction. □

We now close this section with the proof of Lemma 11.

Proof of Lemma 11. We say that a pendant path p of length at least 1 is *contractible* if (i) both of its endpoints have degree 2 and (ii) p is maximal in the sense that it is not part of a longer pendant path whose endpoints both have degree 2. Let $\mathcal{P}$ be the set of contractible pendant paths in H.

We form a hypergraph H' with no long pendant paths as follows.

For each $p \in \mathcal{P}$ with endpoints x, y, we remove all edges and vertices of p except for x, y and do the following: If x, y do not both lie in some edge outside of p, then we contract x, y into a single vertex. Otherwise, we create a new edge containing x, y and $k-2$ new degree 1 vertices; this edge is called a *reduced edge*, and x, y are its endpoints. Note that, since H has no cycle components, this operation does not create any new contractible paths, and it does not destroy any contractible paths. So there is no need to iterate this process, and we contract every path from $\mathcal{P}$. For convenience, we first contract all paths of length at least r in phase A and then all of the remaining paths in phase B.

The resulting hypergraph is H'. Every nonreduced edge in H' either (i) has a vertex of degree at least 3, (ii) has at least three degree 2 vertices, or (iii) is in B_1.

n' is the number of vertices in H'; s is the number of vertices of degree greater than 1; m' is the number of edges in H'; m_2 is the number of reduced edges; m^* is the number of edges with at least one degree 3 vertex.

Since H has no cycle components, the endpoints of any reduced edge must lie in a nonreduced edge. Since those endpoints have degree 2, each edge can hold the endpoints of at most $k/2$ reduced edges. Therefore, $m_2 \leq m' \cdot \frac{k}{2}/(\frac{k}{2}+1) = m'(k/(k+2))$.

The sum of the degrees of the vertices is $n'+s$ plus the sum of all vertices v with $\deg(v) \geq 3$ of $\deg(v)-2$. Each such vertex v lies in $\deg(v)$ edges, and so by counting $(\deg(v)-2)/\deg(v) \geq \frac{1}{3}$ for each of those edges, it follows that this latter sum is at least $m^*/3$. Therefore, since the sum of the degrees of all vertices is km', we have

$$km' \geq n' + s + m^*/3.$$

Furthermore, by counting the number of degree 1 variables in each edge, we have

$$s \geq n' - (k-3)m' - (m^* + m_2) - 2|B_1|.$$

These two equations combine to yield

$$\begin{aligned} km' &\geq 2n' - (k-2)m' + (m' - 2m^*/3 - m_2) - 2|B_1| \\ &\geq 2n' - (k-2)m' + \frac{1}{3}(m' - m_2) - 2|B_1| \\ &\geq 2n' - m'\left(k - 2 - \frac{2}{3(k+2)}\right) - 2|B_1|, \end{aligned}$$

and so $m' \geq (2n' - 2|B_1|)/(2k-2-\frac{2}{3(k+2)}) \geq n'(\frac{1+2\delta}{k-1}) - 2|B_1|$, since $\delta < \frac{1}{3k(k+2)}$ and $2k-2-\frac{2}{3(k+2)} > 1$.

Now we observe that very few vertices were removed during phase A. Any pendant path of length $l \geq r$ contains at least $l-r+1$ pendant paths of length r. So the total of the lengths of all such paths in H is at most $r|B_2|$. Therefore, the number of vertices in the hypergraph at the end of phase A is at least $n - rk|B_2|$.

Now we consider what happened during phase B. Every time we contracted a contractible path p of length $l < r$ to a vertex, the net loss in edges was l and the net loss of vertices was $(k-1)l-1$, and each time we contracted one to an edge, those net losses were $l-1$ and $(k-1)(l-1)-1$, respectively. Thus, for some v, we lost v vertices and at least $v(r-1)/((r-1)(k-1)-1) > v(1+\frac{1}{rk})/(k-1) \geq v(\frac{1+2\delta}{k-1})$ edges during phase B, since $\delta \leq \frac{1}{2rk}$.

Therefore, the hypergraph remaining at the end of phase A has at least

$$(n - rk|B_2|)\left(\frac{1+2\delta}{k-1}\right) - 2|B_1| \geq n\left(\frac{1+\delta}{k-1}\right) + \frac{n\delta}{k-1} - 3r|B_2| - 2|B_1| > n\left(\frac{1+\delta}{k-1}\right)$$

edges. Since phase A does not increase the number of edges, this proves our Lemma. □

4. Proof of Theorem 2. A Z_q*-configuration* is a collection of q vertex-disjoint forcing paths in $\mathcal{I}$, each with possibly length zero (i.e., a single vertex), plus $q(\frac{1+\gamma}{k-1})$ other edges, each containing k endpoints of the paths, where $\gamma = 1/(300k^2)$.

For $\mathcal{I} \in \Omega^{d,k,t}$ and $t \geq (d-1)k^{d-2}$, let $\mathcal{P} = p_1, \ldots, p_q$ be a collection of forcing paths of $\mathcal{I}$ such that: (i) every vertex lies in exactly one of these paths, and (ii) it is not possible to transform $\mathcal{P}$ into a collection of $q-1$ paths meeting condition (i) by adding another forcer from $\mathcal{I}$. Obviously a collection of paths of length 0, one for each variable of $\mathcal{I}$, satisfies (i), and so some collection exists which satisfies (i) and (ii).

LEMMA 12. *For any $\mathcal{I}, \mathcal{P}$ as described above: If $|\mathcal{B}(\mathcal{I})| < q/(72000k^3)$ and if the underlying hypergraph of $\mathcal{I}$ has no cycle components, then $\mathcal{I}$ contains a Z_q-configuration.*

Proof. Suppose that, among the paths in $\mathcal{P}$, exactly $p_1, \ldots, p_r$ $(r \leq q)$ have at least one edge each; the others have length 0. Consider a path p_i $(1 \leq i \leq r)$, and suppose it has $l \geq 1$ edges. Let x_0 and x_l be the start and end points of this path, respectively; so p_i is a $(x_0\!:\!\delta_0) \to (x_l\!:\!\delta_l)$ forcer for some values δ_0, δ_l. Remove from $\mathcal{I}$ all of the clauses and variables of p_i other than x_0, x_l. Then add $k-2$ new variables and a $(x_0\!:\!\delta_0) \to (x_l\!:\!\delta_l)$ forcer. The new forcer is called a *reduced* forcer. We do this for every $p_i, 1 \leq i \leq r$. The new CSP obtained after these operations is denoted by $\mathcal{I}'$. Note that n', the number of variables in $\mathcal{I}'$, is $q + r(k-1)$. Note also that since $p_1, \ldots, p_q$ are vertex-disjoint, no two reduced forcers in $\mathcal{I}'$ share a vertex.

CLAIM 13. *There is no forcer path of length at least 4 in $\mathcal{I}'$.*

Proof. By contradiction, assume that $p = e_1e_2e_3e_4$ is a forcer path in $\mathcal{I}'$, with u_{i-1} and u_i being the start and end points of e_i, respectively. If e_2 is not a reduced forcer, then adding the forcer e_2 to $\mathcal{P}$ would concatenate the path in $\mathcal{P}$ containing u_1 with the one containing u_2 without violating condition (i). This contradicts condition (ii). If e_2 is a reduced forcer, then e_3 cannot be (since no two reduced forcers share a variable). Thus a similar contradiction arises when we consider adding the forcer e_3 to $\mathcal{P}$. □

CLAIM 14. *Every pendant path of length 10 in $\mathcal{I}'$ has a subpath which is in $\mathcal{B}^2(\mathcal{I}) \cup \mathcal{B}^3(\mathcal{I})$.*

Proof. Let P' be a pendant path in $\mathcal{I}'$. By replacing each reduced edge of P' by its corresponding path from $\mathcal{P}$, we obtain a path P in $\mathcal{I}$. Every nonforcer in P' is a nonforcer in P. Assume $P = e_1, \ldots, e_l$, where $x_i = e_i \cap e_{i+1}$, and that e_a and e_b, $b > a$, are two nonforcers in P', and hence in P, such that there is no other nonforcer between them. If $b - a > 1$, then e_{a+1} is a forcer, and it must start at x_{a+1}; otherwise, $\{e_a, e_{a+1}\} \in \mathcal{B}^3(\mathcal{I})$, and we are done. Similarly, e_{b-1} must be a forcer starting at x_{b-2}; otherwise, $\{e_{b-1}, e_b\} \in \mathcal{B}^3(\mathcal{I})$. But these two imply that along the path $e_{a+1}, \ldots, e_{b-1}$ there is a member of $\mathcal{B}^3(\mathcal{I})$. Thus, we can assume that $b - a = 1$, i.e., that the nonforcers in P' are consecutive. Also, if i is the largest index for which e_i is a nonforcer in P, then $e_le_{l-1}\ldots e_{i+1}$ must be a forcing path going into x_i; otherwise, there is a member of $\mathcal{B}^3(\mathcal{I})$ along this path. Therefore the portions of P' on the sides of these nonforcers form forcing paths. Similar arguments show that if P' has no nonforcers, then it contains at most two forcing paths starting at the endpoints of P', or else P' contains a member of $\mathcal{B}^3(\mathcal{I})$.

By Claim 13, the length of each forcing path is at most 3. Therefore, if P' has length 10, then P' has at least 4 consecutive nonforcers, and so that subpath of P' is in $\mathcal{B}^2(\mathcal{I})$.

Let B_1 be the set of clauses which have at most one variable of degree greater than 1 in $\mathcal{I}'$ and B_2 be the set of pendant paths of length 10 in $\mathcal{I}'$. Note that $|\mathcal{B}^1(\mathcal{I})| \geq |B_1|$. Since no subpath can lie in more than 10 members of B_2, Claim 14 implies that $|B_2| \leq 10(|\mathcal{B}^2(\mathcal{I}) + \mathcal{B}^3(\mathcal{I})|)$. Therefore, $|B_1| + |B_2| \leq 10|\mathcal{B}(\mathcal{I})| \leq q/(7200k^3) < n'/(7200k^3)$, as n', the number of variables in $\mathcal{I}'$, is $q + r(k-1)$. So, applying Lemma 11 with $r = 10$ to the underlying hypergraph of $\mathcal{I}'$, the number of clauses in $\mathcal{I}'$ is at least

$n'(\frac{1+\gamma}{k-1})$, and at least $n'(\frac{1+\gamma}{k-1}) - r \geq q(\frac{1+\gamma}{k-1})$ are not reduced forcers. Those clauses and the paths in $\mathcal{P}$ will form a Z_q-configuration in $\mathcal{I}$. □

Lemma 15. *For any constants $d, k \geq 2$ and $(d-1)d^{k-2} \leq t < d^{k-1}$, and for every $c < c^*(d,k,t)$, there exists $\sigma > 0$, such that a.s. $\mathcal{F}^{d,k,t}_{n,M=cn}$ has no Z_q-configuration with $q \leq \sigma n$.*

Proof. For this proof, it will be convenient to work in the $\mathcal{F}^{d,k,t}_{n,p}$ model described in section 2.1, where each of the $\binom{n}{k}$ potential edges is chosen for the constraint hypergraph with probability $ck!/n^{k-1}$. Lemma 25 in Appendix B shows that proving this model a.s. has no Z_q-configuration with $q \leq \sigma n$ will imply that a.s. neither does $\mathcal{F}^{d,k,t}_{n,M=cn}$.

We compute the expected number of Z_q-configurations. To do so, we suppose that the q forcing paths are ordered $p_1, \ldots, p_q$ when we count the number of ways to choose them. Since the forcing paths of a Z_q-configuration are actually unordered, this produces an overcount, which we correct by dividing by $q!$.

We start with the computations related to the forcing paths: For each i, let $a_i \geq 0$ be the number of forcers in p_i, and set $A = \sum_{i=1}^{q} a_i$. The number of ways to choose the endpoints of the paths is at most n^{2q} (fewer if some of the path lengths are 0). For each p_i, there are at most n^{a_i-1} choices for the connecting variables and d^{a_i+1} choices of values to use on the variables to form a forcing path. Also, for each edge of each p_i, there are $\binom{n}{k-2}$ choices for the set of degree 1 variables. The probability that all of the specified hyperedges exist is $(ck!/n^{k-1})^A$. The probability that the constraints all form the specified forcers is, as we argued in the proof of Lemma 9, $(\binom{d^k-(d-1)d^{k-2}}{t-(d-1)d^{k-2}}/\binom{d^k}{t})^A$.

Now we turn our attention to the additional edges: There are at most $\binom{2q}{k}$ potential edges containing only endpoints of the paths, and we must choose $B = q(\frac{1+\gamma}{k-1})$ of them; there are $\binom{\binom{2q}{k}}{B}$ ways to do so. The probability that the B chosen edges are all present in H is $(ck!/n^{k-1})^B$.

Letting $c = (1-\epsilon)c^*$, this yields

$$\begin{aligned}
\mathbf{E}(|Z_q|) &\leq \frac{n^{2q}}{q!} \times \sum_{a_1,\ldots,a_q \geq 0} n^{A-q} d^{A+q} \binom{n}{k-2}^{A} \left[\frac{(1-\epsilon)c^* k!}{n^{k-1}}\right]^{A} \left[\frac{\binom{d^k-(d-1)d^{k-2}}{t-(d-1)d^{k-2}}}{\binom{d^k}{t}}\right]^{A} \\
&\qquad \times \binom{\binom{2q}{k}}{B} \left(\frac{ck!}{n^{k-1}}\right)^{B} \\
&\leq \frac{n^{2q}}{q!} \binom{\binom{2q}{k}}{B} \left(\frac{ck!}{n^{k-1}}\right)^{B} \times \sum_{a_1,\ldots,a_q \geq 0} \binom{n}{k-2}^{A} n^{A-q} d^{A+q} \left[\frac{(1-\epsilon)(k-2)!}{dn^{k-1}}\right]^{A} \\
&\leq \frac{n^{2q}}{q!} \left(\frac{(2q)^k e}{k!B} \frac{ck!}{n^{k-1}}\right)^{B} \times \sum_{a_1,\ldots,a_q \geq 0} \frac{n^{(k-2)A+A-q} d^{A+q}}{(k-2)!^{A}} \left[\frac{(1-\epsilon)(k-2)!}{dn^{k-1}}\right]^{A} \\
&\leq \frac{n^{2q}}{q!} \left(\frac{(2q)^k ec}{n^{k-1}(q/k-1)}\right)^{q(1+\gamma)/(k-1)} \times \frac{d^q}{n^q} \sum_{a_1,\ldots,a_q \geq 0} (1-\epsilon)^{A} \\
&\leq \left(\frac{\psi q}{n}\right)^{\gamma q} \left[\sum_{i \geq 0} (1-\epsilon)^{i}\right]^{q} \\
&\leq \left(\frac{\psi' q}{n}\right)^{\gamma q},
\end{aligned}$$

where $\psi, \psi' > 0$ are functions of $c, d, k, \epsilon, \gamma$, and hence of c, d, k, t since ϵ, γ can be derived from c, d, k, t. Therefore,

$$\sum_{q=1}^{\sigma n} \mathbf{E}(|Z_q|) \le \sum_{q=1}^{\lfloor \log n \rfloor} \left(\frac{\psi' q}{n}\right)^{\gamma q} + \sum_{q=\lfloor \log n \rfloor+1}^{\sigma n} \left(\frac{\psi' q}{n}\right)^{\gamma q}$$
$$\le O\left(\frac{\log^2 n}{n}\right) + \sum_{q=\lfloor \log n \rfloor+1}^{\sigma n} O(n^{-2}) = o(1)$$

for $\sigma > 0$ sufficiently small that $(\psi'\sigma)^\gamma < e^{-3}$. Thus σ is a function of c, d, k, t, γ and hence of c, d, k, t. □

LEMMA 16. *For $c < c^*(d, k, t)$, and for every constant integer $\theta > 0$, a.s. the number of variables of $\mathcal{F}_{n,M=cn}^{d,k,t}$ that are part of a maximal forcing path of length greater than θ is at most $3\theta d(c/c^*)^\theta$.*

Proof. Assume that $c = (1 - \epsilon)c^*$. We will again work in the $H_{n,p}$ model where $p = c \times k!/n^{k-1}$. Lemma 25 from Appendix B permits us to do so.

For any value of θ, let X_θ be the number of forcing paths of length θ, and let Y_θ be the number of maximal forcing paths of length at least θ. Obviously, X_θ is an upper bound for Y_θ (since for every forcing path of length θ there is a unique maximal forcing path of length at least θ). Our goal is to upper bound Y_θ for constant values of θ and for that we upper bound X_θ. We first compute the expected value of X_θ. We have to choose $\theta+1$ variables for the connecting points and the endpoints of the path; there are $\binom{n}{\theta+1}$ ways to do so. We order them; there are $(\theta+1)!$ ways to do so. Then we choose one of d values for each; there are $d^{\theta+1}$ ways to do so. Then we choose the remaining $k-2$ vertices for each of the θ constraints; there are $\binom{n-\theta-1}{(k-2)\theta}\frac{[(k-2)\theta]!}{(k-2)!^\theta}$ ways to do so. Finally, we multiply by the probability that the edge for each constraint is chosen in the underlying hypergraph and that the random constraints chosen for those edges are the specified forcers; the first of these probabilities is p^θ, and the second is $(dk(k-1)/c^*)^\theta$ (the latter computation uses the same arguments found in the proof of Lemma 9). This yields

$$\mathbf{E}(X_\theta) = \binom{n}{\theta+1}(\theta+1)!d^{\theta+1}\binom{n-\theta-1}{(k-2)\theta}\frac{[(k-2)\theta]!}{(k-2)!^\theta}\left(\frac{(1-\epsilon)(k-2)!}{dn^{k-1}}\right)^\theta$$
$$= \frac{n!}{[n-\theta-1-(k-2)\theta]!} \cdot \frac{(1-\epsilon)^\theta d}{n^{(k-1)\theta}}$$
$$= (1+o(1))nd(1-\epsilon)^\theta \qquad \text{for constant values of } \theta.$$

Now we bound the probability of $X_\theta > 2\mathbf{E}(X_\theta)$ using the second moment method. By Chebychev's inequality, this probability is at most $(\mathbf{E}(X_\theta^2) - \mathbf{E}(X_\theta)^2)/\mathbf{E}(X_\theta)^2$. So to prove that this probability is $o(1)$, it will suffice to prove that $\mathbf{E}(X_\theta^2) \le \mathbf{E}(X_\theta)^2(1+o(1))$.

Consider a fixed forcing path A of length θ. For each $i, j \ge 1$ we bound the number of potential forcing paths B of length θ which have exactly i constraints in common with A and for which these i constraints form j segments in A. To compute this, we first choose the j segments by choosing the vertices of their endpoints from A and the positions of those endpoints in B; there are $\binom{\theta}{2j}^2$ ways to make this selection. Then we match the j segments of A to the j segments of B; there are $j!$ choices for this. Then we select $\theta+1-i-j$ points (the rest of the connecting variables of path

B) and a value for each; there are at most $(nd)^{\theta+1-i-j}$ ways to do this. Then for each of the $\theta - i$ constraints in $B - A$ we select the remaining $k-2$ variables; there are $\binom{n}{(k-2)(\theta-i)}\frac{[(k-2)(\theta-i)!]}{(k-2)!^{(\theta-i)}}$ possible choices. Finally we multiply by the probability that the $\theta - i$ constraints in $B - A$ are selected and are forcers; as in the previous calculation, this probability is $(\frac{(1-\epsilon)(k-2)!}{dn^{k-1}})^{\theta-i}$. For any potential forcing path A of length θ, let Q_A be its indicator variable.

$$\begin{aligned}
\mathbf{E}(X_\theta^2) &= \sum_{A,B} \mathbf{Pr}(Q_A = 1 \cap Q_B = 1)\\
&= \sum_{A} \sum_{B: A\cap B = \emptyset} \mathbf{Pr}(Q_A = 1)\mathbf{Pr}(Q_B = 1) + \sum_{A} \sum_{B: A\cap B \neq \emptyset} \mathbf{Pr}(Q_A = 1 \cap Q_B = 1)\\
&\le \mathbf{E}(X_\theta)^2 + \sum_{A} \mathbf{Pr}(Q_A = 1) \times \sum_{i=1}^{\theta}\sum_{j=1}^{i} \binom{\theta}{2j}^2 j! n^{\theta+1-i-j} d^{\theta+1-i-j}\\
&\qquad \times \binom{n}{(k-2)(\theta-i)} \frac{[(k-2)(\theta-i)!]}{(k-2)!^{(\theta-i)}} \left(\frac{(1-\epsilon)(k-2)!}{dn^{k-1}}\right)^{\theta-i}\\
&\le \mathbf{E}(X_\theta)^2 + \mathbf{E}(X_\theta) 9 \sum_{i=1}^{\theta}\sum_{j=1}^{i} \left(\frac{e^2\theta^2}{4j^2}\right)^{2j} j^j n^{1-j} d^{1-j} (1-\epsilon)^{\theta-i}\\
&\le \mathbf{E}(X_\theta)^2 + \mathbf{E}(X_\theta) \cdot nd(1-\epsilon)^{\theta} \sum_{i=1}^{\theta}\sum_{j=1}^{i} \left(\frac{\alpha\theta^4}{j^3 nd}\right)^{j} (1-\epsilon)^{-i} \quad \text{(for a constant } \alpha > 0)\\
&\le \mathbf{E}(X_\theta)^2 + \mathbf{E}(X_\theta)^2 \cdot (1 + o(1)) \cdot O\left(\frac{1}{n}\right).
\end{aligned}$$

Therefore, a.s. $X_\theta \le 2(1+o(1))nd(1-\epsilon)^\theta$. Thus the number of variables which lie on a maximal forcing path of length at least θ is a.s. at most $2(1+o(1))\theta nd(1-\epsilon)^\theta < 3\theta d(c/c^*)^\theta$. □

Proof of Theorem 2. The proof is nearly identical to that of Theorem 1, this time using Lemmas 12 and 15 rather than Lemmas 11 and 10. We will prove that conditions (a) and (b) of Lemma 7 hold with $\alpha = \frac{1}{2}\sigma$ and $\zeta = \alpha/(4\theta \times 72000k^3)$, where θ is a positive integer such that $3\theta d(c/c^*)^\theta \le \alpha/8$.

We start with condition (a). Let $\mathcal{J}$ be a minimally unsatisfiable subproblem of $\mathcal{F}_{n,M}^{d,k,t}$ and H be the underlying hypergraph of $\mathcal{J}$. Clearly H is connected. Next we show that H cannot be a single cycle. By way of contradiction suppose that H is a cycle with constraints $C_1, \ldots, C_\ell$. Since $t < d^{k-1}$ we have the following: For any constraint C_i and any pair of variable/value $(x : \delta)$, with $x \in C_i$, there is a satisfying assignment of values to the variables of C_i in which x gets value δ. Let $x_1 = C_1 \cap C_2$ and $x_\ell = C_1 \cap C_\ell$, and consider $\mathcal{J} - C_1$. Consider any value $\delta \in \mathcal{D}$. There is a satisfying assignment for C_1 in which x gets δ—let δ_2 be the value assigned to x_2 in that assignment. There is a satisfying assignment for C_2 in which x_2 gets δ_2—let δ_3 be the value assigned to x_3 in that assignment. Repeating this argument yields a satisfying assignment for $\mathcal{J} - C_1$ in which x_1 gets δ. Since this is true of every value δ, there are at least d pairs (δ, δ') in which there is a satisfying assignment for $\mathcal{J} - C_1$ in which x_1 gets δ and x_ℓ gets δ'. Since C_1 contains fewer than $d \times d^{k-2}$ restrictions, at least one of these pairs (δ, δ') is such that there is a satisfying assignment to C_1 in which x_1 gets δ and x_ℓ gets δ'. Therefore, $\mathcal{J}$ is satisfiable, which is a contradiction. Thus H is connected and is not a cycle. Since $\mathcal{J}$ is minimally unsatisfiable, Lemma 8 implies that $|\mathcal{B}(\mathcal{J})| = 0$. So by Lemma 12, $\mathcal{J}$ has a Z_q-configuration, and, by

Lemma 15, that configuration has $q \geq \sigma n$. Since the paths of the Z_q-configuration are vertex-disjoint, we have $|\mathcal{J}| \geq \sigma n > \alpha n$ as required.

Now we prove that condition (b) of Lemma 7 holds. Consider a subproblem $\mathcal{J}$ on $\frac{1}{2}\alpha n \leq v \leq \alpha n$ variables, and let H' be the hypergraph remaining after removing all cycle components from the underlying hypergraph of $\mathcal{J}$.

Case 1. $|H'| \leq \frac{1}{4}\alpha n$. Since $|\mathcal{J}| \geq \frac{1}{2}\alpha n$, the total size of the cycle components is at least $\frac{1}{4}\alpha n$. By Lemma 24 in Appendix B, a.s. at most $\log n$ vertices lie on cycles of size at most 4. Every vertex on any other cycle component lies on a member of $\mathcal{B}^2(\mathcal{J})$, and each member of $\mathcal{B}^2(\mathcal{J})$ contains fewer than $4k$ vertices. So $|\mathcal{B}^2(\mathcal{J})| > \frac{1}{4k}(\frac{1}{4}\alpha n - \log n) > \frac{1}{20k}\alpha n > \zeta n$.

Case 2. $|H'| > \frac{1}{4}\alpha n$. Since $3\theta d(c/c^*)^\theta \leq \alpha/8$, Lemma 16 yields that at least $\alpha n/4 - \alpha n/8 > \alpha n/8$ of the variables in H' do not lie on any forcer paths of length at least θ. Thus, any collection of forcer paths which covers all of the variables of H' must contain at least $\frac{\alpha}{8\theta}n$ paths. So we can apply Lemma 12, where $\mathcal{I}$ is the CSP on H' formed by removing all cycle components from $\mathcal{J}$ and where $\frac{\alpha}{4\theta}n \leq q \leq |\mathcal{I}| \leq \alpha n$. Since $q \leq \alpha n < \sigma n$, Lemma 15 implies that a.s. the entire random CSP has no Z_q-configuration and so neither does H'. Since H' has no cycle components, Lemma 12 implies that $\mathcal{B}(\mathcal{I}) \geq q/(72000k^3) \geq \zeta n$. Every boundary element of $\mathcal{I}$ is also a boundary element of $\mathcal{J}$, and so this establishes condition (b). □

5. Proof of Theorem 3.

We will show that a.s. $\mathcal{F}^{d,k,t}_{n,M}$ contains a small unsatisfiable subproblem with a structure that is inspired by the snakes of [12]. Our proof is similar to the corresponding proof in [12].

A *forbidding cycle* is a $x : \delta \to x' : \delta'$ forcing path along with a $x' : \delta' \to x : \delta''$ forcer where $\delta \neq \delta''$. Thus, there is no satisfying assignment where $x = \delta$. We say that the cycle *forbids* $x : \delta$. Consecutive clauses in the cycle intersect in exactly one variable; such variables are called *connecting variables*.

An *r-flower* is the union of d forcing cycles $\mathcal{C}_1, \ldots, \mathcal{C}_d$ such that: (i) each has exactly r forcers; (ii) each cycle contains a particular variable x; (iii) no other variable lies in more than one of the cycles; (iv) each cycle C_i forbids $x : i$. We call x the *center variable*. Thus, any r-flower is unsatisfiable.

LEMMA 17. *For any constants $d, k \geq 2, d + k > 4$ and $(d-1)d^{k-2} \leq t < d^{k-1}$, and for every constant $c > c^*(d,k,t)$, $\mathcal{F}^{d,k,t}_{n,M=cn}$ a.s. contains all constraints of an r-flower, where $r = \lambda \log n$, for some sufficiently large constant λ.*

Theorem 3 follows immediately since if $\mathcal{F}^{d,k,t}_{n,M}$, contains an r-flower, with $r = \lambda \log n$, then we can use an exhaustive search to prove that the r-flower, and hence $\mathcal{F}^{d,k,t}_{n,M}$, is unsatisfiable in $\exp(r) = \text{poly}(n)$ steps. Such a proof can be simulated by a resolution proof with only a polynomial increase in length, so a.s. $\mathbf{RES}(\mathcal{F}^{d,k,t}_{n,M}) = \text{poly}(n)$. (The case $k = d = 2$, i.e., 2-SAT, is already known [12].)

Proof. For any potential flower A, we let X_A be the indicator variable for the event that the clauses of A all appear in the random CSP. With $X = \sum_A X_A$, it is enough to show that $\mathbf{E}(X^2) \leq \mathbf{E}(X)^2(1+o(1))$. Then the theorem follows easily from the Chebyshev inequality.

First, we compute $\mathbf{E}(X)$. We must choose $s = dr - d + 1$ connecting variables (including the center variable) and the $(k-2)dr$ other variables. There are $s!$ ways to arrange the connecting variables and then $[(k-2)dr]!/(k-2)!^{dr}$ ways to arrange the other variables into the flower. For each C_i, we need to choose some value other than i for the center variable and an arbitrary value for each of the other connecting variables. Then we multiply by the probability of all our forcers being present. We have $c = (1+\epsilon)c^*$ for some $\epsilon > 0$. For fixed variables x, y and values $u, v \in D$, the probability that there exists a $(x\!:\!u) \longrightarrow (y\!:\!v)$ forcer is $p = (1+\epsilon)\frac{(k-2)!}{dn^{k-1}}$, by the same calculations as in Lemma 9.

$$\begin{aligned}\mathbf{E}(X) &= \binom{n}{s} s! \binom{n-s}{(k-2)dr} \frac{[(k-2)dr]!}{(k-2)!^{dr}} \times (d-1)^d d^{(r-1)d} p^{dr}\\ &= (1+o(1))(1+\epsilon)^{dr}\left(\frac{d-1}{d}\right)^r n^{1-d}\\ &= \Omega(n^2)\end{aligned}$$

for sufficiently large λ. Next we compute an upper bound for $\mathbf{E}(X^2)$. Consider a fixed r-flower A and its underlying hypergraph H_A. For each $i, j \geq 1$ we will upper bound the number of potential r-flowers B that have exactly i constraints in common with A where these constraints form j connected components in H_A. (A very loose upper bound will suffice.) First, we consider choosing the j components. At most one component contains the center variable—for each C_i, such a component contains either all of C_i, none of C_i, or the portion of C_i between 2 variables. So there are at most $(2+r^2)^d$ choices for such a component. Each of the other components is specified by 2 variables on the same cycle. Thus, the number of choices for the components of H_A is at most $(2+r^2)^d(dr^2)^{j-1}$. To obtain a very loose upper bound on the number of ways that these components can fit into B, we simply multiply by the number of ways to choose j components from B and then multiply by $j!$ for the number of ways to pair them up with the components of A. Note that, since $t < d^{k-1}$ and d, k are not both 2, no constraint can be a $a : \delta \rightarrow b : \gamma$ forcer for more than one choice of a, b, δ, γ. Therefore, if an edge lies in the underlying hypergraph of both A and B, then its forcer in B must be identical to its forcer in A. Therefore, to choose the rest of B, we choose the remaining at most $s-i-j$ variables and a value for each of them (d values if one of them is the center variable) and then choose $k-2$ nonconnecting variables for each of the remaining clauses. Thus, the total number of potential such r-flowers is at most $((2+r^2)^d(dr^2)^{j-1})^2 j! n^{s-i-j} d^{s-i-j+d-1}\binom{n}{k-2}^{dr-i}$. Therefore, $\mathbf{E}(X^2)$ is

$$\begin{aligned}&\sum_{A,B}\mathbf{Pr}(X_A = 1 \wedge X_B = 1)\\ &= \sum_A \sum_{B:A\cap B=\emptyset}\mathbf{Pr}(X_A=1)\mathbf{Pr}(X_B=1) + \sum_A \sum_{B:H_A\cap H_B\neq\emptyset}\mathbf{Pr}(X_A = 1 \wedge X_B = 1)\\ &< \mathbf{E}(X)^2 + \sum_A \mathbf{Pr}(X_A = 1)\\ &\qquad \times \sum_{i=1}^{s}\sum_{j=1}^{i}\left[\left((2+r^2)^d(dr^2)^{j-1}\right)^2 \times j! n^{s-i-j} d^{s-i-j+d-1}\binom{n}{k-2}^{dr-i} p^{dr-i}\right]\\ &= \mathbf{E}(X)^2 + \mathbf{E}(X)\sum_{i=1}^{s}\left[(2+r^2)^{2d} r^{-4} n^{s-i} d^{s-i+d-3} \times \binom{n}{k-2}^{dr-i} p^{dr-i}\sum_{j=1}^{i}\left(\frac{dr^4 j}{n}\right)^j\right]\\ &\leq \mathbf{E}(X)^2 + \mathbf{E}(X) \times \mathbf{E}(X) \times O(r^{4d-4}) \times \sum_{i=1}^{s}(1+\epsilon)^{-i} O\left(\frac{r^4}{n}\right)\\ &\leq \mathbf{E}(X)^2\left(1 + O\left(\frac{r^{4d}}{n}\right)\right),\end{aligned}$$

which completes the proof since $r = O(\log n)$. □

6. Proofs of Theorems 4 and 5.

We close the paper with the proofs of Theorems 4 and 5. Part (a) of Theorem 4 uses a very standard technique whereby we compute the expected number of satisfying assignments. There are other standard techniques around which to improve this theorem (e.g., the techniques from [29]); we made no attempt to do so in part (a) as our only goal was to show that $\mathcal{F}^{d,k,t}_{n,M=cn}$ is a.s. unsatisfiable for some $c < c^*(d,k,t)$. However, we need to work a bit harder for part (c), and so we use the simplest of the techniques from [29].

Proof of Theorem 4. Part (a). Consider any instance $\mathcal{I}$ chosen from $\mathcal{F}^{d,k,t}_{n,M=cn}$. There are d^n assignments to the variables of $\mathcal{I}$. For each such assignment, the probability that all constraints are satisfied is easily seen to be $((d^k-t)/d^k)^{cn}$. Therefore, the expected number of assignments that satisfy $\mathcal{I}$ is

$$d^n\left(\frac{d^k-t}{d^k}\right)^{cn} = e^{(\ln d - c\ln(d^k/d^k-t))n},$$

which is $o(1)$, if $c > \frac{\ln d}{\ln(d^k/d^k-t)}$. This implies that for such c a.s. $\mathcal{I}$ is unsatisfiable.

Part (b). It is straightforward to verify that the statement holds for $d=2$ and $k\in\{4,5\}$. Also, $c^*(d,k,t) > \frac{1}{dk(k-1)} \times (\frac{d^k}{t})^{(d-1)d^{k-2}} > \frac{d^{(d-1)d^{k-2}}}{dk(k-1)}$. So it is enough to show that for $d\ge 2$, $k\ge 6$ and for $d\ge 3$, $3\le k\le 5$

$$\frac{\ln d}{\ln[d^k/(d^k-t)]} < \frac{d^{(d-1)d^{k-2}}}{dk(k-1)}. \tag{6.1}$$

A simple inductive argument shows that, for $k\ge 6$, $k(k-1)<2^k$ and $k<2^{k-2}-4$, which implies $2k(k-1) < 2^{2^{k-2}-3} \le d^{(d-1)d^{k-2}-3}$. Similarly, for $d\ge 3$ and $k\ge 3$, $2k(k-1) < 3^k$ and $k \le 2\times 3^{k-2}-3$, which implies $2k(k-1) < 3^{2\times 3^{k-2}-3} \le d^{(d-1)d^{k-2}-3}$. Therefore, in both cases

$$\begin{gathered}2dk(k-1)\ln d < d^{(d-1)d^{k-2}-2}\ln d,\\ \frac{2dk(k-1)\ln d}{d^{(d-1)d^{k-2}}} < \frac{\ln d}{d^2} < \frac{(d-1)d^{k-2}}{d^k-(d-1)d^{k-2}}.\end{gathered} \tag{6.2}$$

Using (6.2) and the fact that, for $0<x<1$, $e^x < 1+2x$

$$\begin{aligned}\exp\left(\frac{dk(k-1)\ln d}{d^{(d-1)d^{k-2}}}\right) &\le 1+\frac{(d-1)d^{k-2}}{d^k-(d-1)d^{k-2}}\\ &\le \frac{d^k}{d^k-t},\\ \frac{dk(k-1)\ln d}{d^{(d-1)d^{k-2}}} &< \ln\left(\frac{d^k}{d^k-t}\right),\\ \frac{\ln d}{\ln[d^k/(d^k-t)]} &< \frac{d^{(d-1)d^{k-2}}}{dk(k-1)},\end{aligned}$$

thus establishing (6.1) as required.

Part (c). For the case $d=2, k=3, t=2$, we strengthen the bound from part (a) by applying the so-called "1-flips" technique from [29]. We say that a satisfying assignment is *1-maximal* if, for every variable x with value 1, changing the value of x to 2 will result in a nonsatisfying assignment. It is easy to see that if an instance $\mathcal{I}$ chosen from $\mathcal{F}^{d,k,t}_{n,M=cn}$ is satisfiable, then it has a 1-maximal satisfying assignment; indeed, consider an assignment which, among all satisfying assignments, has the greatest number of variables with value 2.

Consider any instance $\mathcal{I}$ chosen from $\mathcal{F}^{d,k,t}_{n,M=cn}$. For each $0 \le a \le n$, consider one of the $\binom{n}{a}$ value assignments ν to the variables of $\mathcal{I}$ in which exactly a variables have value 1. As described in the proof of part (a), the probability that this is a satisfying assignment is $(6/8)^{cn}$. We condition on the fact that it is a satisfying assignment; the effect of this conditioning is that the M constraints are chosen uniformly at random from among the $\binom{7}{2}\binom{n}{3}$ constraints that are not violated by ν.

For ν to be 1-maximal, it must be the case that, for each variable x with value $\nu(x) = 1$, there is a constraint on x and two other variables, say, y, z, that contains $x = 2, y = \nu(y), z = \nu(z)$ as a restriction; we say that such a constraint *blocks* x. There are $6\binom{n-1}{2}$ possible constraints of this form, 6 for each choice of y, z. Conditional on ν being satisfying, the probability that no constraint blocks x is

$$\binom{\binom{7}{2}\binom{n}{3} - 6\binom{n-1}{2}}{M} \bigg/ \binom{\binom{7}{2}\binom{n}{3}}{M} = e^{-6c/7} + o(1).$$

Given two variables x_1, x_2, with $\nu(x_1) = \nu(x_2) = 1$, the events that at least one constraint blocks x_1 and at least one constraint blocks x_2 are not independent—given that one blocks x_1, it is less likely that that constraint blocks x_2, and so the probability that x_2 is blocked is a bit smaller. However, it is easy to see that this dependence goes in the right direction for our purposes, and so the probability that all a variables that are assigned 1 by ν are blocked is less than $(1 - e^{-6c/7} + o(1))^a$. Therefore, the expected number of 1-maximal satisfying assignments is

$$\sum_{a=0}^{n} \binom{n}{a} \left(\frac{6}{8}\right)^{cn} (1 - e^{-6c/7})^a = \left(\frac{6}{8}\right)^{cn} (2 - e^{-6c/7})^n = \left(\left(\frac{3}{4}\right)^c (2 - e^{-6c/7})\right)^n.$$

For $c > 2.114$ we have $(\frac{3}{4})^c(2 - e^{-6c/7}) < 1$, and so this expected number is $o(1)$. Thus a.s. $\mathcal{I}$ has no 1-maximal satisfying assignments and so $\mathcal{I}$ is unsatisfiable.

(Clearly, this technique will easily give an improvement to the bound in part (a) for any d, k, t. It is not hard to see that further techniques from [29] will obtain even better bounds.) □

Our final proof follows the, now rather standard, technique of using a differential equation analysis to show that a particular algorithm will a.s. find a satisfying assignment. See [1] for a good presentation of this method and survey of some of its most important applications.

Proof of Theorem 5. Recall that our assumption is that $c < c^*(2, 3, 3) = 7/9$ and that we wish to show that $\mathcal{F}^{2,3,3}_{n,M=cn}$ is a.s. satisfiable.

We will make use of the fact that $\mathcal{F}^{2,3,3}_{n,M=cn}$ has a sharp threshold in the sense of Friedgut's theorem [22]. This fact was first proven independently by Creignou and Daudé [16] and by Istrate [27]. These papers each proved special cases of a more general conjecture from [15], which was proved in [17], where Creignou and Daudé classified which members of a large family of random CSPs with domain size two exhibit a sharp threshold.

More formally, this fact says that there is a function $c^*(n)$ such that, for every $\epsilon > 0$, $\mathcal{F}^{2,3,3}_{n,M=(c^*(n)-\epsilon)n}$ is a.s. satisfiable and $\mathcal{F}^{2,3,3}_{n,M=(c^*+\epsilon)n}$ is a.s. unsatisfiable. This implies that if, for some constant c, $\mathcal{F}^{2,3,3}_{n,M=cn}$ is satisfiable with probability at least

γ for some $\gamma > 0$, then $\mathcal{F}^{2,3,3}_{n,M=c'n}$ is a.s. satisfiable for every constant $c' < c$: The only alternative is that $\mathcal{F}^{2,3,3}_{n,M=cn}$ and $\mathcal{F}^{2,3,3}_{n,M=c'n}$ are neither a.s satisfiable nor a.s. unsatisfiable. But this contradicts the existence of $c^*(n)$ since, for $\epsilon = (c - c')/3$, either $c' < c^*(n) - \epsilon$ or $c > c^*(n) + \epsilon$.

Thus, it will suffice to prove that, for every $c < 7/9$, $\mathcal{F}^{2,3,3}_{n,M=cn}$ is satisfiable with probability at least γ for some $\gamma = \gamma(c) > 0$, which we do now.

We consider the following algorithm, which we denote the *unit constraint* (UC).

The initial CSP is the input CSP, which is drawn from $\mathcal{F}^{2,3,3}_{n,M=cn}$. Repeatedly, we select a variable x and assign it a value i. We then modify each constraint C containing x as follows. If C contains any restrictions involving $x : i$, then we form a new constraint C' on the variables of C other than x, by taking each restriction of C that contains $x : i$, removing $x : i$ from that restriction, and placing the shortened restriction in C'. Thus, C' can be thought of as the constraint that is implied by C and setting $x = i$. Note that this might result in a constraint on exactly one variable in which each restriction simply dictates a value which that variable is not allowed to receive. We remove C and, if C contained any restrictions involving $x : i$, we replace it with C', unless:

- If C' is on two variables, say, a and b, and if there is some value j for which C' forbids both $(a = 1, b = j)$ and $(a = 2, b = j)$, then we simplify by replacing C with the constraint whose only variable is b and whose only restriction is (j), i.e., a 1-variable constraint that forbids b from taking the value j. If C' also forbids both $(a = j', b = 1)$ and $(a = j', b = 2)$, then we replace C by two 1-variable constraints which forbid $a = j'$ and $b = j$, respectively. Note that this latter case occurs iff C' contains 3 restrictions (since C' cannot contain more than $t = 3$ restrictions).
- If C has exactly one variable, and if it forbids $x = i$, then C' will contain no variables, so we remove C but do not add C', and we say that we formed a *null constraint*. This indicates that our assignment violated one of the original constraints. However, we will continue to run the algorithm, as this will be convenient for its analysis.

Since no constraint has more than $t = 3$ restrictions, it is easy to see that we will never generate a 1-variable constraint with two restrictions.

We choose x and i as follows:

- If there are any clauses on 1 variable, choose one of them uniformly at random, and set its variable so as to satisfy that clause. (As described above, no such clause will ever contain 2 restrictions, and so there is always a unique such setting for that variable.)
- Otherwise, pick x uniformly at random from all unset variables, and pick i uniformly at random from the domain $\{1, 2\}$.

After r variables have been set, we define the following:

- $C_3(r)$—the number of constraints on 3 variables,
- $H_1(r)$—the number of constraints on 2 variables with 1 restriction,
- $H_2(r)$—the number of constraints on 2 variables with 2 restrictions,
- $C_1(r)$—the number of constraints on 1 variable.

Note that the total number of remaining constraints is the sum of these values since every constraint formed has at least one restriction, and no constraint on 2 variables with 3 restrictions is ever added (since, instead of adding such a constraint, we would add the equivalent pair of constraints each on 1 variable).

CLAIM 18. *For each r, the CSP remaining after r steps of UC is uniformly random from amongst all CSPs with $C_3(r)$ constraints with 3 variables and 3 restrictions,*

$H_1(r)$ constraints with 2 variables and 1 restriction, $H_2(r)$ constraints with 2 variables and 2 restrictions, and $C_1(r)$ constraints with 1 variable and 1 restriction.

Proof. Consider two CSPs H_1 and H_2 each with $C_3(r)$ constraints with 3 variables and 3 restrictions, $H_1(r)$ constraints with 2 variables and 1 restriction, $H_2(r)$ constraints with 2 variables and 2 restrictions, and $C_1(r)$ constraints with 1 variable and 1 restriction. Consider any input CSP F_1 with n variables and with cn constraints, each having 3 variables and 3 restrictions, and any sequence of random choices of variables, such that running UC with input F_1 for r steps with that sequence of random choices will result in H_1. It is trivial to see how to modify F_1 into F_2, also with n variables and with cn constraints each having 3 variables and 3 restrictions, such that running UC with input F_2 for r steps with that same sequence of random choices will result in H_2. (Essentially, you simply replace all original constraints in F_1 that became constraints in H_1 with constraints that will instead become constraints of H_2.) This implies that the probability of ending up with H_1 is the same as the probability of ending up with H_2. This, in turn, implies the claim. □

Next, we consider the expected changes in the first 3 variables after step $r+1$. By examining all $\binom{8}{3}$ possible constraints on 3 variables, it is straightforward (but tedious) to verify that we have the following, regardless of whether step $r+1$ sets the variable of a 1-clause or a uniformly random unset variable:

- $\mathbf{Exp}(C_3(r+1) - C_3(r)) = -\frac{3C_3(r)}{n-r}$,
- $\mathbf{Exp}(H_1(r+1) - H_1(r)) = \frac{9}{7} \times \frac{C_3(r)}{n-r} - \frac{2H_1(r)}{n-r}$,
- $\mathbf{Exp}(H_2(r+1)) - H_2(r)) = \frac{3}{7} \times \frac{C_3(r)}{n-r} - \frac{2H_2(r)}{n-r}$.

Furthermore, the expected number of new 1-variable constraints that are formed during step $r+1$ is

$$F(r) = \frac{9}{7} \times \frac{C_3(r)}{n-r} + \frac{H_1(r)}{n-r} + \frac{2H_2(r)}{n-r}.$$

As is standard with this sort of analysis, our goal is to prove that a.s. $F(r)$ is always less than $1-\zeta$ for some $\zeta > 0$, as this will imply that, with sufficiently high probability, no null constraints are formed, i.e., that a.s. our assignment does not violate any of the original constraints.

Consider the following functions:

- $c_3(x) = c(1-x)^3$,
- $h_1(x) = \frac{9c}{7}x(1-x)^2$, and
- $h_2(x) = \frac{3c}{7}x(1-x)^2$.

Note that their derivatives satisfy

- $c_3'(x) = -\frac{3c_3}{1-x}$,
- $h_1'(x) = \frac{9}{7} \times \frac{c_3(x)}{1-x} - \frac{2h_1(x)}{1-x}$, and
- $h_2'(x) = \frac{3}{7} \times \frac{c_3(x)}{1-x} - \frac{2h_2(x)}{1-x}$

and that $C_3(0) = c_1(0)n = n$, $H_1(0) = h_1(0)n = 0$, and $H_2(0) = h_2(0)n = 0$.

We also need to note that, with very high probability, none of these parameters change much during any one iteration. In particular, it is straightforward to show that the probability of either C_3, H_1, or H_2 changing by more than $\log n$ during any one iteration is less than n^{-10}.

Noting the correspondence between these derivatives and the expected values computed above, Wormald's theorem [36] implies that for any $\alpha > 0$ a.s. for every $r \le (1-\alpha)n$ we have the following:

- $C_3(r) = c_3(r/n)n + o(n)$,

- $H_1(r) = h_1(r/n)n + o(n)$, and
- $H_2(r) = h_2(r/n)n + o(n)$.

(See [1] for a statement of Wormald's theorem and a good discussion of how to apply it in settings like this one.)

Thus, a.s. for every $r \leq (1-\alpha)n$, setting $x = r/n$ we have

$$F(r) = \frac{9}{7} \times \frac{c_3(x)}{1-x} + \frac{h_1(x)}{1-x} + \frac{2h_2(x)}{1-x} + o(1) = \frac{9c}{7}(1-x)\left(1 + \frac{2}{3}x\right) + o(1) \leq \frac{9c}{7} + o(1),$$

over the relevant range of $0 \leq x \leq 1$. Thus, a.s. $F(r) < 1 - \zeta$ for some small constant $\zeta > 0$, so long as $c < 7/9$.

We will run UC until a point where at least αn variables remain unset, for some particular small constant α to be named later. First, we bound the probability of creating a null constraint. The sequence $C_1(r)$ follows the pattern of a random walk on the positive integers, with a barrier at 0 and with drift always bounded above by $-\zeta$. Standard arguments imply that a.s. $\sum_{r=1}^{(1-\alpha)n} C_1(r) \leq Wn$ for some constant W. Note that, during step r, the probability that no null constraint is formed is (if $C_1(r) > 0$) equal to $(1 - \frac{1}{2(n-r)})^{C_1(r)-1} > (1 - \frac{1}{2\alpha n})^{C_1(r)}$. Therefore, the probability that no null constraints are formed at all is at least $(1 - \frac{1}{2\alpha n})^{Wn} = e^{-W/2\alpha} + o(1)$.

Again, using the fact that $C_1(r)$ has negative drift, it is straightforward to show that a.s. there is some $(1-\alpha)n - \log^2 n < r \leq (1-\alpha)n$ such that $C_1(r) = 0$. It will be convenient to halt UC there. With high probability, we are left with $c_3(1-\alpha)n + o(n) < \alpha^3 n$ constraints on 3 variables and $(h_1(1-\alpha) + h_2(1-\alpha))n + o(n) < 2\alpha^2 n$ constraints on 2 variables. Let G be the hypergraph whose vertices are the unset variables and where a group of vertices form a hyperedge iff they are the set of variables covered by a constraint.

It is straightforward to show that this random hypergraph G a.s. does not have a giant component. Perhaps the easiest way to see this is to add a third random vertex to each hyperedge of size 2 and call the resulting 3-uniform hypergraph G'. This leaves us with a random 3-uniform hypergraph on $n' = \alpha n$ vertices and with fewer than $3\alpha n' < n'/12$ hyperedges for $\alpha < 1/36$. It is well known that the threshold for such a random hypergraph to have a giant component is when the number of hyperedges is $n'/6$ and, in particular, that with probability at least γ' for some $\gamma' > 0$, every component of G' will be a tree. This would imply that every component of G is a tree. It is easy to see that if every component of G is a tree, then the formula is satisfiable.

Thus, the probability that the original formula is satisfiable is at least $\gamma' \times e^{-W/2\alpha} + o(1) > 0$, as required. □

Appendix A: A short proof of Theorem 6. Each variable v has two literals: v and $\overline{v}$, and we say that they are *complements* of each other. A literal of a variable in a CNF formula F is *pure* if does not appear in any clause of F. As explained in [2], it suffices to prove the following.

LEMMA 19. *For any $\Delta, \epsilon > 0$, consider a random CNF formula $\mathcal{F}$ on n variables with Δn 3-clauses and $(1-\epsilon)n$ 2-clauses where every such formula is equally likely. A.s.:*

(a) *every subformula on at most αn variables is satisfiable, and*

(b) *every subformula on v variables, where $\frac{1}{2}\alpha n \leq v \leq \alpha n$, has at least ζn pure literals.*

Consider any CNF formula F containing only 2-clauses and 3-clauses. Let H_2 be the graph defined as follows: The vertices of H_2 are the variables of F, and two vertices are joined iff their variables form at least one 2-clause.

An isolated cycle of F is a 2-SAT subformula F' such that (i) F' forms a component of H_2 that is a cycle, (ii) no variable of F' lies in a 3-clause, and (iii) no variable of F' has a pure literal.

A *pendant path* of F is a path of H_2 whose internal vertices each have degree 2 in H_2 and do not lie in any 3-clauses. Consider such a path whose variables are, in order, $x_0, x_1, \ldots, x_l$, and suppose that $C_1, \ldots, C_l$ is the corresponding set of 2-clauses in F. If, for each x_i, the literal of x_i appearing in C_i is the complement of the literal appearing in C_{i+1}, then we say that it is a *forcing path.* Note that such a path is bidirectional in the sense that there are two literals a and b such that, in any satisfying assignment, if $x_0 = a$, then $x_l = b$, and if $x_l = \overline{b}$, then $x_0 = \overline{a}$. Trivially, a single vertex is a forcing path of length 0.

For any $r \geq 1$, a Y_r*-configuration* consists of:

- r forcing paths and
- a collection of t_2 additional 2-clauses and t_3 3-clauses whose variables are all endpoints of the r forcing paths for some t_2, t_3, with $\frac{3}{2}t_2 + 3t_3 \geq \frac{5}{3}r$.

Consider a collection of forcing paths $\mathcal{P} = P_1, \ldots, P_r$ of F such that (i) every variable of F appears on exactly one path and (ii) $\mathcal{P}$ is minimal in the sense that it is impossible to form a collection $P_1, \ldots, P_{r-1}$ satisfying (i) by adding a 2-clause from F to $\mathcal{P}$. Obviously a collection of paths of length 0, one for each variable of F, satisfies (i), and so some collection exists which satisfies (i) and (ii).

Lemma 20. *If F has at most $r/3$ pure literals and no isolated cycles, then F has a Y_r-configuration.*

Proof. We call the clauses of $\mathcal{P}$ *path clauses* and the other clauses in F *nonpath clauses.* Note that every nonpath clause contains only variables that are endpoints of the paths in $\mathcal{P}$. We define a set X of literals as follows: For each P_i of length 0, we place both literals of the variable of P_i into X. For each P_i of length at least 1, and for each endpoint v of P_i, we place the literal of v that does not appear on a clause of P_i into X. Thus, $|X| = 2r$, and any literal of X that does not appear in a nonpath clause is pure.

We form a graph G with vertex set X as follows: Every nonpath 2-clause $(a \vee b)$ forms an edge of G: $x(a)$ is defined to be a if $a \in X$ and the complement of a otherwise; $x(b)$ is defined in the same way; the edge of G is between $x(a)$ and $x(b)$.

Let s be the number of pure literals in F, $t_2 = |E(G)|$ be the number of 2-clauses, and t_3 be the number of 3-clauses.

Claim 21. *Every component of G with either* 1 *or* 2 *literals contains a literal which is either pure or in a* 3*-clause.*

Proof: If the component has 1 literal, then it is either pure or in a 3-clause. So suppose the component has 2 literals and the edge joining them corresponds to the clause $(a \vee b)$. If at least one of a, b is not in X, then the complement of that literal is either pure or in a 3-clause. If a and b are both in X and if their variables are the endpoints of the same path in $\mathcal{P}$, then one of them must be in a 3-clause, or else that path plus $(a \vee b)$ would form an isolated cycle. If a and b are both in X and their variables are the endpoints of different paths of $\mathcal{P}$, then it is easy to see that those two paths plus $(a \vee b)$ form a forcing path. This contradicts the minimality of $\mathcal{P}$, i.e., condition (ii) in the definition of $\mathcal{P}$.

Let ℓ_1 be the number of components with exactly 1 literal and ℓ_2 be the number with exactly 2 literals. The remaining components have $2r - \ell_1 - 2\ell_2$ literals and at least $\frac{2}{3}(2r - \ell_1 - 2\ell_2)$ edges (since the components on those literals each have a size of at least 3). Therefore, $t_2 \geq \ell_2 + \frac{2}{3}(2r - \ell_1 - 2\ell_2)$; i.e. $\frac{3}{2}t_2 \geq 2r - \ell_1 - \frac{1}{2}\ell_2$. By

Claim 18, $3t_3 + s \geq \ell_1 + \ell_2$. These combine to yield:

$$\frac{3}{2}t_2 + 3t_3 + s \geq 2r.$$

Since $s \leq r/3$, F has a Y_r-configuration. □

Lemma 22. *For any $\Delta, \epsilon > 0$, consider a random CNF formula $\mathcal{F}$ on n variables with Δn 3-clauses and $(1-\epsilon)n$ 2-clauses where every such formula is equally likely. There is some constant $\alpha > 0$ such that a.s. F has no Y_r-configuration for any $r \leq \alpha n$.*

Proof. Given r, t_2 we specify $t_3 = \lceil \frac{5}{9}r - \frac{1}{2}t_2 \rceil$ to be the smallest integer t_3 such that $\frac{3}{2}t_2 + 3t_3 \geq \frac{5}{3}r$. Clearly, it suffices to show that a.s. there are no Y_r-configurations with such a pair t_2, t_3. First, we compute the expected number of Y_r-configurations for any choice of t_2, t_3 that are both at least $r/100$.

Consider any list of 2-clauses $C_1, \ldots, C_s$. The probability that they all appear in $\mathcal{F}$ is

$$\frac{\binom{4\binom{n}{2}-s}{(1-\epsilon)n-s}}{\binom{4\binom{n}{2}}{(1-\epsilon)n}} < \left(\frac{1-\epsilon}{2(n-1)}\right)^s < \left(\frac{1-\epsilon'}{2n}\right)^s$$

for some $0 < \epsilon' < \epsilon$.

We have at most $\binom{n}{r}n^r$ choices for the r pairs of endpoints. Suppose that the numbers of 2-clauses in the paths are $l_1, \ldots, l_r$, and set $L = l_1 + \cdots + l_r$. Then there are n^{L-r} choices for the interior variables on the paths and 2^{L+r} choices for the literals. We multiply by the probability that all L of these clauses appear and that there are t_2 other 2-clauses and t_3 3-clauses on the endpoints. This gives us an upper bound of

$$\begin{aligned}
&\sum_{l_1,\ldots,l_r \geq 0} \binom{n}{r} n^L 2^{L+r} \left(\frac{1-\epsilon'}{2n}\right)^L \binom{(1-\epsilon)n}{t_2}\binom{\Delta n}{t_3}\left(\frac{2r}{n}\right)^{2t_2+3t_3} \\
&\leq \left(\frac{2en}{r}\right)^r \left(\frac{en}{t_2}\right)^{t_2} \left(\frac{e\Delta n}{t_3}\right)^{t_3} \left(\frac{2r}{n}\right)^{2t_2+3t_3} \sum_{l_1,\ldots,l_r \geq 0} (1-\epsilon')^L \\
&\leq \left(\frac{\gamma r}{n}\right)^{t_2+2t_3-r} \left(\sum_{l \geq 0}(1-\epsilon')^l\right)^r \qquad \text{for some } \gamma > 0, \text{ since } t_2, t_3 \geq r/100 \\
&\leq \left(\frac{\gamma' r}{n}\right)^{r/9}.
\end{aligned}$$

For some constant $\gamma' > 0$ since $t_2 + 2t_3 - r = \frac{2}{3}(\frac{3}{2}t_2 + 3t_3) - r \geq \frac{2}{3} \times \frac{5}{3}r - r = \frac{r}{9}$.

If $t_2 \leq r/100$, then $t_3 \geq (\frac{5}{9} - \frac{1}{200})r$. For such t_2, we compute the expected number of collections of r vertex-disjoint forcing paths along with t_3 3-clauses on their endpoints. Clearly, if there are no such collections, then there is no Y_r-configuration with those values of t_2, t_3. As above, we upper bound this expected number with

$$\left(\frac{2en}{r}\right)^r \left(\frac{e\Delta n}{t_3}\right)^{t_3} \left(\frac{2r}{n}\right)^{3t_3} \left(\sum_{l \geq 0}(1-\epsilon')^l\right)^r < \left(\frac{\gamma' r}{n}\right)^{r/10},$$

with possibly an increase in γ'. If $t_3 \leq r/100$, then we compute the expected number of collections of r vertex-disjoint forcing paths along with t_2 additional 2-clauses on their endpoints. Clearly, if there are no such collections, then there is no Y_r-configuration with those values of t_2, t_3. Again, this expected number is at most $(\frac{\gamma' r}{n})^{r/10}$. Thus, considering that there are $O(r)$ choices for t_2, t_3, it suffices to show that

$$\sum_{r=1}^{\alpha n} r \left(\frac{\gamma' r}{n}\right)^{r/10} = o(1).$$

The first $\log n$ terms of this sum add up to at most $O(\log n/n^{1/10})$, and if $\alpha < \frac{1}{2\gamma'}$, then the rest add up to at most $\sum_{i \geq \log n} (1/2)^i = o(1)$. □

Lemma 23. *A.s. our random $\mathcal{F}$ has at most* $\log n$ *variables lying on isolated cycles.*

Proof. Consider the subformula F_2 formed by the 2-clauses of $\mathcal{F}$. For any variable v lying in an isolated cycle of $\mathcal{F}$, there must be a sequence of 2-clauses in F_2 of the form: $(v \vee x_1), (\overline{x_1} \vee x_2), \ldots, (\overline{x_i} \vee \overline{v})$. A well-known property of random 2-SAT (see, e.g., [11]) says that a.s. there are at most $\log n$ such variables. □

Proof of Lemma 19. If F' is a minimally unsatisfiable subformula of $\mathcal{F}$, then F' must be connected and F' cannot be an isolated cycle. Therefore F' can have no isolated cycles. Furthermore F' can have no pure literals. Therefore, by Lemma 20, F' must have a Y_r-configuration for some $r \geq 1$. Therefore, by Lemma 22, $\mathcal{F}$ a.s. has no minimally unsatisfiable subformula on at most αn variables, and hence a.s. has no unsatisfiable subformula on at most αn variables. This establishes part (a).

Consider any subformula on v variables, where $\frac{1}{2}\alpha n \leq v \leq \alpha n$. By Lemma 23, $\mathcal{F}$ is a.s. such that, after removing all isolated cycles from such a subformula, we are left with a subformula F' on v' variables, where $\frac{1}{2}\alpha n - \log n \leq v' \leq \alpha n$. It will suffice to show that a.s. every such F' has at least ζn pure literals. By Lemma 22, $\mathcal{F}$ is a.s. such that every subformula on at most αn variables does not have a Y_r-configuration for any $r \geq 1$. Therefore, by Lemma 20, there is some $r \geq 1$ such that F' has at least $r/3$ pure literals and F' has a collection $\mathcal{P}$ of r forcer paths which contain all of its variables.

Well-known properties of random 2-SAT (see, e.g., [11]) imply that there is some $\pi > 0$ such that for every $\theta > 0$ a.s. F has fewer than $e^{-\pi\theta}$ variables that lie on forcing paths of length at least θ. In fact, the same would be true if we removed from the definition of *forcing path* the stipulation that no internal variables lie in any 3-clauses, thus reducing forcing paths to be defined only in terms of the $(1-\epsilon)n$ random 2-clauses of F. Pick θ so that $e^{-\pi\theta} < \alpha/4$. Thus, at least $(\alpha/4)n - \log n$ variables of F' lie on paths in $\mathcal{P}$ of length less than θ. Therefore, $r > \alpha n/(5\theta)$, and so F' has at least ζn pure literals for $\zeta = \alpha/(15\theta)$. This establishes part (b). □

Appendix B: Two simple lemmas. Here we translate two standard facts from random graph theory into the setting of this paper.

Lemma 24. *A.s. the underlying random hypergraph of $\mathcal{F}_{n,M}^{d,k,t}$ has fewer than* $\log n$ *cycles of length at most* 4.

Remark. This statement remains true when "4" is replaced by any constant. It is well known for random graphs, but we can't find the statement for random hypergraphs recorded in the literature. So we include the simple proof here.

Proof. For each constant integer $r \geq 2$, we compute the expected number of cycles of length r, by pretending that the vertices of degree 2 in the cycle are labeled $v_1 \ldots, v_r$, in order around the cycle; this creates an overcount, which we correct by dividing by $2r$, the number of ways to label each cycle. (For $k = 2$, we take $r \geq 3$ since a simple graph contains no 2-cycles; for $r = 2$, we must divide by r instead of $2r$ since there are only 2 ways to label each cycle.)

There are at most n^r choices for $v_1, \dots, v_r$ and at most $\binom{n-r}{k-2}^r$ choices for the other vertices. This specifies the r edges. There are $M^r = (cn)^r$ choices for which random edges correspond to the edges of the cycle. The probability that each such random edge is the desired one is $\binom{n}{k}^{-r}$. So the expected number of cycles of length r is

$$\frac{1}{2r} n^r \binom{n-r}{k-2}^r (cn)^r \binom{n}{k}^{-r} = O(1).$$

(For $r = 2$ we replace $\frac{1}{2r}$ by $\frac{1}{r}$.) Thus, the expected number of cycles of length $2 \le r \le 4$ is $O(1)$. So by Markov's inequality, the probability that this number is at least $\log n$ is $O(1/\log n) = o(1)$ as required. □

Remark. With more work, one can show that the probability is much lower than $O(1/\log n)$; but that is not needed here.

Our second lemma shows how the $\mathcal{F}_{n,M}^{d,k,t}$ and $\mathcal{F}_{n,p}^{d,k,t}$ models are, in many senses, equivalent and, in particular, allows us to use the $\mathcal{F}_{n,p}^{d,k,t}$ model in the proofs of Lemmas 15 and 16.

We say that a property A of CSPs in $\Omega^{d,k,t}$ is *monotone increasing* if for every $F_1, F_2 \in \Omega^{d,k,t}$ with every constraint of F_1 also in F_2, if F_1 has A, then so does F_2. A is *monotone decreasing* if the same holds whenever every constraint of F_2 is also in F_1. A is *monotone* if it is either monotone increasing or monotone decreasing.

For example, it is easy to see that the properties considered in the statements of Lemmas 15 and 16 are both monotone decreasing.

Lemma 25. *Let A be any monotone property of CSPs in $\Omega^{d,k,t}$, and let $c > 0$ be any positive constant. A holds a.s. for $\mathcal{F}_{n,p}^{d,k,t}$, with $p = c \times k!/n^{k-1}$ iff for every real constant x, A holds a.s. for $\mathcal{F}_{n,M}^{d,k,t}$, with $M = \lceil cn + x\sqrt{n} \rceil$.*

In particular, taking $x = 0$ allows us to show that A holds a.s. in $\mathcal{F}_{n,M}^{d,k,t}$ by proving that it holds a.s. in $\mathcal{F}_{n,p}^{d,k,t}$, as we do in the proofs of Lemmas 15 and 16. The proof is very straightforward and follows similar proofs in, e.g., [10, 28].

Proof. We assume that A is monotone increasing (the monotone decreasing case is nearly identical).

Suppose that $\mathcal{F}_{n,p}^{d,k,t}$ a.s. has A. Fix any real x, and set $M = \lceil cn + x\sqrt{n} \rceil$. Let $\gamma(n) = \mathbf{Pr}(\mathcal{F}_{n,M}^{d,k,t}$ does not have $A)$. The probability that the number of edges in $\mathcal{F}_{n,p}^{d,k,t}$ is at most M is well known to be at least a positive constant $g(x)$, since this number is a binomial variable with mean cn. By the monotonicity of A, the probability that $\mathcal{F}_{n,p}^{d,k,t}$ does not have A is at least $g(x) \times \gamma(n)$. Therefore, $\lim_{n\to\infty} \gamma(n) = 0$, as required.

For the other direction, suppose that, for every real constant x, A holds a.s. for $\mathcal{F}_{n,M}^{d,k,t}$, with $M = \lceil cn + x\sqrt{n} \rceil$. For any $\epsilon > 0$, there exists $x_1 < 0 < x_2$ such that the probability that the number of edges in $\mathcal{F}_{n,p}^{d,k,t}$ is in $\lceil cn + x_1\sqrt{n} \rceil, \dots, \lceil cn + x_2\sqrt{n} \rceil$ is at least $1 - \epsilon$. Therefore, the probability that $\mathcal{F}_{n,p}^{d,k,t}$ does not have A is at most $\epsilon + o(1)$. Since this is true for every $\epsilon > 0$, $\mathcal{F}_{n,p}^{d,k,t}$ a.s. has A. (Note that this part did not require A to be monotonic.) □

Acknowledgments. We thank Joe Culberson and Yong Gao for spotting an error in Theorem 4 of the conference version of this paper [34]. We also thank two anonymous referees for a very careful reading and many suggested improvements in the presentation. Most of this research was completed while the first author was a visiting researcher at Microsoft Research and the second author was with Department of Computer Science, University of Toronto.

REFERENCES

[1] D. Achlioptas, *Lower bounds for random 3-SAT via differential equations*, Theoret. Comput. Sci., 265 (2001), pp. 159–185.

[2] D. Achlioptas, P. Beame, and M. Molloy, *A sharp threshold in proof complexity*, J. Comput. System Sci., 68 (2004), pp. 238–268.

[3] D. Achlioptas, A. Chtcherba, G. Istrate, and C. Moore, *The phase transition in NAE-SAT and 1-in-k SAT*, in Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 01), pp. 721–722.

[4] D. Achlioptas, L. Kirousis, E. Kranakis, and D. Krizanc, *Rigorous results for random $(2+p)$-SAT*, Theoret. Comput. Sci., 265 (2001), pp. 109–129.

[5] D. Achlioptas, L. Kirousis, E. Kranakis, D. Krizanc, M. Molloy, and Y. Stamatiou, *Random constraint satisfaction: A more accurate picture*, Constraints, 6 (2001), pp. 329–324.

[6] P. Beame, J. Culberson, and D. Mitchell, *The resolution complexity of random graph k-colorability*, Discrete Appl. Math., 153 (2005), pp. 25–47.

[7] P. Beame and T. Pitassi, *Simplified and improved resolution lower bounds*, in Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS 1996), IEEE, New York, pp. 274–282.

[8] P. Beame, R. Karp, T. Pitassi, and M. Saks, *The efficiency of resolution and Davis-Putnam procedures*, SIAM J. Comput., 31 (2002), pp. 1048–1075.

[9] E. Ben-Sasson and A. Wigderson, *Short proofs are narrow - resolution made simple*, J. ACM, 48 (2001), pp. 149–169.

[10] B. Bollobás, *Random Graphs*, Academic, London, 1985.

[11] B. Bollobás, C. Borgs, J. T. Chayes, J. H. Kim, and D. B. Wilson, *The scaling window of the 2-SAT transition*, Random Structures Algorithms, 18 (2001), pp. 201–256.

[12] V. Chvátal and B. Reed, *Mick gets some (the odds are on his side)*, in Proceedings of the 33rd Annual Symposium on Foundations of Computer Science, Pittsburgh, PA, IEEE, New York, 1992, pp. 620–627.

[13] V. Chvátal and E. Szemerédi, *Many hard examples for resolution*, J. ACM, 35 (1988), pp. 759–768.

[14] N. Creignou and H. Daude, *Satisfiability threshold for random XOR-CNF formulas*, Discrete Appl. Math., 96-97 (1999), pp. 41–53.

[15] N. Creignou and H. Daudé, *Generalized satisfiability problems: Minimal elements and phase transitions*, Theoret. Comput. Sci., 1-3 (2003), pp. 417–430.

[16] N. Creignou and H. Daudé, *Combinatorial sharpness criterion and phase transition classification for random CSPs*, Inform. and Comput., 190 (2004), pp. 220–238.

[17] N. Creignou and H. Daudé, *Coarse and sharp transitions for random generalized satisfiability problems*, in Mathematics and Computer Science III: Algorithms, Trees, Combinatorics and Probabilities, in Proceedings of the Third Colloquium on Mathematics and Computer Science, Vienna, 2004, M. Drmota, P. Flajolet, D. Gardy, and B. Gittenberger, eds., Birkhauser-Springer, 2004, pp. 507–516.

[18] N. Creignou, H. Daude, and O. Dubois, *Approximating the satisfiability threshold of random k-XOR-formulas*, Combin. Probab. Comput., 12 (2003), pp. 113–126.

[19] O. Dubois and J. Mandler, *The 3-XORSAT threshold (strong evidence in favour of the replica method of statistical physics)*, in Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2002).

[20] M. Dyer, A. Frieze, and M. Molloy, *A probabilistic analysis of randomly generated binary constraint satisfaction problems*, Theoret. Comput. Sci., 290 (2003), pp. 1815–1828.

[21] W. Fernandex de la Vega, *On Random 2-SAT*, manuscript, 1992.

[22] E. Friedgut and an appendix by J. Bourgain, *Sharp thresholds of graph properties and the k-SAT problem*, J. Amer. Math. Soc., 12 (1999), pp. 1017–1054.

[23] Y. Gao and J. Culberson, *Resolution complexity of random constraint satisfaction problems: Another half of the story*, Discrete Appl. Math., 153 (2005), pp. 124–140.

[24] I. Gent, E. MacIntyre, P. Prosser, B. Smith, and T. Walsh, *Random constraint satisfaction: Flaws and structure*, Constraints, 6 (2001), pp. 345–372.

[25] A. Goerdt, *A threshold for unsatisfiability*, J. Comput. System. Sci., 53 (1996), pp. 469–486.

[26] A. Haken, *The intractability of resolution*, Theoret. Comput. Sci., 39 (1985), pp. 297–305.

[27] G. Istrate, *Threshold properties of random Boolean constraint satisfaction problems*, Discrete Appl. Math., 153 (2005), pp. 141–152.

[28] S. Janson, T. Łuczak, and A. Ruciński, *Random Graphs*, Wiley, New York, 2000.

[29] L. Kirousis, E. Kranakis, D. Krizanc, and Y. Stamatiou, *Approximating the unsatisfiability threshold of random formulas*, Random Structures Algorithms, 12 (1998), pp. 253–269.

[30] T. Łuczak, *Size and connectivity of the k-core of a random graph*, Discrete Math., 91 (1991), pp. 61–68.

[31] D. Mitchell, *Resolution complexity of random constraints*, in Proceedings of the Eighth International Conference on Principles and Practices of Constraint Programming, Ithaca, New York, Lect. Notes Comput. Sci. 2470, Springer, New York, 2002.

[32] D. Mitchell, *The resolution complexity of constraint satisfaction*, Ph.D. Thesis, University of Toronto, 2002.

[33] M. Molloy, *Models for random constraint satisfaction problems*, SIAM J. Comput., 32 (2003), pp. 935–949.

[34] M. Molloy and M. Salavatipour, *The resolution complexity of random constraint satisfaction problems*, in Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2003).

[35] R. Monasson and R. Zecchina, *Tricritical point in the random 2+p-SAT problem*, J. Phys. A, 31 (1998), p. 9209.

[36] N. Wormald, *Differential equations for random processes and random graphs*, Ann. Appl. Probab., 5 (1995), pp. 1217–1235.

[37] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky, *2+p-SAT: Relation of typical-case complexity to the nature of the phase transition*, Random Structure Algorithms, 15 (1999), p. 414.

[38] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky, *Phase transition and search cost in the 2+p-sat problem*, in Proceedings of the 4th Workshop on Physics and Computation (PhysComp 96), Boston, 1996, T. Toffoli, M. Biafore, J. Leao, eds., New England Complex Systems Institute, Cambridge, MA, 1996.

[39] A. Urquhart, *Hard examples for resolution*, J. ACM, 34 (1987), pp. 209–219.

[40] K. Xu and W. Li, *Exact phase transitions in random constraint satisfaction problems*, J. Artificial Intelligence Res., 12 (2000), pp. 93–103.

SIAM J. COMPUT.
Vol. 37, No. 3, pp. 923–937

A MIN-MAX THEOREM ON TOURNAMENTS*

XUJIN CHEN†, XIAODONG HU†, AND WENAN ZANG‡

Abstract. We present a structural characterization of all tournaments $T = (V, A)$ such that, for any nonnegative integral weight function defined on V, the maximum size of a feedback vertex set packing is equal to the minimum weight of a triangle in T. We also answer a question of Frank by showing that it is NP-complete to decide whether the vertex set of a given tournament can be partitioned into two feedback vertex sets. In addition, we give exact and approximation algorithms for the feedback vertex set packing problem on tournaments.

Key words. min-max relation, feedback vertex set, tournament, packing, covering

AMS subject classifications. 90C10, 90C27, 90C57

DOI. 10.1137/060649987

1. Introduction.

A rich variety of combinatorial optimization problems falls within the general framework of packing and covering in hypergraphs. A *hypergraph* is a pair $\mathcal{H} = (V, \mathcal{E})$, where V is a finite set and $\mathcal{E}$ is a family of subsets of V. Elements of V and $\mathcal{E}$ are called the *vertices* and *edges* of $\mathcal{H}$, respectively. A *vertex cover* of $\mathcal{H}$ is a vertex subset that intersects all edges of $\mathcal{H}$. Let w be a nonnegative integral weight function defined on V. A family $\mathcal{S}$ of edges (repetition is allowed) of $\mathcal{H}$ is called a *w-packing* of $\mathcal{H}$ if each $v \in V$ belongs to at most $w(v)$ members of $\mathcal{S}$. Let $\nu_w(\mathcal{H})$ denote the maximum size of a w-packing of $\mathcal{H}$, and let $\tau_w(\mathcal{H})$ denote the minimum total weight of a vertex cover. Clearly $\nu_w(\mathcal{H}) \leq \tau_w(\mathcal{H})$; this inequality, however, need not hold equality in general. We say that $\mathcal{H}$ is *Mengerian* if the min-max relation $\nu_w(\mathcal{H}) = \tau_w(\mathcal{H})$ is satisfied for any nonnegative integral function w defined on V. Many celebrated results and conjectures in combinatorial optimization can be rephrased by saying that certain hypergraphs are Mengerian (see section 79.1 of [19]), so Mengerian hypergraphs have been subjects of extensive research. As conjectured by Edmonds and Giles [9, 18] and proved recently by Ding, Feng, and Zang [4], the problem of recognizing Mengerian hypergraphs is NP-hard in general, and hence it cannot be solved in polynomial time unless $NP = P$. In this paper we study a special class of Mengerian hypergraphs; our work is a continuation of those done in [1, 2, 3, 5, 6].

Let $G = (V, E)$ be a graph (directed or undirected), and let $\mathcal{C}_G = (V, \mathcal{E})$, where $\mathcal{E}$ consists of $V(C)$, for all induced cycles C in G. Throughout this paper, by a cycle in a digraph we always mean a directed one. In [6], Ding and Zang obtained a structural description of all undirected graphs G for which $\mathcal{C}_G$ is Mengerian. Due to the long list of forbidden structures, to find a good characterization of all digraphs G with Mengerian $\mathcal{C}_G$ seems to be extremely difficult. While this characterization problem

*Received by the editors January 15, 2006; accepted for publication (in revised form) May 8, 2007; published electronically August 22, 2007.

http://www.siam.org/journals/sicomp/37-3/64998.html

†Institute of Applied Mathematics, Chinese Academy of Sciences, Beijing 100080, China (xchen@amss.ac.cn, xdhu@amss.ac.cn). The work of the first author was supported in part by National Postdoctoral Foundation of China. The work of the second author was supported in part by the NSF of China under grant 70221001.

‡Corresponding author. Department of Mathematics, The University of Hong Kong, Hong Kong, China (wzang@maths.hku.hk). This author's work was supported in part by the Research Grants Council of Hong Kong.

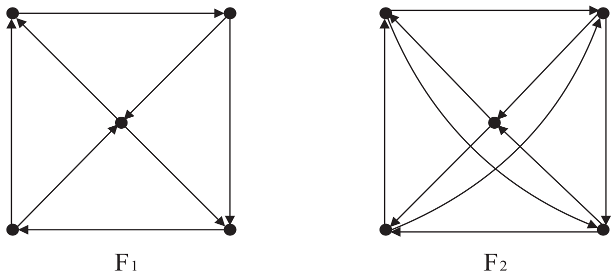


Fig. 1. *Forbidden subtournaments F_1 and F_2, where the two arcs not shown in F_1 may take any directions.*

remains open in general, it was completely solved on tournaments by Cai, Deng, and Zang [1], where a *tournament* is an orientation of an undirected complete graph.

Theorem 1.1 ([1]). *Let T be a tournament. Then hypergraph $\mathcal{C}_T$ is Mengerian if and only if T has no subtournament isomorphic to F_1 nor F_2.*

(Note that F_2 is the tournament in which every vertex is incident with precisely two incoming arcs and two outgoing arcs.) One objective of this paper is to establish a closely related min-max relation which is motivated as follows.

Every hypergraph $\mathcal{H} = (V, \mathcal{E})$ is naturally associated with another hypergraph $b(\mathcal{H}) = (V, \mathcal{E}')$, where $\mathcal{E}'$ consists of all minimal (with respect to set inclusion) vertex covers of $\mathcal{H}$. Usually $b(\mathcal{H})$ is called the *blocker* of $\mathcal{H}$. Although in general the blocker of a Mengerian hypergraph does not have to be Mengerian (see section 79.2 of [19]), the famous max-flow-min-cut theorem and a Fulkerson theorem [11] (see p. 115 of [18]) assert that both the hypergraph of r-s paths in a graph and its blocker are Mengerian; so are the hypergraph of r-arborescences and its blocker by Edmonds' disjoint arborescence theorem [7] and Fulkerson's optimum arborescence theorem [12]. Recently, Chen et al. [3] managed to characterize all undirected graphs G for which $b(\mathcal{C}_G)$ is Mengerian; it turns out that $b(\mathcal{C}_G)$ is Mengerian if and only if $\mathcal{C}_G$ is. So a natural question is to ask: What is the blocker version of Theorem 1.1?

Theorem 1.2. *Let T be a tournament. Then hypergraph $b(\mathcal{C}_T)$ is Mengerian if and only if T has no subtournament isomorphic to F_1 nor F_2 (see Figure 1).*

An immediate corollary of Theorems 1.1 and 1.2 is the following.

Corollary 1.3. *Let T be a tournament. Then $b(\mathcal{C}_T)$ is Mengerian if and only if $\mathcal{C}_T$ is.*

Let us define a few terms before presenting an equivalent of the above statements. Let $G = (V, E)$ be a digraph with a nonnegative integral weight $w(v)$ on each vertex v. A *feedback vertex set* (FVS) of G is a vertex subset that intersects each cycle in G, and a *w-FVS packing* of G is a collection $\mathcal{F}$ of minimal FVSs (repetition is allowed) such that each vertex v is contained in at most $w(v)$ members of $\mathcal{F}$. Similarly, a *w-cycle packing* of G is a collection $\mathcal{C}$ of induced cycles (repetition is allowed) such that each vertex v is contained in at most $w(v)$ members of $\mathcal{C}$. The *weight* of a cycle (resp., an FVS) is the sum of weights of all vertices in this cycle (resp., FVS). Observe that every minimal FVS of G uniquely corresponds to an edge of $b(\mathcal{C}_G)$, and vice versa. So there is $1-1$ correspondence between a *w-FVS packing* of G and a w-packing of $b(\mathcal{C}_G)$ and $1-1$ correspondence between a *w-cycle packing* of G and a w-packing of $\mathcal{C}_G$. Moreover, if G is a tournament, then every cycle in a cycle packing is a triangle (a cycle of length three), and hence a cycle packing is actually a triangle packing.

Let $\boldsymbol{Z}_+$ denote the set of nonnegative integers. Then Theorems 1.1 and 1.2 can be restated as follows.

THEOREM 1.4. *The following three statements are equivalent for a tournament* $T = (V, A)$*:*

(i) *For any weight function* $w \in \mathbf{Z}_+^V$*, the minimum weight of an FVS in* T *is equal to the maximum size of a* w*-triangle packing of* T*;*

(ii) *for any weight function* $w \in \mathbf{Z}_+^V$*, the minimum weight of a triangle in* T *is equal to the maximum size of a* w*-FVS packing of* T*;*

(iii) T *has no subtournament isomorphic to* F_1 *nor* F_2*.*

It is worthwhile pointing out that the above statement (i) is closely related to the famous Lucchesi–Younger theorem [15], which, when restricted to a planar digraph $G = (V, E)$, is equivalent to saying that for any $w \in \mathbf{Z}_+^E$ the minimum weight of a feedback arc set in G is equal to the maximum size of a cycle packing of G, where a *feedback arc set* of G is a set of arcs that intersects each cycle in G; statement (ii) is closely related to the well-known Woodall conjecture [20] on packing feedback arc sets and the Edmonds–Giles conjecture [8, 17] on packing directed cut covers.

Given a digraph $G = (V, E)$ with a nonnegative integral weight $w(v)$ on each vertex v, the *FVS packing problem* is to find a w-FVS packing of maximum size in G. In connection with this problem, Frank suggested the following question.

Question 1.5 ([10]). Given a digraph G, can we decide in polynomial time whether each vertex of G can be colored by red or blue so that every cycle contains at least one red vertex and at least one blue vertex? Or is this an NP-complete problem?

Our next theorem states that Frank's problem is NP-complete even when G is restricted to a tournament.

THEOREM 1.6. *It is NP-complete to decide whether the vertex set of a given tournament can be partitioned into two feedback vertex sets.*

We shall also present algorithms for the FVS packing problem.

THEOREM 1.7. *The FVS packing problem on a tournament* $T = (V, A)$ *with no* F_1 *nor* F_2 *can be solved exactly in* $O(|V|^4)$ *time.*

For the problem on a general tournament, we shall give an approximation algorithm.

THEOREM 1.8. *The FVS packing problem on a general tournament can be approximated within a factor of* $2/5$.

The remainder of this paper is organized as follows: In section 2, we give a proof of Theorem 1.2, which relies heavily on the structural description of tournaments with no F_1 nor F_2 obtained in [1]. In section 3, we prove Theorem 1.6 by using the so-called NOT-ALL-EQUAL 3-SATISFIABILITY problem as the source problem. In section 4, we present an exact algorithm for the FVS packing problem on tournaments with no F_1 nor F_2 and describe a 2/5-approximation algorithm for the problem on general tournaments. In section 5, we conclude this paper with some open problems.

2. Min-max relation.

The purpose of this section is to prove Theorem 1.2. We break the proof into a series of lemmas and shall implicitly and frequently use the fact that a vertex subset of a tournament is an FVS if and only if it intersects every triangle. As usual, a digraph G is called *strongly connected* if, for any two vertices x and y, there exist a (directed) path from x to y and a (directed) path from y to x in G. Our proof relies heavily on the following structural description obtained in [1].

LEMMA 2.1 ([1]). *Let* $T = (V, A)$ *be a strongly connected tournament. Then* T *has no subtournament isomorphic to* F_1 *nor* F_2 *if and only if* V *can be partitioned into* $V_1, V_2, \ldots, V_k$ *for some* $3 \le k \le |V|$*, which have the following properties:*

(i) *For any* i, j *with* $1 \le i \le j-2 \le k-2$*, each arc between* V_i *and* V_j *is directed from* V_j *to* V_i*.*

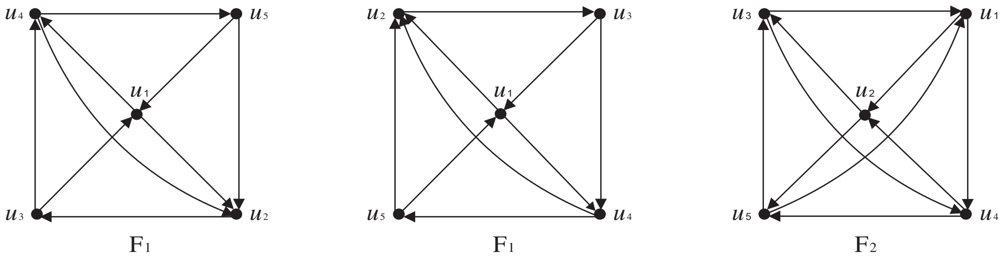


FIG. 2. *Three triangles $u_1u_2u_3u_1$, $u_2u_3u_4u_2$, and $u_1u_4u_5u_1$ in F_1 and F_2.*

(ii) *For any triangle $xyzx$ in T, there exists an i with $1 \le i \le k-2$ such that $x \in V_i$, $y \in V_{i+1}$, and $z \in V_{i+2}$ (renaming x, y, and z if necessary).*

We make two remarks on the above lemma: First, for notational convenience, the order of the indices used in the above partition $V_1, V_2, \ldots, V_k$ is precisely the reverse of the one used in [1]. Second, as depicted in Figure 2, the vertices of both F_1 and F_2 can be labeled as $u_1, u_2, \ldots, u_5$ such that $\{u_1, u_2, u_3\}$, $\{u_2, u_3, u_4\}$, and $\{u_1, u_4, u_5\}$ are vertex sets of three triangles. Using these triangles, we can immediately see the sufficiency.

Let $T = (V, A)$ be a tournament and $u, v \in V$. The arc in T with tail u and head v is written as (u, v) and called the arc from u to v. For any subtournament K of T, let $V(K)$ and $A(K)$ denote the vertex set and arc set of K, respectively. For any vertex u of T, let $T\backslash u$ denote the tournament obtained from T by deleting u, and let $T\langle u\rangle$ denote the tournament obtained from T by introducing a new vertex u' and then adding arcs in such a way that

$$(1^*)\qquad \text{for each } v \in V - \{u\},\ (u', v) \text{ is an arc in } T\langle u\rangle \text{ if and only if } (u, v) \text{ is an arc in } T.$$

(There is no direction constraint on the arc between u and u'.) We propose to call u' the *image* of u and call $T\langle u\rangle$ an *augmentation of* T (*with respect to* u). It can be seen from (1^*) that

$$(2^*)\qquad \text{no triangle in } T\langle u\rangle \text{ contains } \{u, u'\}.$$

LEMMA 2.2. *Let $T\langle u\rangle$ be an augmentation of a tournament $T = (V, A)$. If T contains no F_1 nor F_2, then neither does $T\langle u\rangle$.*

Proof. Assume the contrary: $T\langle u\rangle$ contains a subtournament F isomorphic to F_1 or F_2. Let u' be the image of u. Then F contains both u and u', for otherwise, by (1^*), $V(F\backslash u') \cup \{u\}$ would induce a subtournament in T isomorphic to F, which is a contradiction.

(1) We may assume that T is strongly connected.

Suppose not, let K be the strongly connected component of $T\langle u\rangle$ that contains F (such K is available since F is strongly connected). Then $K\backslash u'$ is strongly connected, for otherwise the vertex set of $K\backslash u'$ can be partitioned into X and Y such that all arcs between X and Y are directed to Y. Without loss of generality, we assume that $u \in X$. Since u' is the image of u, all arcs in K between $X \cup \{u'\}$ and Y are directed to Y, contradicting the strong connectivity of K. Since K is an augmentation of $K\backslash u'$ (with respect to u), we get (1); otherwise, replace T by $K\backslash u'$ and $T\langle u\rangle$ by K.

It follows from (1) that the vertex set V of T admits a partition $V_1, V_2, \ldots, V_k$ with properties (i) and (ii) as described in Lemma 2.1. Suppose $u \in V_h$. Let us partition the vertex set $V \cup \{u'\}$ of $T\langle u\rangle$ into k sets V'_i such that $V'_h = V_h \cup \{u'\}$ and $V'_j = V_j$ for all other j with $1 \le j \le k$. From (1*), we see that the partition $V'_1, V'_2, \dots, V'_k$ satisfies (i) in Lemma 2.1 with respect to $T\langle u\rangle$. Since F is contained in $T\langle u\rangle$, Lemma 2.1 guarantees the existence of a triangle $xyzx$ in $T\langle u\rangle$ that violates (ii) in the lemma with respect to the partition $V'_1, V'_2, \dots, V'_k$. Note that $\{x, y, z\}$ contains at most one of u and u' by (2*). Set $Q = \{x, y, z\}$ if $u' \notin \{x, y, z\}$ and $Q = (\{x, y, z\} - \{u'\}) \cup \{u\}$ otherwise. Then Q would induce a triangle in T that violates Lemma 2.1(ii) with respect to the partition $V_1, V_2, \dots, V_k$, which is a contradiction. □

Let $T = (V, A)$ be a tournament, and let $S \subseteq V$. We shall use the following notations in our proof:

$$(3^*) \qquad \mathscr{D}_S := \{C : C \text{is a triangle in } T \text{ and } |V(C) \cap S| = 2\},$$

$$(4^*) \qquad \mathscr{F}_S := \{C : C \text{ is a triangle in } T,\ V(C) \subseteq S, \text{ and } |V(C) \cap V(C')| \le 1 \text{ for every } C' \in \mathscr{D}_S\},$$

$$(5^*) \qquad \mathscr{F}^+_S := \{C : C \text{ is a triangle in } T,\ V(C) \subseteq S, \text{ and } |V(C) \cap V(C')| = 2 \text{ for some } C' \in \mathscr{D}_S\}.$$

Let $\mathscr{C}$ be a collection of some triangles in T. Write $V(\mathscr{C}) = \cup_{C \in \mathscr{C}} V(C)$. It follows from the definition that $V(\mathscr{D}_S) - S \ne \emptyset$ if $\mathscr{D}_S \ne \emptyset$ and that $V(\mathscr{F}_S) \cup V(\mathscr{F}^+_S) \subseteq S$.

LEMMA 2.3. *Let $T = (V, A)$ be a tournament with no subtournament isomorphic to F_1 nor F_2. Suppose S is a subset of V such that $\mathscr{D}_S \ne \emptyset$ and that $|S \cap V(C)| \ge 2$ for every triangle C of T. Then there exists $R \subseteq S$ such that $|R \cap V(C)| = 1$ for every triangle $C \in \mathscr{D}_S$. Moreover, given S, such an R can be found in $O(|V|^3)$ time.*

Proof. Let us first construct an undirected graph G with vertex set S as follows: uv is an edge of G if and only if there is a triangle C in T such that $\{u, v\} = S \cap V(C)$. If G is a bipartite graph, let R be one color class of G, and then R is as desired. So we assume that G is nonbipartite and aim to reach a contradiction. To this end, let $x_1 x_2 \dots x_{2l+1} x_1$ be the shortest odd cycle of G. From the construction of G, we see that, for every i with $1 \le i \le 2l + 1$, there exists a vertex y_i in $V - S$ such that $\{x_i, x_{i+1}, y_i\}$ induces a triangle, denoted by $\triangle_i$, in $\mathscr{D}_S$. Note that y_i's may not be distinct.

Let T_0 denote the subtournament of T induced by vertex subset $\{x_i, y_i : 1 \le i \le 2l + 1\}$. Then

(1) $\triangle_i$, for $i = 1, 2, \dots, 2l + 1$, are $2l + 1$ triangles in T_0, where $x_{2l+2} = x_1$.

Let us perform a sequence of $2l + 1$ augmentations in the following iterative way: $T_i := T_{i-1}\langle y_i\rangle$; that is, T_i is an augmentation of T_{i-1} with respect to y_i, for $i = 1, 2, \dots, 2l+1$. Let y'_i be the image of y_i involved in the construction of T_i, and let C_i denote the triangle $y'_i x_i x_{i+1} y'_i$ if $\triangle_i = y_i x_i x_{i+1} y_i$ and $y'_i x_{i+1} x_i y'_i$ otherwise. Since $\{x_1, x_2, \dots, x_{2l+1}\} \cap \{y_1, y_2, \dots, y_{2l+1}\} = \emptyset = \{y'_1, y'_2, \dots, y'_{2l+1}\} \cap \{y_1, y_2, \dots, y_{2l+1}\}$, and since $x_1, x_2, \dots, x_{2l+1}, y'_1, y'_2, \dots, y'_{2l+1}$ are distinct vertices, by (1) we have

(2) C_i, for $i = 1, 2, \dots, 2l + 1$, are $2l + 1$ triangles in T_{2l+1}, with the property that no vertex of T_{2l+1} is contained in more than two of them.

Since T_0 is a subtournament of T, it contains no F_1 nor F_2. Repeated applications of Lemma 2.2 yield the following:

(3) For any $0 \le i \le 2l + 1$, tournament T_i contains no F_1 nor F_2.

Let us make one more simple observation.

(4) For any $0 \le i \le 2l + 1$, every triangle in T_i contains at least two vertices from $\{x_1, x_2, \dots, x_{2l+1}\}$.

To justify (4), we apply induction on i. For $i = 0$, since $S \cap V(T_0) = \{x_1, x_2, \dots, x_{2l+1}\}$ and $|S \cap V(C)| \ge 2$ for every triangle C of T_0 (by hypothesis), the desired statement follows. Suppose we have established the assertion for T_{i-1}. Let us proceed to the induction step for T_i. Let $xyzx$ be an arbitrary triangle in T_i. Set $Q = \{x, y, z\}$ if $y'_i \notin \{x, y, z\}$ and $Q = (\{x, y, z\} - \{y'_i\}) \cup \{y_i\}$ otherwise. It follows from (1*) and (2*) that Q induces a triangle in T_{i-1}. So it contains at least two vertices from $\{x_1, x_2, \ldots, x_{2l+1}\}$ by induction hypothesis. We can thus deduce that the triangle $xyzx$ also contains at least two vertices from $\{x_1, x_2, \ldots, x_{2l+1}\}$ as y_i and y'_i are both outside $\{x_1, x_2, \ldots, x_{2l+1}\}$. So (4) is proved.

It can be seen from (2) that the minimum size of an FVS of T_{2l+1} is at least $l+1$. In view of (3), Theorem 1.1 (which is equivalent to Theorem 1.4(i)) guarantees the existence of at least $l+1$ vertex-disjoint triangles in T_{2l+1}. By (4), each of these $l+1$ triangles contains at least two vertices from $\{x_1, x_2, \ldots, x_{2l+1}\}$. Hence the size of $\{x_1, x_2, \ldots, x_{2l+1}\}$ is at least $2(l+1)$, which is a contradiction.

Since there are $O(|V|^3)$ triangles altogether in T, it takes $O(|V|^3)$ time to find the edge set of G. From the proof we see that G is a bipartite graph. Since the two color classes of G can be obtained in linear time by using depth first search, R can be found in $O(|V|^3)$ time. □

LEMMA 2.4. *Let $T = (V, A)$ be a tournament with no subtournament isomorphic to F_1 nor F_2. Suppose S is a subset of V that contains at least two vertices from every triangle in T. Then $V(C) \nsubseteq V(\mathscr{D}_S)$ for every triangle $C \in \mathscr{F}_S$.*

Proof. We may assume that T is strongly connected; otherwise, we turn to consider the strongly connected components of T separately. Thus V admits a partition $V_1, V_2, \ldots, V_k$ as described in Lemma 2.1. For every $v \in V$, we use $l(v)$ to denote the index i such that $v \in V_i$. Let $D = (V, B)$ be the digraph obtained from T by deleting all arcs from V_j to V_i with $i \leq j-2$; in other words, $(u, v) \in B$ if and only if $(u, v) \in A$ and $|l(u) - l(v)| \leq 1$. So each arc (u, v) in D falls into precisely one of the following three categories: We call (u, v) an *upward* arc if $l(u) = l(v) - 1$, a *downward* arc if $l(u) = l(v) + 1$, and a *level* arc if $l(u) = l(v)$. By Lemma 2.1(ii), we have the following:

(1) D contains no triangle. A (directed) path in D is called *upward* if it consists of three vertices and two upward arcs. It follows from Lemma 2.1 that an upward path P in D corresponds to a triangle in T (induced by $V(P)$), and vice versa. By the hypothesis on S, we get

(2) $|V(P) \cap S| \geq 2$ for any upward path P in D. We prove the lemma by contradiction. Assume the contrary: $\{a, b, c\} \subseteq V(\mathscr{D}_S)$ for some triangle $abca \in \mathscr{F}_S$. Suppose $i = l(a) = l(b) - 1 = l(c) - 2$ for some $1 \leq i \leq k-2$. Then (4*) guarantees the existence of three triangles $xx'x''x$, $yy'y''y$, and $zz'z''z$ in $\mathscr{D}_S$ such that $a \in \{x, x', x''\}$, $b \in \{y, y', y''\}$, and $c \in \{z, z', z''\}$ and that

(3) $xx'x''$, $yy'y''$, and $zz'z''$ are upward paths; that is, $l(x) = l(x') - 1 = l(x'') - 2$, $l(y) = l(y') - 1 = l(y'') - 2$, and $l(z) = l(z') - 1 = l(z'') - 2$.

Since each upward path in D corresponds to a triangle in T, it follows from (4*) that

(4) no upward path in D can go through two vertices in $\{a, b, c\}$ and a vertex in $V - S$. In particular, for any $u \in V - S$ and $v \in \{a, b, c\}$ with $|l(u) - l(v)| = 1$, the arc between u and v is downward unless $v \in \{a, c\}$ and $l(u) = i + 1$.

Using (1), (3), (4), and the fact $|\{x, x', x''\} \cap S| = |\{y, y', y''\} \cap S| = |\{z, z', z''\} \cap S| = 2$ (by (3*)), we can enumerate all possible configurations of the three triangles $xx'x''x$, $yy'y''y$, and $zz'z''z$, which are described in (5), (6), and (7), respectively; see Figure 3 for an illustration, where vertices in S are indicated by black points and those outside S by small circles.

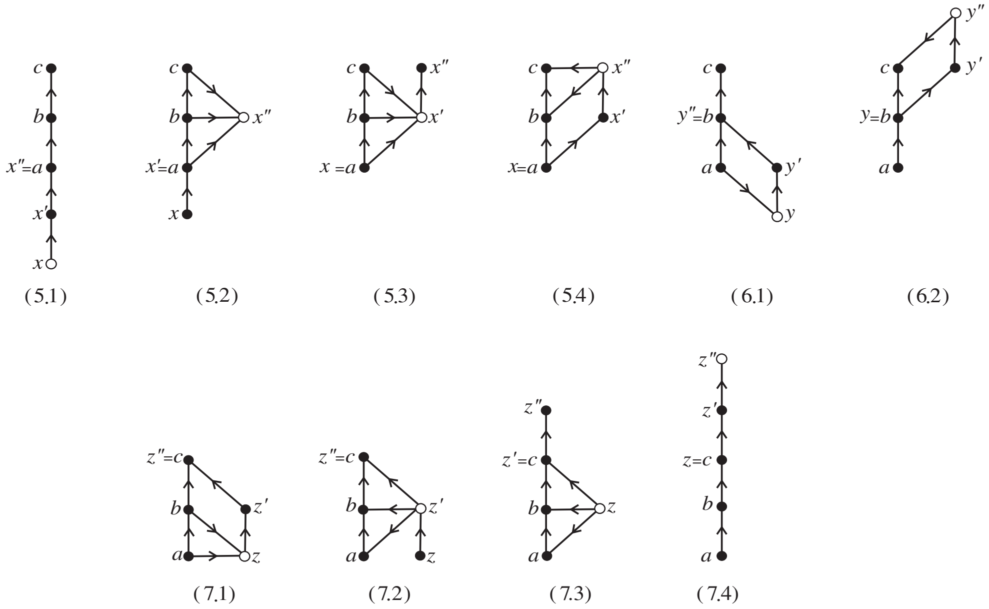


FIG. 3. *Possible configurations of triangles $xx'x''x$, $yy'y''y$, and $zz'z''z$.*

(5) For triangle $xx'x''x$, exactly one of the following holds:

(5.1) $x \in V_{i-2} - S$, $x' \in V_{i-1} \cap S$, $x'' = a \in V_i \cap S$;

(5.2) $x \in V_{i-1} \cap S$, $x' = a \in V_i \cap S$, $x'' \in V_{i+1} - S$, and downward $(c, x'') \in B$, level $(b, x'') \in B$;

(5.3) $x = a \in V_i \cap S$, $x' \in V_{i+1} - S$, $x'' \in V_{i+2} \cap S$, and downward $(c, x') \in B$, level $(b, x') \in B$;

(5.4) $x = a \in V_i \cap S$, $x' \in V_{i+1} \cap S$, $x'' \in V_{i+2} - S$, and downward $(x'', b) \in B$, level $(x'', c) \in B$.

(6) For triangle $yy'y''y$, exactly one of the following holds:

(6.1) $y \in V_{i-1} - S$, $y' \in V_i \cap S$, $y'' = b \in V_{i+1} \cap S$, and downward $(a, y) \in B$;

(6.2) $y = b \in V_{i+1} \cap S$, $y' = V_{i+2} \cap S$, $y'' \in V_{i+3} - S$, and downward $(y'', c) \in B$.

(7) For triangle $zz'z''z$, exactly one of the following holds:

(7.1) $z \in V_i - S$, $z' \in V_{i+1} \cap S$, $z'' = c \in V_{i+2} \cap S$, and downward $(b, z) \in B$, level $(a, z) \in B$;

(7.2) $z \in V_i \cap S$, $z' \in V_{i+1} - S$, $z'' = c \in V_{i+2} \cap S$, and downward $(z', a) \in B$, level $(z', b) \in B$;

(7.3) $z \in V_{i+1} - S$, $z' = c \in V_{i+2} \cap S$, $z'' \in V_{i+3} \cap S$, and downward $(z, a) \in B$, level $(z, b) \in B$;

(7.4) $z = c \in V_{i+2} \cap S$, $z' \in V_{i+3} \cap S$, $z'' \in V_{i+4} - S$.

(8) The following statements hold:

(8.1) Either (5.1) or (6.1) fails;

(8.2) either (6.2) or (7.4) fails;

(8.3) either (5.4) or (6.2) fails;

(8.4) either (6.1) or (7.1) fails.

To justify (8.1), suppose to the contrary that both (5.1) and (6.1) hold. Using (1) and path $x'x''y = x'ay$, we get level $(x', y) \in B$, which in turn gives upward $(x, y) \in B$ (as path $xx'y$ does not correspond to a triangle in D). Thus xyy' is an upward path with x, y outside S, contradicting (2). Hence we have (8.1). Similarly, the violation of (8.2) (resp., (8.3), (8.4)) would give $\{(y'', z'), (y'', z'')\} \subseteq B$ (resp., $\{(x'', y'), (x'', y'')\} \subseteq B$, $\{(y', z), (y, z)\} \subseteq B$) and upward path $y'y''z''$ (resp., $x'x''y''$, yzz'), contradicting (2) again.

(9) The following statements hold:

(9.1) Either (5.1) or (7.1) fails;

(9.2) either (5.4) or (7.4) fails.

Indeed, if both (5.1) and (7.1) hold, then, using (1) and path $x'az$, we have upward $(x', z) \in B$ and hence have the upward path $xx'z$ with $\{x, z\} \subseteq V - S$, contradicting (2). Similarly, if both (5.4) and (7.4) hold, then $(x'', z') \in B$, and so the upward path $x''z'z''$ contradicts (2).

(10) The following statements hold:

(10.1) Either (5.4) or (7.2) fails;

(10.2) either (5.3) or (7.1) fails.

Indeed, if both (5.4) and (7.2) hold, then, using (1) and path $z'ax'$, we have level $(z', x') \in B$. In view of path $z'x'x''$, we further have upward $(z', x'') \in B$. Thus $zz'x''$ contradicts (2). Similarly, if both (5.3) and (7.1) hold, then we have level $(z', x') \in B$ and upward $(z, x') \in B$. It follows that the upward path $zx'x''$ contradicts (2).

(11) The following statements hold:

(11.1) Either (5.4) or (7.3) fails;

(11.2) either (5.2) or (7.1) fails.

Indeed, if both (5.4) and (7.3) hold, then, using (1) and paths $x''cz''$ and zax', we have upward $(x'', z'') \in B$ and level $(z, x') \in B$, respectively. Using path $zx'x''$, we obtain upward $(z, x'') \in B$. Thus the upward path $zx''z''$ contradicts (2). Similarly, if both (5.2) and (7.1) hold, then we have $\{(x, z), (z', x''), (z, x'')\} \subseteq B$. Thus the upward path xzx'' contradicts (2).

(12) Either (5.4) or (6.2) holds.

Suppose otherwise; then from (6) and (5), we see that (6.1) and one of (5.1)–(5.3) hold. In view of (8.1), we further conclude that (5.2) or (5.3) holds. From (5.2) and (5.3), it follows that $\{u\} = \{x', x''\} \cap V_{i+1} \subseteq V - S$ and $(b, u) \in B$ is level. Using (1) and path $y'y''u$, we have upward $(y', u) \in B$ and hence the upward path $yy'u$ with $\{y, u\} \subseteq V - S$, contradicting (2).

(13) Either (6.1) or (7.1) holds.

Suppose otherwise; then (6) and (7) imply that (6.2) and one of (7.2)–(7.4) hold. Using (8.2), we further conclude that (7.2) or (7.3) holds. By (7.2) and (7.3), we have $\{u\} = \{z, z'\} \cap V_{i+1} - S$ and level $(u, b) \in B$. Using (1) and path $uby' = uyy'$, we get upward $(u, y') \in B$ and hence upward path $uy'y''$, which contradicts (2).

(14) (6.2) holds (so (6.1) fails).

Suppose otherwise; (6.2) fails (so (6.1) holds by (6)). It follows from (12) and (8.4) that (5.4) holds and (7.1) fails. Hence, by (7), one of (7.2), (7.3), and (7.4) holds, which leads to a contradiction to one of (10.1), (11.1), and (9.2).

It follows from (14) and (13) that (7.1) holds, which, together with (9.1) and (10.2), implies that neither (5.1) nor (5.3) holds. Moreover, the combination of (14) and (8.3) yields the failure of (5.4). Thus from (5) we see that (5.2) holds, contradicting (11.2). □

Lemma 2.5. *Let $T = (V, A)$ be a tournament with no subtournament isomorphic to F_1 nor F_2. Suppose S is a subset of V such that $\mathscr{D}_S \cup \mathscr{F}_S \neq \emptyset$ and that $|S \cap V(C)| \geq 2$ for every triangle C of T. Then there exists $R \subseteq S$ such that $|R \cap V(C)| = 1$ for every triangle C in $\mathscr{D}_S \cup \mathscr{F}_S$. Moreover, given S, such an R can be found in $O(|V|^3)$ time.*

Proof. We prove by contradiction. Assume (T, S) is a counterexample with minimum $|S|$. It follows instantly from Lemma 2.3 that $\mathscr{F}_S \neq \emptyset$. Let C_0 be a triangle in $\mathscr{F}_S$. Then Lemma 2.4 guarantees the existence of some $v \in V(C_0)$ with $v \notin V(\mathscr{D}_S)$. By considering $(T, S - \{v\})$, we deduce from the minimality of S that there exists $R \subseteq S - \{v\}$ which contains exactly one vertex from each triangle in $\mathscr{D}_{S-\{v\}} \cup \mathscr{F}_{S-\{v\}}$. Note that $C_0 \in \mathscr{D}_{S-\{v\}}$. So $|R \cap V(C)| = 1$ for every triangle C in $\mathscr{D}_S \cup \mathscr{F}_S$, which is a contradiction.

Let $s_1, s_2, \ldots, s_k$ be all of the vertices in S. We apply the following algorithm to S. While $i \leq k$, do: set $S = S - \{s_i\}$ if s_i is contained in no triangle C such that $|V(C) \cap S| = 2$.

Since there are $O(|V|^2)$ triangles altogether in T containing s_i, each iteration takes $O(|V|^2)$ time, and hence the whole algorithm runs in $O(|V|^3)$ time. Let S' denote the resulting S. From the above proof, we see that $\mathscr{F}_{S'} = \emptyset$ and that $\mathscr{D}_{S'} = \mathscr{D}_S \cup \mathscr{F}_S$, where S is the initial one. By Lemma 2.3 (with S' in place of S over there), we can find a subset R of S' in $O(|V|^3)$ time, such that $|R \cap V(C)| = 1$ for every triangle $C \in \mathscr{D}_{S'}$. This R is clearly as desired. $\square$

Now we are ready to establish the min-max relation.

Proof of Theorem 1.2. We shall actually show that statements (ii) and (iii) in Theorem 1.4 are equivalent. For convenience, we use the following notations in our proof. Given a tournament $T = (V, A)$ and a weight function $w \in \mathbf{Z}_+^V$, let τ_w denote the minimum weight of a triangle in T, and let ν_w denote the maximum size of a w-FVS packing of T. Recall that we always have

(1) $\nu_w \leq \tau_w$.

(ii)⇒(iii) Suppose the contrary: $T = (V, A)$ contains a subtournament F isomorphic to F_1 or F_2. Define $w \in \boldsymbol{Z}_+^V$ as $w(v) = 1$ for each $v \in V$. Then $\tau_w = 3$. It is easy to see that each FVS of T contains at least two vertices in F. Since $|V(F)| = 5$, we have $\nu_w \leq 2$. Hence $\tau_w \neq \nu_w$, contradicting (ii).

(iii)⇒(ii) Let $T = (V, A)$ be a tournament with no F_1 nor F_2. To prove that $\nu_w = \tau_w$ for any $w \in \boldsymbol{Z}_+^V$, we apply induction on $|V|$.

The min-max relation holds trivially when $|V| \leq 3$. So we proceed to the induction step and assume that we have already proved the assertion for any tournament with no F_1 nor F_2 and with fewer vertices than T.

To establish the induction step, we apply induction on τ_w. Clearly, $\tau_w = \nu_w$ if $\tau_w = 0$. So we assume $\tau_w > 0$ and distinguish between two cases.

Case 1. $w(z) \geq \tau_w$ for some vertex $z \in V$. Set $w' = w|_{V-\{z\}}$. By the induction hypothesis on $T \backslash z$ (with respect to the weight function w'), we get $\nu_{w'} = \tau_{w'}$. So it can be seen that

- either $T \backslash z$ is acyclic,
- or there exists a w'-FVS packing $\mathcal{S}'$ of $T \backslash z$ with size τ_w (for $\tau_{w'} \geq \tau_w$).

In the former case, define $\mathcal{S}$ to be the multiset consisting of τ_w copies of $\{z\}$; in the latter case, define $\mathcal{S} := \{S' \cup \{z\} : S' \in \mathcal{S}'\}$. Then $\mathcal{S}$ is a collection of FVSs of T with size τ_w, which clearly yields a w-FVS packing of T with size τ_w (by the assumption of case 1). So by (1) we have $\nu_w = \tau_w$.

Case 2. $w(z) < \tau_w$ for any vertex $z \in V$. Set $S := \{v \in V : w(v) \geq 1\}$. It follows from the assumption of the present case that

(2) $|S \cap V(C)| \geq 2$ for every triangle C in T.

In view of (2), the set of triangles of T is the disjoint union of three sets $\mathscr{D}_S$, $\mathscr{F}_S$, and $\mathscr{F}_S^+$ (recall (3*)–(5*)). It follows from the definition of $\mathscr{F}_S^+$ that

(3) all triangles C with $\sum_{v \in V(C)} w(v) = \tau_w$ are contained in $\mathscr{D}_S \cup \mathscr{F}_S$.

By Lemma 2.5, there exists $R \subseteq S$ such that $|R \cap V(C)| = 1$ for every triangle C in $\mathscr{D}_S \cup \mathscr{F}_S$. From the definition of $\mathscr{F}_S^+$, we see that R is an FVS of T. Set $\delta = \min\{w(v) : v \in R\}$. Then $\delta \geq 1$. Define $w' \in \boldsymbol{Z}_+^V$ as $w'(v) = w(v) - \delta|R \cap \{v\}|$ for all $v \in V$.

(4) $\tau_{w'} = \tau_w - \delta$.

To justify (4), it suffices to show that $|R \cap V(C)| \leq 2$ for every $C \in \mathscr{F}_S^+$ (by (3) and the selection of R). It is the case since any such C shares with some triangle in $\mathscr{D}_S$ two vertices, one of which is in $S - R$. So (4) follows.

By the induction hypothesis on $\tau_{w'}$ and by (4), T has a w'-FVS packing $\mathcal{S}$ of size $\tau_w - \delta$. Clearly, $\{R, R, \ldots, R\} \cup \mathcal{S}$ is a collection of FVSs of T with size τ_w, where the multiplicity of R is δ. This collection clearly yields a w-FVS packing of T with size τ_w. So by (1) we have $\nu_w = \tau_w$.

Combining the above two cases, we complete the proof of the induction step and hence our min-max theorem. □

3. NP-completeness.

For convenience, let us call the problem addressed in Theorem 1.6 the *partition problem.* We show its NP-completeness in this section.

Proof of Theorem 1.6. Clearly, the partition problem is in NP. To prove the assertion, we appeal to the following Not-All-Equal 3-Satisfiability problem (Not-All-Equal-3SAT): Given n Boolean variables $\lambda_1, \lambda_2, \ldots, \lambda_n$ and m clauses $c_1, c_2, \ldots, c_m$ in CNF, each of which contains exactly three literals (variables or their negation), determine whether there exists an assignment of Boolean values to the variables such that for each clause at least one literal is true and at least one literal is false. It was shown by Schaefer [16] that Not-All-Equal-3SAT is NP-complete. Our objective is to reduce Not-All-Equal-3SAT to the partition problem.

For this purpose, let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the set of variables, and let $c_1, c_2, \ldots, c_m$ be the set of clauses in an arbitrary instance of Not-All-Equal-3SAT. We propose to construct a tournament T with $5n + 3m + 3$ vertices such that the vertex set of T can be partitioned into two FVSs if and only if $c_1 \wedge c_2 \wedge \cdots \wedge c_m$ is satisfiable (with respect to Not-All-Equal-3SAT). The construction goes as follows (see Figures 4 and 5 for an illustration):

(i) To every variable λ_i, $1 \leq i \leq n$, we associate a tournament X_i with vertex set

$$V(X_i) = \{x_i^h : h = 1, 2, 3, 4, 5\}$$

and arc set

$$A(X_i) = \{(x_i^g, x_i^h) : 1 \leq g < h \leq 5 \text{ and } (g, h) \notin \{(1,5), (2,4)\}\} \cup \{(x_i^5, x_i^1), (x_i^4, x_i^2)\};$$

(ii) to every clause $c_j = c_j^1 \vee c_j^2 \vee c_j^3$, $1 \leq j \leq m$, we associate a triangle $Z_j = z_j^1 z_j^2 z_j^3 z_j^1$;

(iii) let $V := (\cup_{i=1}^n V(X_i)) \cup V(Y) \cup (\cup_{j=1}^m V(Z_j))$, where $Y = y_1 y_2 y_3 y_1$ is a triangle, and all X_i's, Y, and Z_j's are pairwise disjoint;

(iv) to every $z = z_l^k \in \cup_{j=1}^m V(Z_j)$, $1 \leq k \leq 3$, $1 \leq l \leq m$, we associate an arc α_z from z_l^k to x_i^1 if $c_l^k = \lambda_i$ and from z_l^k to x_i^5 if $c_l^k = \bar{\lambda}_i$;

(v) let A be the disjoint union of $\cup_{i=1}^n A(X_i)$, $A(Y)$, $\cup_{j=1}^m (A(Z_j) \cup \{\alpha_z : z \in V(Z_j)\})$, and $\{(u, v) : u \text{ and } v \text{ satisfy one of (a)–(e)}\}$:

(a) $u \in V(X_i)$, $v \in V(X_{i'})$, and $i < i'$;

(b) $u \in \cup_{i=1}^n V(X_i)$ and $v \in V(Y)$;

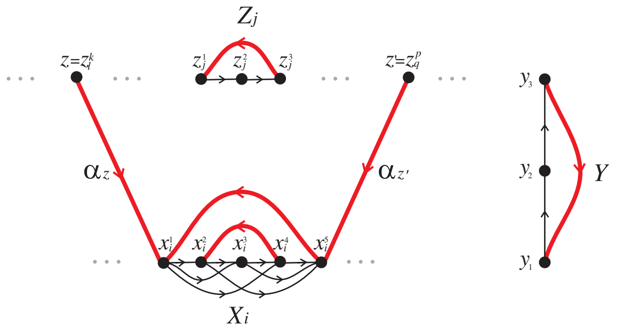


FIG. 4. *An illustration of constructions* (i)–(iv), *where* $c_l^k = \lambda_i$ *and* $c_q^p = \bar{\lambda}_i$.

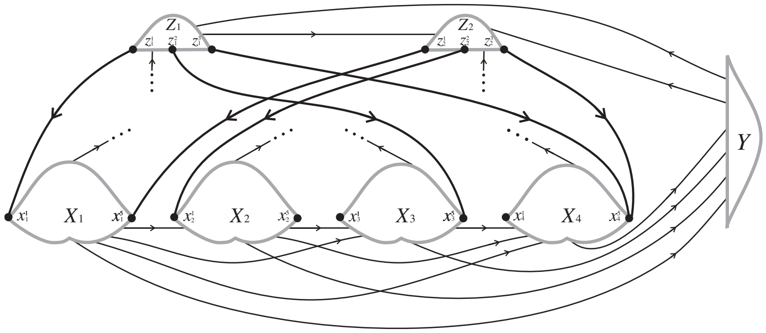


FIG. 5. *Tournament* T *resulting from the instance* $(\lambda_1 \vee \bar{\lambda}_3 \vee \bar{\lambda}_4) \wedge (\bar{\lambda}_1 \vee \lambda_2 \vee \bar{\lambda}_4)$.

(c) $u \in \cup_{i=1}^{n} V(X_i)$, $v \in \cup_{j=1}^{m} V(Z_j)$, and α_v is not directed to u;
(d) $u \in V(Y)$ and $v \in \cup_{j=1}^{m} V(Z_j)$;
(e) $u \in V(Z_j)$, $v \in V(Z_{j'})$, and $j < j'$.

The construction is completed. It is easy to see that the construction can be accomplished in polynomial time, and the resulting digraph $T = (V, A)$ is a tournament. The tournament T resulting from the NOT-ALL-EQUAL-3SAT instance with $n = 4$, $m = 2$, $c_1 = \lambda_1 \vee \bar{\lambda}_3 \vee \bar{\lambda}_4$, and $c_2 = \bar{\lambda}_1 \vee \lambda_2 \vee \bar{\lambda}_4$ is illustrated in Figure 5.

Let us define a linear order $\prec$ on the vertex set of T as follows: $x_1^1 \prec x_1^2 \prec x_1^3 \prec x_1^4 \prec x_1^5 \prec x_2^1 \prec x_2^2 \prec x_2^3 \prec x_2^4 \prec x_2^5 \prec \cdots \prec x_n^1 \prec x_n^2 \prec x_n^3 \prec x_n^4 \prec x_n^5 \prec y_1 \prec y_2 \prec y_3 \prec z_1^1 \prec z_1^2 \prec z_1^3 \prec z_2^1 \prec z_2^2 \prec z_2^3 \prec \cdots \prec z_m^1 \prec z_m^2 \prec z_m^3$. Observe

(1) Set

$$\begin{aligned} B := {} & \{(x_i^5, x_i^1), (x_i^4, x_i^2) : 1 \le i \le n\} \cup \{(y_3, y_1)\} \\ & \cup \{(z_j^3, z_j^1) : 1 \le j \le m\} \cup \{\alpha_z : z \in \cup_{j=1}^{m} V(Z_j)\}. \end{aligned}$$

(In Figure 4, the arcs in B are bold lined.) Then for any $u, v \in V$ with $u \prec v$, arc $(v, u) \in A$ if and only if $(v, u) \in B$;

(2) for every $1 \le i \le n$, there are four triangles

$$X_i^1 = x_i^2 x_i^3 x_i^4 x_i^2 \text{and } X_i^h = x_i^1 x_i^h x_i^5 x_i^1,\ h = 2, 3, 4,$$

altogether in tournament X_i; and

(3) for every $z \in \cup_{j=1}^{m} V(Z_j)$, there are three triangles Y_z^i, $i = 1, 2, 3$, altogether in T through α_z and y_i.

It follows from (1) that every triangle in T contains one or two arcs in B. Furthermore, since no two arcs in $B - \{\alpha_z : z \in \cup_{j=1}^m V(Z_j)\}$ have a common end, from the construction of T and (2) we see that

(4) every triangle of T is either in $\{X_i^h : 1 \le h \le 4, 1 \le i \le n\} \cup \{Y\} \cup \{Z_j : 1 \le j \le m\}$ or contains α_z for some $z \in \cup_{j=1}^m V(Z_j)$.

Now we are ready to show that the vertex set of T can be partitioned into two FVSs if and only if the Not-All-Equal-3SAT instance $c_1 \wedge c_2 \wedge \cdots \wedge c_m$ is satisfiable.

Sufficiency. Suppose there is a truth assignment for $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ such that each clause c_j, $1 \le j \le m$, contains at least one true literal and at least one false literal. Set

- $X := \{x_i^1 : \lambda_i \text{ is true}, 1 \le i \le n\} \cup \{x_i^5 : \lambda_i \text{ is false}, 1 \le i \le n\}$;
- $\bar{X} := \{x_i^1 : \lambda_i \text{ is false}, 1 \le i \le n\} \cup \{x_i^5 : \lambda_i \text{ is true}, 1 \le i \le n\}$;
- $Z := \{z \in \cup_{j=1}^m V(Z_j) : \alpha_z = (z, x), x \in X\}$; and
- $\bar{Z} := \cup_{j=1}^m V(Z_j) - Z = \{z \in \cup_{j=1}^m V(Z_j) : \alpha_z = (z, x), x \in \bar{X}\}$.

It is easy to see that

(5) $|X \cap V(X_i^h)| = |\bar{X} \cap V(X_i^h)| = 1$ for every $h = 2, 3, 4$ and $1 \le i \le n$;

(6) for every $z \in \cup_{j=1}^m V(Z_j)$, if the head of α_z is in X (resp., $\bar{X}$), then its tail is in Z (resp., $\bar{Z}$); and

(7) V is the disjoint union of two sets
   - $S_1 := X \cup \bar{Z} \cup \{x_i^2 : 1 \le i \le n\} \cup \{y_1\}$ and
   - $S_2 := \bar{X} \cup Z \cup \{x_i^3, x_i^4 : 1 \le i \le n\} \cup \{y_2, y_3\}$.

We claim that both S_1 and S_2 are FVSs of T. To justify this, let C be an arbitrary triangle C of T. Let us show that C meets both S_1 and S_2. By (5) and (6), the statement holds if $C \in \{X_i^h : h = 2, 3, 4;\ 1 \le i \le n\}$ or if C contains some α_z. If $C = X_i^1$ for some $1 \le i \le n$, then we derive from (2) and (7) that $x_i^2 \in V(C) \cap S_1$ and $x_i^3 \in V(C) \cap S_2$. If $C = Y$, then $y_1 \in V(C) \cap S_1$ and $y_2 \in V(C) \cap S_2$. So by (4) it remains to consider the case when $C = Z_j$ for some $1 \le j \le m$. Recall that c_j^h is true and c_j^i is false for some $1 \le h \ne i \le 3$. From (iv) and the definitions of X and $\bar{X}$, we deduce that $x \in X$ and $x' \in \bar{X}$, where (z_j^h, x) and (z_j^i, x') are arcs associated to z_j^h and z_j^i, respectively, as described in (iv). It follows from (6) and (7) that $z_j^h \in Z \cap V(C) \subseteq S_2 \cap V(C)$ and $z_j^i \in \bar{Z} \cap V(C) \subseteq S_1 \cap V(C)$. Therefore both S_1 and S_2 are FVSs of T, as claimed. By (7), we are done.

Necessity. Suppose the vertex set of T can be partitioned into two FVSs S_1 and S_2. For $1 \le i \le n$, set λ_i to be true if $x_i^1 \in S_1$ and false otherwise. Let us show that this assignment enables every c_j, $1 \le j \le m$, to contain at least one true literal and at least one false literal. To this end, we first show that

(8) $|\{x_i^1, x_i^5\} \cap S_1| = |\{x_i^1, x_i^5\} \cap S_2| = 1$ for all $1 \le i \le n$.

Indeed, by (2), we have $x_i^g \in S_1 \cap V(X_i^1)$ and $x_i^h \in S_2 \cap V(X_i^1)$ for some $2 \le g \ne h \le 4$. This, in turn, implies that $\{x_i^1, x_i^5\} \cap S_2 \ne \emptyset$ and $\{x_i^1, x_i^5\} \cap S_1 \ne \emptyset$ by considering triangles $X_i^g = x_i^1 x_i^g x_i^5 x_i^1$ and $X_i^h = x_i^1 x_i^h x_i^5 x_i^1$. So (8) is established.

Next we observe that

(9) $|\{x, z\} \cap S_1| = |\{x, z\} \cap S_2| = 1$ for all $\alpha_z = (z, x)$ with $z \in \cup_{j=1}^m V(Z_j)$ and $x \in \{x_i^1, x_i^5 : 1 \le i \le n\}$.

Indeed, by (iii) and the definition of S_1 and S_2, triangle $y_1 y_2 y_3 y_1$ contains $y_g \in S_1$ and $y_h \in S_2$ for some $1 \le g \ne h \le 3$. Recall (3), T contains triangles $Y_z^i = x y_i z x$, $i = 1, 2, 3$. Now using triangles Y_z^g and Y_z^h, we obtain $\{x, z\} \cap S_2 \ne \emptyset$ and $\{x, z\} \cap S_1 \ne \emptyset$. Hence (9) holds.

For $1 \le j \le m$, by (ii) triangle Z_j contains some $z_j^g \in S_1$ and $z_j^h \in S_2$. Suppose (z_j^g, x) and (z_j^h, x') are arcs associated to z_j^g and z_j^h, respectively, as described in (iv), where $\{x, x'\} \subseteq \{x_i^1, x_i^5 : 1 \leq i \leq n\}$. It follows from (9) that $x \in S_2$ and $x' \in S_1$. Suppose $x \in \{x_i^1, x_i^5\}$ for some $1 \leq i \leq n$.

- If $x = x_i^1$, then, by (iv), $c_j^g = \lambda_i$ is false as $x_i^1 \notin S_1$;
- if $x = x_i^5$, then, by (8), $x_i^1 \in S_1$. So λ_i is true, and hence, by (iv), $c_j^g = \bar{\lambda}_i$ is false.

Therefore c_j^g is false (in either case). Similarly, it can be deduced from $x' \in S_1$ that c_j^h is true. Hence c_j contains both false literal c_j^g and true literal c_j^h; equivalently, $c_1 \wedge c_2 \wedge \cdots \wedge c_m$ is satisfiable, completing the proof. □

4. Algorithms. For simplicity, we use the same notations as introduced before. In particular, given a tournament $T = (V, A)$ and a weight function $w \in \mathbf{Z}_+^V$, let τ_w denote the minimum weight of a triangle in T, and let ν_w denote the maximum size of a w-FVS packing of T.

For the case when T contains no F_1 nor F_2, we present the following algorithm for finding an optimal w-FVS packing of size ν_w.

Algorithm Opt_Pack OPTIMAL_FVS_PACKING

```
Input   A tournament $T = (V, A)$ with no $F_1$ nor $F_2$ and a weight $w \in \boldsymbol{Z}_+^V$
Output  A maximum $w$-FVS packing $\mathcal{S}$ of $T$ with $|\mathcal{S}| = \nu_w$
  1. $\tau_w \leftarrow$ the minimum weight of a triangle in $T$
  2. if $\tau_w = 0$ or $T$ is acyclic, then return $\mathcal{S} \leftarrow \emptyset$
  3. if $\exists\, z \in V$ with $w(z) \geq \tau_w$, then
  4.     if $T \backslash z$ is acyclic, then return $\mathcal{S} \leftarrow \{S_i : S_i = \{z\},\ i = 1, 2, \ldots, \tau_w\}$
  5.                 else $\{S_i : 1 \leq i \leq \tau_{w|_{V-\{z\}}}\} \leftarrow$ Opt_Pack$(T \backslash z, w|_{V-\{z\}})$
                          return $\mathcal{S} \leftarrow \{\{z\} \cup S_i : 1 \leq i \leq \tau_w\}$
  6. $S \leftarrow \{v \in V : w(v) \geq 1\}$, $R \leftarrow$ a subset of $S$ with $|R \cap V(C)| = 1$ for all
     $C \in \mathscr{D}_S \cup \mathscr{F}_S$
  7. $\delta \leftarrow \min\{w(v) : v \in R\}$, $w'(v) \leftarrow w(v) - \delta|R \cap \{v\}|$ for all $v \in V$
  8. return $\mathcal{S} \leftarrow \{S_i : S_i = R,\ i = 1, 2, \ldots, \delta\} \cup$ Opt_Pack$(T, w')$
```

Remark. Note that $\mathcal{S}$ is a collection of FVSs of T with size ν_w, which obviously yields a w-FVS packing of T with the same size.

THEOREM 4.1. *Let $T = (V, A)$ be a tournament with no F_1 nor F_2. Then Algorithm* OPTIMAL_FVS_PACKING *solves the FVS packing problem on T exactly in $O(|V|^4)$ time.*

Proof. The correctness of the algorithm follows instantly from the proof of Theorem 1.2. Let us now estimate the time complexity of the algorithm.

Note that either in steps 3–5 one vertex z is eliminated from our consideration or in step 7 the weight of at least one vertex becomes zero (from nonzero one). So the whole algorithm takes $O(|V|)$ iterations. From Lemma 2.5, we can conclude that each iteration takes $O(|V|^3)$ time. Hence the total running time of the algorithm is $O(|V|^4)$. □

Let us proceed to the FVS packing problem on a general tournament T. For this general case, we can easily obtain a 1/3-approximation algorithm: Set $R = \emptyset$. While T contains a triangle C, do: let v be a vertex in $V(C)$ of maximum weight. Set $R = R \cup \{v\}$ and $T = T \backslash v$. Obviously, $\{R, R, \ldots, R\}$, where the multiplicity of R is $\min\{w(v) : v \in R\}$, is an FVS packing in the original T with size at least $\frac{1}{3}\nu_w$. By exploiting the structural characterization given in our min-max theorem and using the above exact algorithm as a subroutine, we can obtain a better approximation algorithm based on the subgraph removal technique.

Algorithm Apx_Pack APPROXIMATE_FVS_PACKING

```
Input   A tournament $T = (V, A)$ and a weight $w \in \boldsymbol{Z}_+^V$
Output  A $w$-FVS packing $\mathcal{S}$ of $T$ with $|\mathcal{S}| \geq \frac{2}{5}\nu_w$
   1. $\tau_w \leftarrow$ the minimum weight of a triangle in $T$, $R \leftarrow \emptyset$
   2. if $\tau_w = 0$ or $T$ is acyclic, then return $\mathcal{S} \leftarrow \emptyset$
   3. while $T$ contains a subtournament $F$ isomorphic to $F_1$ or $F_2$ do
   4.         $v \leftarrow$ a vertex in $V(F)$ of maximum weight, $R \leftarrow R \cup \{v\}$, $T \leftarrow T \backslash v$
   5. end-while
   6. $\mathcal{S}' = \{S_i : 1 \leq i \leq \tau_{w|_{V(T)}}\} \leftarrow$ Opt_Pack$(T, w|_{V(T)})$, $\delta \leftarrow \min\{w(v) : v \in R\}$
   7. if $\mathcal{S}' = \emptyset$, then return $\mathcal{S} \leftarrow \{S_i : S_i = R, i = 1, 2, \ldots, \delta\}$
   8.             else return $\mathcal{S} \leftarrow \{S_i \cup R : i = 1, 2, \ldots, \min\{\tau_{w|_{V(T)}}, \delta\}\}$
```

Remark. Again $\mathcal{S}$ is a collection of FVSs of T, which obviously yields a w-FVS packing of T with the same size.

THEOREM 4.2. *Let $T = (V, A)$ be an arbitrary tournament. Then Algorithm* APPROXIMATE_FVS_PACKING *approximates the FVS packing problem on T within a factor of* $2/5$ *in $O(|V|^4)$ time.*

Proof. Clearly, $\mathcal{S}$ is a collection of FVSs of T. To get the approximation ratio, it suffices to prove that

(1) $|\mathcal{S}| \geq \frac{2}{5}\nu_w$.

For this purpose, we turn to show that

(2) $\delta \geq \frac{2}{5}\nu_w$ if $\delta > 0$.

To justify (2), let u be a vertex in R with $w(u) = \delta$. Suppose u is added to R because of subtournament F (recall the while-loop of the algorithm), and suppose $\mathcal{S}^*$ is a w-FVS packing of T with size ν_w. Since we need to delete at least two vertices in F in order to destroy all triangles in F, each FVS in $\mathcal{S}^*$ contains at least two vertices in F. From the definition of a w-FVS packing, we deduce that $2|\mathcal{S}^*| \leq \sum_{v \in V(F)} w(v)$. Since u is a vertex with maximum weight in F and $|V(F)| = 5$, we have $2\nu_w = 2|\mathcal{S}^*| \leq 5w(u) = 5\delta$, yielding (2).

To establish (1), we may assume $\tau_w > 0$, for otherwise the statement holds trivially. So we have $\delta > 0$ when $R \neq \emptyset$. If $\mathcal{S}' = \emptyset$, then it follows from (2) and step 7 of the algorithm that (1) holds. Otherwise, $\tau_{w|_{V(T)}}$ in step 6 of the algorithm is at least $\tau_w (\geq \nu_w)$. Thus from step 8 of the algorithm we can also conclude (1).

It was shown in [1] that F in step 3 can be obtained in $O(|V|^2)$ time if it exists. Thus we deduce from Theorem 4.1 that APPROXIMATE_FVS_PACKING runs in $O(|V|^4)$ time. □

It is easy to see that Theorems 1.7 and 1.8 follow from the above two theorems, respectively.

5. Concluding remarks.

In this paper we have characterized all tournaments T with Mengerian hypergraph $b(\mathcal{C}_T)$. Coincidently, $b(\mathcal{C}_T)$ is Mengerian if and only if $\mathcal{C}_T$ is. Major open problems in this research direction are to characterize all digraphs G with Mengerian $\mathcal{C}_G$ and those with Mengerian $b(\mathcal{C}_G)$. The arc versions of these problems are equally interesting. While these problems are extremely hard in general, Guenin and Thomas [14] successfully characterized all digraphs that pack, where a digraph G *packs* if for any subdigraph H of G the maximum number of disjoint cycles is equal to the minimum number of vertices in a feedback vertex set in H. Guenin strongly believes that the blocker version of their theorem holds on exactly the same digraphs.

Conjecture 5.1 ([13]). *A digraph G packs if and only if for any subdigraph H of G the maximum number of disjoint feedback vertex sets is equal to the length of the shortest cycle in H.*

We close this paper by the aforementioned Woodall's conjecture on packing feedback arc sets.

Conjecture 5.2 ([20]). *In any planar digraph the maximum number of disjoint feedback arc sets is equal to the length of the shortest cycle.*

Certainly, these two beautiful conjectures deserve arduous research efforts.

Acknowledgment. The authors are grateful to two anonymous referees for their invaluable comments and suggestions.

REFERENCES

[1] M. Cai, X. Deng, and W. Zang, *An approximation algorithm for feedback vertex sets in tournaments*, SIAM J. Comput., 30 (2001), pp. 1993–2007.

[2] M. Cai, X. Deng, and W. Zang, *A min-max theorem on feedback vertex sets*, Math. Oper. Res., 27 (2002), pp. 361–371.

[3] X. Chen, G. Ding, X. Hu, and W. Zang, *A min-max relation on packing feedback vertex sets*, Math. Oper. Res., 31 (2006), pp. 777–788.

[4] G. Ding, L. Feng, and W. Zang, *The complexity of recognizing linear systems with certain integrality properties*, Math. Program. Ser. A, to appear.

[5] G. Ding, Z. Xu, and W. Zang, *Packing cycles in graphs*, II, J. Combin. Theory Ser. B, 87 (2003), pp. 244–253.

[6] G. Ding and W. Zang, *Packing cycles in graphs*, J. Combin. Theory Ser. B, 86 (2002), pp. 381–407.

[7] J. Edmonds, *Edge-disjoint branchings*, in Combinatorial Algorithms, B. Rustin, ed., Academic Press, New York, 1973, pp. 91–96.

[8] J. Edmonds and R. Giles, *A min-max relation for submodular functions on graphs*, Ann. Discrete Math., 1 (1977), pp. 185–204.

[9] J. Edmonds and R. Giles, *Total dual integrality of linear inequality systems*, in Progress in Combinatorial Optimization, W. R. Pulleyblank, ed., Academic Press, Toronto, 1984, pp. 117–129.

[10] A. Frank, *A Coloring Question on Digraphs*, DMANET, 1998.

[11] D.R. Fulkerson, *Networks, frames, and blocking systems*, in Mathematics of the Decision Sciences, Part I, G. B. Dantzig and A. F. Veinott, eds., American Mathematical Society, Providence, RI, 1968, pp. 303–334.

[12] D.R. Fulkerson, *Packing rooted directed cuts in a weighted directed graph*, Math. Program., 6 (1974), pp. 1–13.

[13] B. Guenin, *private communication*, 2005.

[14] B. Guenin and R. Thomas, *Packing directed circuits exactly*, Combinatorica, to appear.

[15] C.L. Lucchesi and D.H. Younger, *A minimax theorem for directed graphs*, J. London Math. Soc., 17 (1978), pp. 369–374.

[16] T. J. Schaefer, *The complexity of satisfiability problems*, in Proceedings of the 10th ACM Symposium on Theory of Computing, New York, 1986, pp. 216–226.

[17] A. Schrijver, *A counterexample to a conjecture of Edmonds and Giles*, Discrete Math., 32 (1980), pp. 213–214.

[18] A. Schrijver, *Theory of Linear and Integer Programming*, John Wiley & Sons, New York, 1986.

[19] A. Schrijver, *Combinatorial Optimization - Polyhedra and Efficiency*, Springer-Verlag, Berlin, 2003.

[20] D.R. Woodall, *Menger and Kőnig systems*, Lecture Notes in Math., 642 (1978), pp. 620–635.

SIAM J. COMPUT.
Vol. 37, No. 3, pp. 938–958

THE POWER OF STRONG FOURIER SAMPLING: QUANTUM ALGORITHMS FOR AFFINE GROUPS AND HIDDEN SHIFTS[*]

CRISTOPHER MOORE[†], DANIEL ROCKMORE[‡], ALEXANDER RUSSELL[§], AND LEONARD J. SCHULMAN[¶]

Abstract. Many quantum algorithms, including Shor's celebrated factoring and discrete log algorithms, proceed by reduction to a *hidden subgroup problem*, in which an unknown subgroup H of a group G must be determined from a quantum state ψ over G that is uniformly supported on a left coset of H. These hidden subgroup problems are typically solved by *Fourier sampling*: the quantum Fourier transform of ψ is computed and measured. When the underlying group is nonabelian, two important variants of the Fourier sampling paradigm have been identified: the *weak standard method*, where only representation *names* are measured, and the *strong standard method*, where full measurement (i.e., the row and column of the representation, in a suitably chosen basis, as well as its name) occurs. It has remained open whether the strong standard method is indeed stronger, that is, whether there are hidden subgroups that can be reconstructed via the strong method but *not* by the weak, or any other known, method. In this article, we settle this question in the affirmative. We show that hidden subgroups H of the q-hedral groups, i.e., semidirect products $\mathbb{Z}_q \ltimes \mathbb{Z}_p$, where $q \mid (p-1)$, and in particular the affine groups A_p, can be information-theoretically reconstructed using the strong standard method. Moreover, if $|H| = p/\text{polylog}(p)$, these subgroups can be fully reconstructed with a polynomial amount of quantum and classical computation. We compare our algorithms to two weaker methods that have been discussed in the literature—the "forgetful" abelian method, and measurement in a random basis—and show that both of these are weaker than the strong standard method. Thus, at least for some families of groups, it is crucial to use the full power of representation theory and nonabelian Fourier analysis, namely, to measure the high-dimensional representations in an *adapted basis* that respects the group's subgroup structure. We apply our algorithm for the hidden subgroup problem to new families of cryptographically motivated *hidden shift problems*, generalizing the work of van Dam, Hallgren, and Ip on shifts of multiplicative characters. Finally, we close by proving a simple closure property for the class of groups over which the hidden subgroup problem can be solved efficiently.

Key words. quantum computation, hidden subgroup problem, Fourier analysis, group representations

AMS subject classifications. 81P68, 20C35

DOI. 10.1137/S0097539705447177

1. The hidden subgroup problem.

One of the principal quantum algorithmic paradigms is the use of the abelian Fourier transform to discover a function's hidden periodicities. In the examples relevant to quantum computing, an oracle function f defined on an abelian group G has "hidden periodicity" if there is a "hidden" subgroup H of G so that f is precisely invariant under translation by H or, equivalently, f is constant on the cosets of H and takes distinct values on distinct cosets. The *hidden*

[*] Received by the editors March 9, 2005; accepted for publication (in revised form) December 20, 2006; published electronically August 24, 2007. Support for this work was provided by the California Institute of Technology's Institute for Quantum Information (IQI), the Mathematical Sciences Research Institute (MSRI), the Institute for Advanced Study (IAS), NSF grants ITR-0220070, ITR-0220264, CCR-0093065, CCR-0049092, EIA-0218443, QuBIC-0218563, and CCF-0524613, the Charles Lee Powell Foundation, and the Bell Fund.

http://www.siam.org/journals/sicomp/37-3/44717.html

[†] University of New Mexico, Albuquerque, NM 87131, and Santa Fe Institute, Santa Fe, NM 87501 (moore@cs.unm.edu)

[‡] Dartmouth College, Hanover, NH 03755 (rockmore@cs.dartmouth.edu).

[§] University of Connecticut, Storrs, CT 06269 (acr@cse.uconn.edu).

[¶] California Institute of Technology, Pasadena, CA 91125 (schulman@caltech.edu).

subgroup problem is the problem of determining the subgroup H from such a function. Algorithms for these problems typically adopt the approach detailed below, called *Fourier sampling* [3].

Step 1. Prepare two registers, the first in a uniform superposition over the elements of a group G and the second with the value zero, yielding the state

$$\psi_1 = \frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle \otimes |0\rangle \,.$$

Step 2. Calculate (or, if it is an oracle, query) the function f defined on G and XOR it with the second register. This entangles the two registers and results in the state

$$\psi_2 = \frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle \otimes |f(g)\rangle \,.$$

Step 3. Measure the second register. This produces a uniform superposition over one of f's level sets, i.e., the set of group elements g for which $f(g)$ takes the measured value f_0. As the level sets of f are the cosets of H, this puts the first register in a uniform distribution over superpositions on one of those cosets, namely cH, where $f(c) = f_0$ for some f_0. Moreover, it disentangles the two registers, resulting in the state $\psi_3 \otimes |f_0\rangle$, where ψ_3 is a so-called *coset state*,

$$\psi_3 = |cH\rangle = \frac{1}{\sqrt{|H|}} \sum_{h \in H} |ch\rangle \,.$$

Alternately, since the value f_0 we observe has no bearing on the algorithm, we can use the formulation in which the environment, rather than the user, measures f. In that case, tracing over f yields a mixed state with density matrix

$$\rho_H = \frac{1}{[G:H]} \sum_{f_0} |\psi_3\rangle \langle\psi_3| = \frac{1}{|G|} \sum_{c} |cH\rangle \langle cH| \,,$$

i.e., a classical mixture consisting of one pure state ψ_3 for each coset. Kuperberg refers to this as the *coherent* hidden subgroup problem [18].

Step 4. Carry out the quantum Fourier transform on ψ_3 or ρ_H and measure the result.

For example, in Simon's algorithm [26], the "ambient" group G over which the Fourier transform is performed is $\mathbb{Z}_2^n$, f is an oracle with the promise that $f(x) = f(x+y)$ for some y, and $H = \{0, y\}$ is a subgroup of order 2. In Shor's factoring algorithm [25] G is the group $\mathbb{Z}_n^*$, where n is the number we wish to factor, $f(x) = r^x \bmod n$ for a random $r < n$, and H is the subgroup of $\mathbb{Z}_n^*$ of index order(r). (However, since $|\mathbb{Z}_n^*|$ is unknown, Shor's algorithm actually performs the transform over $\mathbb{Z}_q$, where q is polynomially bounded by n; see [25] or [11, 12].)

These are all abelian instances of the *hidden subgroup problem* (HSP). Interest in *nonabelian* versions of the HSP evolved from the relation to the elusive Graph Automorphism problem: if one could efficiently solve the HSP over the symmetric group S_n, this would yield an efficient quantum algorithm for graph automorphism (see, e.g., Jozsa [16] for a review). This was the impetus behind the development of the first nonabelian quantum Fourier transform [2] and is, in part, the reason that the nonabelian HSP has remained such an active area of research in quantum algorithms.

In general, we will say that the HSP for a family of groups G has a *Fourier sampling* algorithm if a procedure similar to that outlined above works. Specifically, the algorithm prepares a coset state as defined above,

$$|cH\rangle = \frac{1}{\sqrt{|H|}} \sum_{h \in H} |ch\rangle \,,$$

over a random coset cH of the hidden subgroup H, computes the (quantum) Fourier transform of this state, and measures the result. After a polynomial number of such trials, a polynomial amount of classical computation, and, perhaps, a polynomial number of classical queries to the function h to confirm the result, the algorithm produces a set of generators for the subgroup H with high probability.

When G is abelian, measuring a state's Fourier transform has a clear meaning: one observes the frequency χ with probability equal to the squared magnitude of the transform at that frequency. In the case where G is a *nonabelian* group, however, in order to define a full measurement it is necessary to select bases for each representation of G. (We explain this in more detail below.) The subject of this article is the relationship between this choice of basis and the information gleaned from the measurement: are some bases more useful for computation than others?

Since we are typically interested in exponentially large groups, we will take the size of our input to be $n = \log |G|$. Throughout, "polynomial" means polynomial in n and thus polylogarithmic in $|G|$.

1.1. Nonabelian hidden subgroup problems. Although a number of interesting results have been obtained on the nonabelian HSP, the groups for which efficient solutions are known remain woefully few. On the positive side, Roetteler and Beth [22] give an algorithm for the wreath product $\mathbb{Z}_2^k \wr \mathbb{Z}_2$. Ivanyos, Magniez, and Santha [15] extend this to the more general case of semidirect products $K \ltimes \mathbb{Z}_2^k$, where K is of polynomial size, and also give an algorithm for groups whose commutator subgroup is of polynomial size. Friedl, Ivanyos, Magniez, Santha, and Sen [8] solve a problem they call hidden translation and thus generalize this further to what they call "smoothly solvable" groups: these are solvable groups whose derived series is of constant length and whose abelian factors are each the direct product of an abelian group of bounded exponent and one of polynomial size. (See also section 8.)

In another vein, Ettinger and Høyer [6] show that the HSP is solvable for the dihedral groups in an *information-theoretic* sense; namely, a polynomial number of quantum queries to the function oracle gives enough information to reconstruct the subgroup, but the best known reconstruction algorithm takes exponential time. More generally, Ettinger, Høyer, and Knill [7] show that for *arbitrary* groups the HSP can be solved information-theoretically with a finite number of quantum queries. However, their algorithm calls for a quantum measurement for each possible subgroup, and since there might be $|G|^{\Omega(\log |G|)}$ of these, it requires an exponential number of quantum operations.

Our current understanding of the HSP, then, divides group families into three classes.

I. *Fully reconstructible.* Subgroups of a family of groups $\{G_i\}$ are *fully reconstructible* if the HSP can be solved with high probability by a quantum circuit of size polynomial in $\log |G_i|$.

II. *Information-theoretically reconstructible.* Subgroups of a family of groups $\{G_i\}$ are *information-theoretically reconstructible* if the solution to the HSP for G_i is determined information-theoretically by the fully measured result of a quantum circuit of size polynomial in $\log |G_i|$.

III. *Quantum information-theoretically reconstructible.* Subgroups of a family of groups $\{G_i\}$ are *quantum information-theoretically reconstructible* if the solution to the HSP for G_i is determined by the quantum state resulting from a quantum circuit of polynomial size in $\log |G_i|$, in the sense that there exists a positive operator-valued measurement (POVM) that yields the subgroup H with constant probability but where it may or may not be possible to carry out this POVM with a quantum circuit of polynomial size.

In each case, the quantum circuit has oracle access to a function $f : G \to S$, for some set S, with the property that f is constant on each left coset of a subgroup H and distinct on distinct cosets.

In this language, then, subgroups of abelian groups are fully reconstructible, while the result of [7] shows that subgroups of arbitrary groups are quantum information-theoretically reconstructible. The other work cited above has labored to place specific families of nonabelian groups into the more algorithmically meaningful classes I and II.

1.2. Nonabelian Fourier transforms. In this section we give a brief review of nonabelian Fourier analysis but only to the extent needed to set down notation. We refer the reader to [9, 24] for a more complete exposition.

Fourier analysis over a finite abelian group A expresses a function $\phi : A \to \mathbb{C}$ as a linear combination of homomorphisms $\chi : A \to \mathbb{C}$. If $A = \mathbb{Z}_p$, for example, these are the familiar basis functions $\chi_t : z \mapsto \omega_p^{tz}$, where ω_p denotes the pth root of unity $\mathrm{e}^{2\pi i/p}$. Any function $\phi : A \to \mathbb{C}$ can be uniquely expressed as a linear combination of these χ_t, and this change of basis is the Fourier transform.

When G is a nonabelian group, however, this same procedure cannot work: in particular, there are not enough homomorphisms of G into $\mathbb{C}$ to span the space of all $\mathbb{C}$-valued functions on G. To define a sufficient basis, the representation theory of finite groups considers more general functions, namely homomorphisms from G into groups of unitary matrices.

A *representation* of a finite group G is a homomorphism $\rho : G \to \mathrm{U}(d)$, where $\mathrm{U}(d)$ denotes the group of unitary $d \times d$ matrices (with entries from $\mathbb{C}$); the dimension $d = d_\rho$ is referred to as the *dimension* of ρ. If $\rho : G \to \mathrm{U}(d)$ is a representation, a subspace W of $\mathbb{C}^d$ is said to be *invariant* if $\rho(g)(W) \subset W$ for all g. A representation is said to be *irreducible* if the only invariant subspaces are the trivial subspaces $\mathbb{C}^d$ and $\{\vec{0}\}$.

For a function $\phi : G \to \mathbb{C}$ and an irreducible representation ρ, $\hat{\phi}(\rho)$ denotes *the Fourier transform of* ϕ *at* ρ and is defined by

$$\hat{\phi}(\rho) = \sqrt{\frac{d_\rho}{|G|}} \sum_g \phi(g)\rho(g).$$

Note that ϕ takes values in $\mathbb{C}$ while ρ is matrix-valued. It is a fact that a finite group has a finite number of distinct irreducible representations up to isomorphism (i.e., up to a unitary change of basis). The *Fourier transform* of a function $\phi : G \to \mathbb{C}$ is then the collection of matrices $\hat{\phi}(\rho)$, taken over all nonisomorphic irreducible representations ρ.

Fixing a group G and a subgroup H, we shall focus primarily on the functions

$\varphi_c : G \to \mathbb{C}$ of the form

$$\varphi_c(g) = \begin{cases} 1/\sqrt{|H|} & \text{if } g \in cH, \\ 0 & \text{otherwise,} \end{cases}$$

corresponding to the first register of the state ψ_3 resulting from Step 3 above, which is a uniform superposition over the coset cH. The Fourier transform of such a function is

$$\widehat{\varphi_c}(\rho) = \sqrt{\frac{d_\rho}{|G||H|}}\, \rho(c) \cdot \sum_{h \in H} \rho(h).$$

Note, as above, that $\widehat{\varphi_c}(\rho)$ is a $d_\rho \times d_\rho$ matrix.

For any subgroup H, the sum $\sum_h \rho(h)$ is precisely $|H|$ times a projection operator (see, e.g., [13]); we write

$$\sum_h \rho(h) = |H|\, \pi_H(\rho).$$

With this notation, we can express $\widehat{\varphi_c}(\rho)$ as $\sqrt{n_\rho}\, \rho(c) \cdot \pi_H(\rho)$, where $n_\rho = d_\rho |H|/|G|$. For a $d \times d$ matrix M, we let $\|M\|$ denote the matrix norm given by

$$\|M\|^2 = \mathbf{tr}\left(M^\dagger M\right) = \sum_{ij} |M_{ij}|^2,$$

where $M^\dagger$ denotes the conjugate transpose of M. Then the probability that we observe the representation ρ is

$$\begin{aligned} \|\widehat{\varphi_c}(\rho)\|^2 &= \left\|\sqrt{n_\rho}\, \rho(c)\, \pi_H(\rho)\right\|^2 \\ &= n_\rho \left\|\pi_H(\rho)\right\|^2 \\ &= n_\rho\, \mathbf{rk}\, \pi_H(\rho), \end{aligned} \tag{1}$$

where $\mathbf{rk}\, \pi_H(\rho)$ denotes the rank of the projection operator $\pi_H(\rho)$. See [13] for more discussion.

1.3. Weak vs. strong sampling and the choice of basis. Hallgren, Russell, and Ta-Shma [13] show that by measuring only the *names* of representations—the so-called *weak standard method* in the terminology of [10]—it is possible to reconstruct normal subgroups (and thus solve the HSP for *Hamiltonian groups*, all of whose subgroups are normal). More generally, this method reconstructs the *normal core* of a subgroup, i.e., the intersection of all its conjugates. On the other hand, they show that this is insufficient to solve Graph Automorphism, since even in an information-theoretic sense this method cannot distinguish between the trivial subgroup of S_n and subgroups of order 2 consisting of the identity and an involution.

Therefore, in order to solve the HSP for nonabelian groups, we need to measure not just the name of the representation we are in but also the row and column. In order for this measurement to be well defined, we need to choose a basis for $\mathrm{U}(d_\rho)$ for each ρ. Grigni, Schulman, Vazirani, and Vazirani [10] call this the *strong standard method*. They show that if we measure using a uniformly *random* basis, then trivial and nontrivial subgroups are still information-theoretically indistinguishable. However, they leave open the question of whether the strong standard method with a clever choice of basis, rather than a random one, allows us to solve the HSP in nonabelian groups, yielding an algorithm for Graph Automorphism.

Indeed, from a computational perspective the representation theory of a finite group G does distinguish certain "preferred" bases, those which give the matrices $\rho(g)$ unusually structured or sparse form. In particular, Moore, Rockmore, and Russell [20] showed that so-called *adapted bases* yield highly efficient algorithms for the quantum Fourier transform.

1.4. Contributions of this paper. As stated above, [13] and [10] leave an important open question, namely, whether there are cases where the strong standard method, with the proper choice of basis, offers an advantage over a simple abelian transform or the weak standard method. We settle this question in the affirmative. Our results deal primarily with the *q-hedral* groups, i.e., semidirect products of the form $\mathbb{Z}_q \ltimes \mathbb{Z}_p$, where $q \mid (p-1)$, and in particular the *affine* groups $A_p \cong \mathbb{Z}_p^* \ltimes \mathbb{Z}_p$.

We begin in section 3 by focusing on full reconstructibility. We define the *hidden conjugate problem* (HCP) as follows: given a group G, a nonnormal subgroup H, and a function which is promised to be constant on the cosets of some conjugate $H^b = bHb^{-1}$ of H (and distinct on distinct cosets), determine the subgroup H^b by finding an element $c \in G$ so that $H^c = H^b$. We adopt the above classification (fully, information-theoretically, quantum information-theoretically) for this problem in the natural way. Then we show that given a subgroup of sufficiently small (but still exponentially large) index, hidden conjugates in A_p are fully reconstructible (Theorem 1). This almost immediately implies that, for prime $q = (p-1)/\text{polylog}(p)$, subgroups of the q-hedral groups $\mathbb{Z}_q \ltimes Z_p$ are fully reconstructible (Theorem 2).

Section 4 concerns itself with information-theoretic reconstructibility. We generalize the results of Ettinger and Høyer on the dihedral group and show that hidden conjugates of any subgroup are information-theoretically reconstructible in the affine groups and, more generally, the q-hedral groups for all q (Theorem 3). We then show that we can identify the order, and thus the conjugacy class, of a hidden subgroup, and this implies that all subgroups of the affine and q-hedral groups are information-theoretically reconstructible (Theorem 5).

The results of sections 3 and 4 rely crucially on measuring the high-dimensional representations of the affine and q-hedral groups in a well-chosen basis, namely an *adapted* basis that respects the group's subgroup structure. We show in section 5 that we lose information-theoretic reconstructibility if we measure in a *random* basis instead. Specifically, we need an exponential number of measurements to distinguish conjugates of small subgroups of A_p. This establishes for the first time that the strong standard method is indeed stronger than measuring in a random basis: some bases provide much more information about the hidden subgroup than others.

For some nonabelian groups, the HSP can be solved with a "forgetful" approach, where we erase the group's nonabelian structure and perform an abelian Fourier transform instead. In section 6 we show that this is not the case for the affine groups. Specifically, if we treat A_p as a direct product rather than a semidirect one, its conjugate subgroups become indistinguishable.

As an application, in section 7 we consider *hidden shift* problems. In the setting we consider, one must reconstruct a "hidden shift" $s \in \mathbb{Z}_p$ from an oracle $f_s(x) = f(x-s)$, where f is any function that is constant on the (multiplicative) cosets of a known multiplicative subgroup of $\mathbb{Z}_p^*$. These functions have been studied in some depth for their pseudorandom properties, and several instances have been suggested as cryptographically strong pseudorandom generators. By associating f_s with its isotropy subgroup, and using our reconstruction algorithm to find that subgroup, we give an efficient quantum algorithm for the hidden shift problem in the case where $f(x)$ is a function of x's multiplicative order mod r for some $r = \text{polylog}(p)$. This generalizes the work of van Dam, Hallgren, and Ip [4], who give an algorithm for hidden shift problems in the case where f is precisely a multiplicative character.

Finally, in section 8 we show that the set of groups for which the HSP can be solved in polynomial time has the following closure property: if $\mathcal{H} = \{H_n\}$ is a family of groups for which we can efficiently solve the HSP and $\mathcal{K} = \{K_n\}$ is a family of groups for which $|K_n| = \text{polylog}(|H_n|)$, we can also efficiently solve the HSP for the family $\{G_n\}$, where each G_n is any extension of K_n by H_n. This subsumes the results of [13] on Hamiltonian groups, and also those of [15] on groups with commutator subgroups of polynomial size.

We note that subsequent to this work, Bacon, Childs, and van Dam [1] found additional algorithms for the HSP in the affine groups. Their approach uses the "pretty good measurement," which they showed is optimal for certain cases of the HSP (see also [21]). Their work extends to a number of other group families, such as the Heisenberg groups.

2. The affine and *q*-hedral groups. Let A_p be the *affine group*, consisting of ordered pairs $(a, b) \in \mathbb{Z}_p^* \times \mathbb{Z}_p$, where p is prime, under the multiplication rule $(a_1, b_1) \cdot (a_2, b_2) = (a_1a_2, b_1 + a_1b_2)$. A_p can be viewed as the set of affine functions $f_{(a,b)} : \mathbb{Z}_p \to \mathbb{Z}_p$ given by $f_{(a,b)} : x \mapsto ax + b$ where multiplication in A_p is given by function composition. Structurally, A_p is a semidirect product $\mathbb{Z}_p^* \ltimes \mathbb{Z}_p \cong \mathbb{Z}_{p-1} \ltimes \mathbb{Z}_p$. Its subgroups can be described as follows:

- Let $N \cong \mathbb{Z}_p$ be the normal subgroup of size p consisting of elements of the form $(1, b)$. Geometrically, this is the set of affine functions with slope 1.
- Let $H \cong \mathbb{Z}_p^* \cong \mathbb{Z}_{p-1}$ be the nonnormal subgroup of size $p - 1$ consisting of the elements of the form $(a, 0)$. Geometrically, this is the set of lines passing through the origin.
- For each $b \in \mathbb{Z}_p$, the conjugate subgroup $H^b = (1, b) \cdot H \cdot (1, -b)$ consists of elements of the form $(a, (1 - a)b)$. In the action on $\mathbb{Z}_p$, H^b is the stabilizer of b; geometrically, H^b is the set of lines intersecting the diagonal at (b, b).
- If $a \in \mathbb{Z}_p^*$ has order q, where q divides $p - 1$, let $N_q \cong \mathbb{Z}_q \ltimes \mathbb{Z}_p$ be the normal subgroup consisting of all elements of the form (a^t, b). Geometrically, N^q is the set of lines whose slope is a power of a.
- Similarly, if $a \in \mathbb{Z}_p$ has order q, let H_q be the nonnormal subgroup $H_q = \langle (a, 0) \rangle$ of size q. Then H_q consists of the elements of the form $(a^t, 0)$, and its conjugates $H_q^b = (1, b) \cdot H_q \cdot (1, -b)$ consist of the elements of the form $(a^t, (1-a^t)b)$. Geometrically, these are the subsets of H and H^b, respectively, consisting of lines whose slope is a power of a.

Construction of the representations of A_p requires that we fix a generator γ of $\mathbb{Z}_p^*$. Define $\log : \mathbb{Z}_p^* \to \mathbb{Z}_{p-1}$ to be the isomorphism $\log \gamma^t = t$. Let ω_p denote the pth root of unity $e^{2\pi i/p}$. Then A_p has $p - 1$ one-dimensional representations σ_s, namely the representations of $\mathbb{Z}_p^* \cong \mathbb{Z}_{p-1}$ given by $\sigma_t((a, b)) = \omega_{p-1}^{t \log a}$. In addition, A_p has one $(p - 1)$-dimensional representation ρ given by

$$\rho((a, b))_{j,k} = \begin{cases} \omega_p^{bj}, & k = aj \bmod p, \\ 0 & \text{otherwise}, \end{cases} \quad 1 \le j, k < p, \tag{2}$$

where the indices i and j are elements of $\mathbb{Z}_p^*$. See [24, section 8.2] for a more detailed discussion.

Similarly, given a prime p and a divisor q of $p-1$, we consider the *q-hedral groups*, namely semidirect products $\mathbb{Z}_q \ltimes \mathbb{Z}_p$. These embed in A_p in a natural way, namely, as the normal subgroups N_q defined above. The *dihedral* groups are the special case where $q = 2$.

The representations of $\mathbb{Z}_q \ltimes \mathbb{Z}_p$ include the q one-dimensional representations of $\mathbb{Z}_q$ given by $\sigma_\ell((a^t, b)) = \omega_q^{\ell t}$ for $\ell \in \mathbb{Z}_q$ and the $(p-1)/q$ distinct q-dimensional representations ρ_k given by

$$\rho_k((a^u, b))_{s,t} = \begin{cases} \omega_p^{ka^s b}, & t = s + u \bmod q, \\ 0 & \text{otherwise} \end{cases}$$

for each $0 \leq s, t < q$. Here k ranges over the elements of $\mathbb{Z}_p^*/\mathbb{Z}_q$, or, to put it differently, k takes values in $\mathbb{Z}_p^*$, but ρ_k and $\rho_{k'}$ are isomorphic if k and k' are in the same coset of $\langle a \rangle$.

The representations of the affine and q-hedral groups are related as follows. The restriction of the $(p-1)$-dimensional representation ρ of A_p to N_q is reducible and is isomorphic to the direct product of the ρ_k. Moreover, if we measure ρ in a *Gel'fand–Tsetlin* basis such as (2) which is *adapted* to the tower of subgroups

$$A_p > N_q > \{1\},$$

then ρ becomes block-diagonal, with $(p-1)/q$ blocks of size q, and these blocks are exactly the representations ρ_k of N_q. (See [20] for an introduction to adapted bases and their uses in quantum computation.) We will use this fact in sections 4 and 5 below.

The affine and q-hedral groups are *metacyclic* groups, i.e., extensions of a cyclic group $\mathbb{Z}_p$ by a cyclic group $\mathbb{Z}_q$. In [14], Høyer shows how to perform the nonabelian Fourier transform over such groups (up to an overall phase factor) with a polynomial, i.e., polylog(p), number of elementary quantum operations.

3. Full reconstructibility. In this section we show that conjugates of sufficiently large subgroups of the affine groups are fully reconstructible in polynomial time. For some values of p and q, this allows us to completely solve the HSP for the q-hedral group $\mathbb{Z}_q \ltimes \mathbb{Z}_p$.

THEOREM 1. *Let p be prime and let q be a divisor of $p-1$ for which $(p-1)/q =$ polylog(p). Then the hidden conjugates H_q^b of H_q in A_p are fully reconstructible.*

Proof. First, consider the maximal nonnormal subgroup $H = \langle(\gamma, 0)\rangle$, where γ is a generator of $\mathbb{Z}_p^*$. Carrying out Steps 1 through 3 of the Fourier sampling procedure outlined in the introduction results in a state ψ_3 over the group G which is uniformly supported on a random left coset of the conjugate H^b. Using the procedure of [14], we now compute the quantum Fourier transform of this state over A_p in the basis (2). The associated projection operator is

$$\pi_{H^b}(\rho)_{j,k} = \frac{1}{p-1}\,\omega_p^{b(j-k)}$$

for $1 \leq j, k < p$. This is a circulant matrix of rank 1. More specifically, every column is some root of unity times the vector

$$(u_b)_j = \frac{1}{p-1}\,\omega_p^{bj},$$

$1 \leq j < p$. This is also true of $\rho(c) \cdot \pi_{H^b}(\rho)$; since $\rho(c)$ has one nonzero entry per column, left multiplying by $\rho(c)$ simply multiplies each column of $\pi_{H^b}(\rho)$ by a phase. Note that in this case

$$n_\rho = d_\rho |H|/|G| = (p-1)/p = 1 - 1/p,$$

and so by (1) we observe the $(p-1)$-dimensional representation ρ with overwhelming probability $1 - 1/p$.

Assuming that we observe ρ, we perform another change of basis: namely, we Fourier transform each column by left multiplying $\rho(cH)$ by the unitary matrix

$$Q_{\ell,j} = \frac{1}{\sqrt{p-1}} \, \omega_{p-1}^{-\ell j}.$$

In terms of quantum operations, we apply the quantum Fourier transform over $\mathbb{Z}_{p-1}$ to the row register, while leaving the column register unchanged. We can now infer b by measuring the frequency ℓ. Specifically, we observe a given value of ℓ with probability

$$(3) \qquad P(\ell) = \left| \frac{1}{p-1} \sum_{j=1}^{p-1} \omega_p^{bj} \omega_{p-1}^{-\ell j} \right|^2 = \frac{1}{(p-1)^2} \left| \sum_{j=1}^{p-1} \mathrm{e}^{2i\theta j} \right|^2 = \frac{1}{(p-1)^2} \frac{\sin^2 (p-1)\theta}{\sin^2 \theta},$$

where

$$\theta = \left(\frac{b}{p} - \frac{\ell}{p-1} \right) \pi.$$

Now note that for any b there is an ℓ such that $|\theta| \leq \pi/(2(p-1))$. Since

$$(2x/\pi)^2 \leq \sin^2 x \leq x^2$$

for $|x| \leq \pi/2$, this gives $P(\ell) \geq (2/\pi)^2$.

Recall that the probability that we observed the $(p-1)$-dimensional representation ρ in the first place is $n_\rho = 1 - 1/p$. Thus if we measure ρ, the column, and then ℓ and then guess that b minimizes $|\theta|$, we will be correct with constant probability. This can be boosted to high probability, i.e., $1 - o(1)$, by repeating the experiment a polynomial number of times. Alternately, we can use a POVM rather than the von Neumann measurement presented here and obtain the correct value of b with high probability in a single measurement [1].

Now consider the more general case, where the hidden subgroup is a conjugate of the subgroup H_q of order q. For convenience, let $\langle (a, 0) \rangle$ be a generator for H_q; then a given conjugate H_q^b consists of the elements of the form $(a^t, (1 - a^t)b)$. We have

$$\pi_{H_q^b}(\rho)_{j,k} = \frac{1}{q} \begin{cases} \omega_p^{b(j-k)}, & k = a^t j \text{ for some } t, \\ 0 & \text{otherwise} \end{cases}$$

for $1 \leq j, k < p$. In other words, the nonzero entries are those for which j and k lie in the same coset of $\langle a \rangle \subset \mathbb{Z}_p^*$. The rank of this projection operator is thus the number of cosets equal to the index $(p-1)/q$ of $\langle a \rangle$ in $\mathbb{Z}_p^*$. Since n_ρ is now q/p, we again observe ρ with probability

$$n_\rho \, \mathbf{rk} \, \pi_{H_q}(\rho) = (p-1)/p = 1 - 1/p.$$

Following the same procedure as before, we carry out a partial measurement on the columns of ρ and then Fourier transform the rows. After changing the variable of summation from t to $-t$ and adding a phase shift of $\mathrm{e}^{-i\theta(p-1)}$ inside the $|\cdot|^2$, we obtain the probability that we observe a frequency ℓ conditional on finding ourselves in the kth column:

$$P(\ell) = \left| \frac{1}{\sqrt{q(p-1)}} \sum_{t=0}^{q-1} \omega_p^{b(a^t k \bmod p)} \omega_{p-1}^{-\ell(a^t k \bmod p)} \right|^2 = \frac{1}{q(p-1)} \left| \sum_{t=0}^{q-1} \mathrm{e}^{2i\theta(a^t k \bmod p)} \right|^2 . \tag{4}$$

Now note that the terms in the sum are of the form $\mathrm{e}^{i\phi}$, where (assuming without loss of generality that θ is positive)

$$\phi \in [-\theta(p-1), \theta(p-1)].$$

If we again take ℓ so that $|\theta| \leq \pi/(2(p-1))$, then $\phi \in [-\pi/2, \pi/2]$ and all the terms in the sum have nonnegative real parts. We will obtain a lower bound on the real part of the sum by showing that a constant fraction of the terms have $\phi \in (-\pi/3, \pi/3)$ and thus have real part more than 1/2. This is the case whenever $a^t k \in (p/6, 5p/6)$, and so it is sufficient to prove the following lemma.

LEMMA 1. *Let a have order $q = p/\mathrm{polylog}(p)$ in $\mathbb{Z}_p^*$, where p is prime. Then at least $(1/3 - o(1))q$ of the elements in the coset $\langle a \rangle k$ are in the interval $(p/6, 5p/6)$.*

Proof. We will prove this using *Gauss sums*, which quantify the interplay between the characters of $\mathbb{Z}_p$ and the characters of $\mathbb{Z}_p^*$. In particular, Gauss sums establish bounds on the distribution of powers of a. Specifically, if a has order q in $\mathbb{Z}_p^*$, then for any integer $k \not\equiv 0 \bmod p$ we have

$$\sum_{t=0}^{q-1} \omega_p^{a^t k} = O(p^{1/2}) = o(p).$$

(See [17] and Appendix A.)

Now suppose s of the elements x in $\langle a \rangle k$ are in the set $(p/6, 5p/6)$, for which $\mathrm{Re}\, \omega_p^x \geq -1$, and the other $q - s$ elements are in $[0, p/6] \cup [5p/6, p)$, for which $\mathrm{Re}\, \omega_p^x \geq 1/2$. Thus we have

$$\mathrm{Re} \sum_{t=0}^{q-1} \omega_p^{a^t k} \geq \ (q/2) - \ (3s/2).$$

If $s \leq (1/3 - \epsilon)q$ for any $\epsilon > 0$, this is $\Theta(q)$, a contradiction. ∎

Now that we know that a fraction $1/3 - \epsilon$ of the terms in (4) have real part at least 1/2 and the others have real part at least 0, we can take $\epsilon = 1/12$ (say) and write

$$P(\ell) \geq \frac{1}{q(p-1)} \left(\frac{q}{8}\right)^2 = \frac{1}{64} \frac{q}{p-1} = \frac{1}{\mathrm{polylog}(p)}.$$

Thus we observe the correct frequency with polynomially small probability, and this can be boosted to high probability by a polynomial number of repetitions. ∎

As we will now show, Theorem 1 implies that we can completely solve the HSP for certain q-hedral groups.

THEOREM 2. *Let p and q be prime with $q = (p-1)/\mathrm{polylog}(p)$. Then subgroups of the q-hedral group $\mathbb{Z}_q \ltimes \mathbb{Z}_p$ are fully reconstructible.*

Proof. First, note that we can fully reconstruct H if it is nontrivial and normal. We do this by reconstructing the normal core of H,

$$C(H) = \bigcap_{\gamma \in G} \gamma H \gamma^{-1},$$

using the techniques of [13] (the weak standard method). The q-hedral groups have the special property that nonnormal subgroups contain no nontrivial normal subgroups; in particular, if H is nonnormal, then $C(H)$ is the trivial subgroup. Thus by reconstructing $C(H)$, we either learn that $H = C(H)$ or learn that H is either trivial or nonnormal. Furthermore, if H is trivial, we will learn this by checking our reconstruction against the oracle f and finding that it is incorrect. Therefore, it suffices to consider the nonnormal subgroups.

If q is prime, then the nonnormal subgroups of $\mathbb{Z}_q \ltimes \mathbb{Z}_p$ are all conjugate to a single subgroup $K \cong \mathbb{Z}_q$, as any such subgroup has the form $\{(a, (1-a)z) \mid a \in \mathbb{Z}_q < \mathrm{Aut}(\mathbb{Z}_p) \cong \mathbb{Z}_p^*\}$; in this case the HSP reduces to the HCP for K. While one can construct a proof similar to that of Theorem 1 directly for the q-hedral groups, it is convenient to embed them in A_p using the isomorphisms $N_q \cong \mathbb{Z}_q \ltimes \mathbb{Z}_p$ and $H_q \cong K$ and appeal to Theorem 1.

Now suppose we have an oracle $f : \mathbb{Z}_q \times \mathbb{Z}_p \to S$. We extend this to an oracle f' on A_p as follows. Choose a generator $\gamma \in \mathbb{Z}_p^*$ and one of the $q-1$ elements $a \in \mathbb{Z}_p^*$ of order q, and let

$$f' : A_p \to S \times \langle a \rangle,$$

where

$$f'((a,b)) = \left(f\left(\left(\left\lfloor \frac{\log a}{(p-1)/q} \right\rfloor, b \right) \right), a^q \right),$$

recalling that $\log \gamma^t = t$. The second component of f' serves to distinguish the cosets of N_q from each other, while the first component maps each coset of N_q to $\mathbb{Z}_q \ltimes \mathbb{Z}_p$ with the element of $\mathbb{Z}_q$ written additively, rather than multiplicatively. (This last step is not strictly necessary—after all, we could have written the elements of A_p in additive form in the first place—but it can be carried out with Shor's algorithm for the discrete logarithm [25].) This reduces the HCP for K (and therefore the HSP) on $\mathbb{Z}_q \ltimes \mathbb{Z}_p$ to the HCP for H_q on A_p, completing the proof. □

As an example of Theorem 2, if q is a *Sophie Germain* prime, i.e., one for which $p = 2q + 1$ is also a prime, we can completely solve the HSP for $\mathbb{Z}_q \ltimes \mathbb{Z}_p$.

4. Information-theoretic reconstructibility. In this section, we show that *all* subgroups of the affine and q-hedral groups, regardless of their size, are information-theoretically reconstructible. We start by considering the HCP for subgroups H_q in A_p. Then in Theorem 5 we show that we can identify the conjugacy class of a hidden subgroup and therefore the subgroup itself. This generalizes the results of Ettinger and Høyer [6], who show information-theoretic reconstructibility for the dihedral groups, i.e., the case $q = 2$.

THEOREM 3. *Let p be prime and let q divide $p-1$. Then the hidden conjugates of H_q in A_p are information-theoretically reconstructible.*

Proof. Suppose $a \in \mathbb{Z}_p^*$ has order q. Recall that H_q and its conjugates H_q^b are maximal in the subgroup $N_q \cong \mathbb{Z}_q \ltimes \mathbb{Z}_p$. We wish to show that there is a measurement whose outcomes, given two distinct values of b, have large (i.e., $1/\text{polylog}(p)$) total variation distance. First, we perform a series of partial measurements as follows:

(i) Measure the name of the representation of A_p. If this is not ρ, try again. Otherwise, continue.

(ii) Measure the name of the representation ρ_k of N_q inside ρ.

(iii) Measure the column of ρ_k.

(iv) Perform a POVM with q outcomes, in each of which the row $s \in \mathbb{Z}_q$ is u or $(u+1) \bmod q$.

As in Theorem 1, we measure the $(p-1)$-dimensional representation of A_p in a chosen basis. Recall that in the adapted basis (2) the restriction of ρ to N_q is block-diagonal, where the $(p-1)/q$ blocks are the q-dimensional representations ρ_k of N_q. Therefore, the projection operator $\pi_{H_q^b}(\rho)$ is block-diagonal, and each of its blocks is one of the projection operators $\pi_{H_q^b}(\rho_k)$. Summing ρ_k over $H_q^b = \{(a^t, (1-a^t)b)\}$ gives

$$\left(\pi_{H_q^b}(\rho_k)\right)_{s,t} = \frac{1}{q}\,\omega_p^{k(a^s-a^t)b}$$

for $0 \le s,t < q$. This is a matrix of rank 1, where each column (even after left multiplication by $\rho_k(c)$) is some root of unity times the vector $(u_k)_s = (1/q)\ \omega_p^{ka^s b}$. Since $n_\rho = q/p$, the probability that we observe a particular ρ_k is q/p. Since $\pi_{H_q^b}(\rho)$ has $(p-1)/q$ blocks of this kind, it has rank $(p-1)/q$, and the total probability that we observe ρ is $(p-1)/p = 1-1/p$ as before.

Then these four partial measurements determine k, remove the effect of the coset, and determine that the row has one of two values, u or $u+1$. Up to an overall phase we can write this as a two-dimensional vector:

$$\frac{1}{\sqrt{2}}\begin{pmatrix} \omega_p^{ka^u b} \\ \omega_p^{ka^{u+1} b} \end{pmatrix}.$$

Our only goal in doing this is to create a one-qubit state where the relative phase between the two basis vectors depends on the conjugate b. Moreover, the relative phase is multiplied by the irrep label k, which is uniformly random. As a result, the typical angle between the states corresponding to any two distinct cosets b, b' will be $\Omega(1)$, and a simple measurement yields a constant variation distance between them.

To make this precise, apply the Hadamard transform

$$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

and measure according to the resulting basis. The probability that we observe the first or second basis vector is then $\cos^2\theta$ and $\sin^2\theta$, respectively, where $\theta = (ka^u(a-1)b\pi)/p$. Now when we observe a q-dimensional representation ρ_k of N_k, the observed label k is uniformly distributed over $\mathbb{Z}_p^*/\mathbb{Z}_q$. Moreover, when we perform the POVM in step (iv) above, the u we observe is uniformly distributed over $\mathbb{Z}_q$. It follows that the coefficient $m = ka^u(u-1)$ is uniformly distributed over $\mathbb{Z}_p^*$. For any two distinct b, b', the total variation distance is then

$$\frac{1}{2(p-1)}\sum_{m\in\mathbb{Z}_p^*}\left(\left|\cos^2\frac{\pi mb}{p} - \cos^2\frac{\pi mb'}{p}\right| + \left|\sin^2\frac{\pi mb}{p} - \sin^2\frac{\pi mb'}{p}\right|\right).$$

This we rewrite

$$\frac{1}{p-1}\sum_{m\in\mathbb{Z}_p^*}\left|\cos^2\frac{\pi mb}{p}-\cos^2\frac{\pi mb'}{p}\right|$$

$$=\frac{1}{2(p-1)}\sum_{m\in\mathbb{Z}_p}\left|\cos\frac{2\pi mb}{p}-\cos\frac{2\pi mb'}{p}\right|$$

$$\geq\frac{1}{4(p-1)}\sum_{m\in\mathbb{Z}_p}\left(\cos\frac{2\pi mb}{p}-\cos\frac{2\pi mb'}{p}\right)^2$$

$$=\frac{p}{4(p-1)}>\frac{1}{4}.$$

(Adding the $m=0$ term contributes zero to the sum in the second line. In the third line we use the facts that $|x|\leq x^2/2$ for all $|x|\leq 2$, the average of $\cos^2 x$ is $1/2$, and the two cosines have zero inner product.)

Since the total variation distance between any two distinct conjugates is bounded below by a constant, we can distinguish between the p different conjugates with only $O(\log p)=\text{poly}(n)$ samples. Thus, hidden conjugates in A_p are information-theoretically reconstructible, completing the proof. □

By embedding the q-hedral groups in A_p as in Theorem 2, we can generalize Theorem 3 to the q-hedral groups (q not necessarily prime) as follows. Let p be prime, let q be a divisor of $p-1$, and let q' be a divisor of q. The q-hedral group $\mathbb{Z}_q\ltimes\mathbb{Z}_p$ has a unique normal subgroup $K\cong\mathbb{Z}_{q'}\ltimes\mathbb{Z}_p$, in which there is a maximal subgroup $H_{q'}$ of order q'. Moreover, all nonnormal subgroups of order q' are conjugates of $H_{q'}$.

THEOREM 4. *If p,q,q', and $H_{q'}$ are as above, then the hidden conjugates of $H_{q'}$ in $\mathbb{Z}_q\ltimes\mathbb{Z}_p$ are information-theoretically reconstructible.*

We now wish to information-theoretically reconstruct all subgroups of the affine and q-hedral groups. We can do this by using the fact that, except for normal subgroups, there is a unique conjugacy class of each order q', namely the conjugates of $H_{q'}$. Thus if we can determine the order of H, this determines its conjugacy class, and we can information-theoretically reconstruct which conjugate it is using, Theorem 3 or Theorem 4.

THEOREM 5. *Subgroups of the q-hedral groups $\mathbb{Z}_q\ltimes\mathbb{Z}_p$, including the affine groups A_p, are information-theoretically reconstructible.*

Proof. As in Theorem 2, we can (fully) reconstruct normal subgroups, and so it suffices to consider nonnormal subgroups H. As discussed above, if we can determine $|H|=q'$, then we know that it is one of the conjugates of $H_{q'}$, and we can reconstruct it information-theoretically using Theorem 3 or Theorem 4.

Now let the oracle be $f:\mathbb{Z}_q\ltimes\mathbb{Z}_p\to S$, and let $p_1^{\alpha_1}\ldots p_k^{\alpha_k}$ be the prime factorization of q, in which case $k\leq\sum_i\alpha_i=O(\log q)$. For each $i\in\{1,\ldots,k\}$ and each $\alpha\in\{0,\ldots,\alpha_i\}$, we will determine if $p_i^\alpha\mid|H|$, and taking the largest such α for each i gives the prime factorization of $|H|$.

To do this, for each $i\in[k]$ and $1\leq\alpha\leq\alpha_i$, let $\Upsilon_i^\alpha:\mathbb{Z}_q\ltimes\mathbb{Z}_p\to\mathbb{Z}_{q/p_i^\alpha}$ be the homomorphism given by

$$\Upsilon_i^\alpha:(a,b)\mapsto a^{p_i^\alpha}.$$

Then let

$$A_i^{\alpha_i} = \ker \Upsilon_i^\alpha = \{\gamma \in \mathbb{Z}_q \ltimes \mathbb{Z}_p \mid \gamma^{p_i^{\alpha_i}} = \mathbf{1}\},$$

where $\mathbf{1}$ denotes the identity element of $\mathbb{Z}_q \ltimes \mathbb{Z}_p$. $A_i^{\alpha_i}$ is the subgroup of $\mathbb{Z}_q \ltimes \mathbb{Z}_p$ consisting of all elements whose orders are a multiple of p_i^α. Now consider the function

$$f' : \mathbb{Z}_q \ltimes \mathbb{Z}_p \to S \times \mathbb{Z}_{q/p_i^\alpha}$$

given by

$$f'(\gamma) = (f(\gamma), \Upsilon_i^\alpha(\gamma))\,.$$

Observe that f' is constant (and distinct) on the left cosets of $H \cap A_i^\alpha$ and, furthermore, the subgroup $H \cap A_i^\alpha$ has order p^α if and only if p^α divides $|H|$. We may then determine if $H \cap A_i^\alpha$ has order p^α by assuming that it does, reconstructing H with Theorem 4 using f' as the oracle, and checking the result against the original oracle f. This allows us to determine the prime factorization of $|H|$ as desired, and the theorem follows. □

As in the dihedral case [6], we know of no polynomial-time algorithm which can reconstruct the most likely b from these queries. However, Kuperberg [18] gives a quantum algorithm for the HSP in the dihedral group, and more generally the hidden shift problem, that runs in subexponential ($e^{O(\log^{1/2} p)}$) time. Since we can reduce the HSP on $\mathbb{Z}_q \ltimes \mathbb{Z}_p$ to a hidden shift problem by focusing on two cosets of $\mathbb{Z}_p$, this algorithm applies to the q-hedral groups as well.

5. Random vs. adapted bases.

In Theorems 3 and 5, we measured the high-dimensional representation ρ in a specific basis which is adapted to the subgroup structure of A_p and the q-hedral groups. In contrast, we show in this section that if we measure ρ in a *random* basis instead, then for all but the largest values of q we need an exponential number of measurements in order to information-theoretically distinguish conjugate subgroups from each other.

THEOREM 6. *Let p be prime and let q be a divisor of $p-1$ for which $q < p^{1-\epsilon}$ for some $\epsilon > 0$. Let $P_b(v)$ be the probability that we observe a basis vector v in the Fourier basis if the hidden subgroup is H_q^b. If we measure ρ in a random basis, then for any two b, b', with high probability the ℓ_1 distance between these probability distributions is exponentially small. In particular, there exists $\beta > 0$, depending only on ϵ, such that*

$$\sum_v |P_b(v) - P_{b'}(v)| < p^{-\beta}.$$

Thus it takes an exponentially large number of measurements to distinguish the conjugates H_q^b and $H_q^{b'}$.

Proof. Since we observe the high-dimensional representation ρ with probability $1-1/p$, it suffices to consider the ℓ_1 distance summed over the $d_\rho = p-1$ basis vectors of ρ. In fact, we will show that $P_b(v)$ is exponentially close to the uniform distribution for all b.

Write $\pi = \pi_{H_q^b}(\rho)$. Then the probability that we observe a given basis vector v, conditioned on observing ρ, is

$$P_b(v) = \frac{1}{\mathbf{rk}\,\pi} |\pi \cdot v|^2\,.$$

If v is uniformly random with norm 1, the expectation of $|\pi \cdot v|_2^2$ is $(\mathbf{rk}\,\pi)/d_\rho$, and so the expectation of $P_b(v)$ is $1/d_\rho$. We will use the following lemma to show that when $\mathbf{rk}\,\pi$ is sufficiently large, $P_b(v)$ is tightly concentrated around this expectation.

LEMMA 2. *Let π be a projection operator of rank r in a d-dimensional space, and let v be a random d-dimensional vector of unit length. Then for all* $0 < \delta < 2$,

$$\Pr\left[\left||\pi \cdot v|_2^2 - \frac{r}{d}\right| > \delta\frac{r}{d}\right] < 4\mathrm{e}^{-r\delta^2/48}.$$

Proof. We use an argument similar to [10]. We can think of a random d-dimensional complex vector v as a random $2d$-dimensional real vector of the same length, and we can think of this in turn as

$$v_i = \frac{w_i}{\sum_{i=1}^{2d} w_i^2},$$

where the w_i are independent Gaussian variables with zero mean and unit variance. By choosing a basis in which π projects onto the first r (complex) components of v, we have

$$|\pi \cdot v|_2^2 = \frac{\sum_{i=1}^{2r} w_i^2}{\sum_{i=1}^{2d} w_i^2} = \frac{r}{d}\frac{(1/2r)\sum_{i=1}^{2r} w_i^2}{(1/2d)\sum_{i=1}^{2d} w_i^2}.$$

Now we use the following Chernoff bound, which can be derived from the moment generating function. For any t, we have

$$\Pr\left[\left|\left(\frac{1}{t}\sum_{i=1}^{t} w_i^2\right) - 1\right| > \epsilon\right] < 2\left[(1+\epsilon)^{1/2}\,\mathrm{e}^{-\epsilon/2}\right]^t.$$

For $|\epsilon| < 1/2$, we have $\ln(1+\epsilon) < \epsilon - \epsilon^2/3$, and this becomes

$$\Pr\left[\left|\left(\frac{1}{t}\sum_{i=1}^{t} w_i^2\right) - 1\right| > \epsilon\right] < 2\mathrm{e}^{-t\epsilon^2/6}. \tag{5}$$

Now, for any a, b, if $|a/b - 1| > \delta$, where $\delta < 2$, then either $|a - 1| > \delta/4$ or $|b - 1| > \delta/4$. Taking the union bound over these events where $a = (1/2r)\sum_{i=1}^{2r} w_i^2$ and $b = (1/2d)\sum_{i=1}^{2d} w_i^2$, setting $\epsilon = \delta/4$ and $t = 2r \le 2d$ in (5) gives the stated bound. □

Setting $d = d_\rho$ and $r = \mathbf{rk}\,\pi$, Lemma 2 and the union bound imply that, for any constant $A > \sqrt{48}$, if

$$\delta = A\sqrt{\frac{\log d_\rho}{\mathbf{rk}\,\pi}}, \tag{6}$$

then, with high probability, for all d_ρ basis vectors v we have

$$\left|P_b(v) - \frac{1}{d_\rho}\right| < \frac{\delta}{d_\rho}.$$

Summing over all v, this implies that the ℓ_1 distance between $P_b(v)$ and the uniform distribution is at most δ. Now recall that $\mathbf{rk}\ \pi = (p-1)/q$. If $q < p^{1-\epsilon}$, then $\mathbf{rk}\ \pi > p^{\epsilon}$, and (6) gives $\delta < p^{-\beta}$, where $\beta = \epsilon/3$, say. Since $P_b(v)$ is within δ of the uniform distribution for all b, doubling the constant A and using the triangle inequality completes the proof. □

Several remarks are in order. First, just as for the dihedral group, we can information-theoretically distinguish conjugate subgroups if we use a random basis *within* each q-dimensional block of ρ. The problem is that rather than having this block-diagonal structure, a random basis cuts across these blocks, mixing different "frequencies" ρ_k and canceling out the useful information. To be more precise, a random basis is not adapted to the subgroup structure of A_p; it does not "know" that ρ decomposes into a direct sum of the ρ_k when restricted to N_q.

Second, it is worth noting that for the values of q for which we have an algorithm for full (as opposed to information-theoretic) reconstruction, namely $q = p/\text{polylog}(p)$, a random basis works as well, since the ℓ_1 distance δ becomes $1/\text{polylog}(p)$. Based on the strong evidence from representation theory that some bases are much better for computation than others, we conjecture that, for some families of groups, adapted bases allow full reconstruction while random bases do not; but this remains an open question.

Third, while we focused above on distinguishing conjugate subgroups from each other, in fact our proof shows that if $q < p^{1-\epsilon}$, a random basis is incapable of distinguishing H_q from the *trivial* subgroup. In contrast, Theorems 3 and 5 show that an adapted basis allows us to do this.

6. Failure of the abelian Fourier transform.

In [6] the abelian Fourier transform over $\mathbb{Z}_2 \times \mathbb{Z}_p$ is used in a reconstruction algorithm for the dihedral groups. Using this sort of "forgetful" abelian Fourier analysis it is similarly information-theoretically possible to reconstruct subgroups of the q-hedral groups when q is small enough.

However, it does not seem possible to reconstruct subgroups of A_p using the abelian Fourier transform. In particular, we show in this section that if we think of the affine group as a direct product $\mathbb{Z}_p^* \times \mathbb{Z}_p$ rather than a semidirect product, then the conjugates of the maximal subgroup become indistinguishable. This is not surprising, since in an abelian group conjugates are identical by definition, but it helps illustrate that nonabelian HSPs require nonabelian approaches (most naturally, in our view, representation theory).

Let us consider the HCP for the maximal subgroup $H = \langle(\gamma, 0)\rangle$, where γ is a generator of $\mathbb{Z}_p^*$. In that case, the characters of $\mathbb{Z}_p^* \times \mathbb{Z}_p$ are simply $\rho_{k,\ell}(\gamma^t, b) = \omega_{p-1}^{kt}\omega_p^{\ell b}$. Summing these over $H_q = \{(z, (1-z)b \mid z \in \mathbb{Z}_p^*\}$ shows that we observe the character (k, ℓ) with probability

$$P(k,\ell) = \frac{1}{p\,(p-1)^2}\left|\sum_{t\in\mathbb{Z}_{p-1}} \omega_{p-1}^{kt}\omega_p^{\ell(1-\gamma^t)b}\right|^2$$

$$= \frac{1}{p\,(p-1)^2}\left|\sum_{x\in\mathbb{Z}_p^*} \omega_{p-1}^{k\log_\gamma x}\omega_p^{-\ell x b}\right|^2 .$$

This is the inner product of a multiplicative character with an additive one, which is another Gauss sum. In particular, assuming $b \neq 0$, we have

$$\begin{aligned} P(0,0) &= 1/p, \\ P(0,\ell \neq 0) &= 1/(p\,(p-1)^2), \\ P(k \neq 0, 0) &= 0, \\ P(k \neq 0, \ell \neq 0) &= 1/(p-1)^2 \end{aligned}$$

(see Appendix A). Since these probabilities do not depend on b, the different conjugates H^b with $b \neq 0$ are indistinguishable from each other. Thus it appears essential to use the nonabelian Fourier transform and the high-dimensional representations of A_p.

7. Hidden shift problems. Using the natural action of the affine group on $\mathbb{Z}_p$, we can apply our algorithm for the HCP studied above to a natural family of *hidden shift problems*. Specifically, let M be a multiplicative subgroup of $\mathbb{Z}_p^*$ of index $r > 1$, let S be some set of $r+1$ symbols, and let $f : \mathbb{Z}_p \to S$ be a function for which

$$f(x) = f(mx) \Leftrightarrow m \in M$$

for every $x \in \mathbb{Z}_p$. Observe that f is constant on the (multiplicative) cosets of M and takes distinct values on distinct cosets; to put it differently, $f(x)$ is an injective function of the multiplicative order of x mod r. Furthermore, $f(0) \neq f(x)$ for any nonzero x. The hidden shift problem associated with f is the problem of determining an unknown element $s \in \mathbb{Z}_p$ given oracle access to the shifted function

$$f_s(x) = f(x - s).$$

Such functions have remarkable pseudorandom properties, and have been proposed as pseudorandom generators for cryptographic purposes, where s acts as the seed to generate the sequence (see, e.g., [5]).

The special case when $f : \mathbb{Z}_p \to \mathbb{C}$ is a *Legendre symbol*, that is, a multiplicative character of $\mathbb{Z}_p^*$ extended to all of $\mathbb{Z}_p$ by setting $f(0) = 0$, was studied by van Dam, Hallgren, and Ip [4]. They give efficient quantum algorithms for these hidden shift problems for all characters of $\mathbb{Z}_p^*$. Their algorithms, however, make explicit use of the complex values taken by the character, whereas the algorithms we present here depend only on the symmetries of the underlying function f; in particular, in our case f can be an arbitrary injective function from a multiplicative character into a set S. On the other hand, their algorithms are efficient for characters of any order, while our algorithms require that r be at most polylogarithmic in p.

Returning to the general problem defined above, let $\mathcal{F}(\mathbb{Z}_p, S)$ denote the collection of S-valued functions on $\mathbb{Z}_p$. Note that the affine group A_p acts on the set $\mathcal{F}(\mathbb{Z}_p, S)$ by assigning $\alpha \cdot g(x) = g(\alpha^{-1}(x))$ for each $\alpha \in A_p$ and $g \in F(\mathbb{Z}_p, S)$. In particular, $f_s = (1, s) \cdot f$.

Now note that the isotropy subgroup of f, namely the subgroup of A_p that fixes the cosets of M, is precisely $H_q = \langle (a, 0) \rangle$, where $a \in \mathbb{Z}_p^*$ has order $q = (p-1)/r$. As we have $f_s = (1, s) \cdot f$, the isotropy subgroup of f_s is the conjugate subgroup $H_q^s = (1, s) \cdot H_q \cdot (1, -s)$. Now observe that if we define $F_s : A_p \to (\mathbb{Z}_p)^p$ so that $F_s(\alpha)$ is the p-tuple $(\alpha f_s(0), \alpha f_s(1), \ldots, \alpha f_s(p-1))$, then

$$F_s(\alpha) = F_s(\beta) \Leftrightarrow \alpha^{-1}\beta \in H_q^s; \tag{7}$$

i.e., F_s is constant precisely on the left cosets of H_q^s. Evidently, then, the solution to the HCP given by the oracle F_s determines the solution to the hidden shift problem given by f_s. Unfortunately, the *values* of the oracle F_s are of exponential size—we cannot afford to evaluate $\alpha f_s(x)$ for all $x \in \mathbb{Z}_p$. The same symmetry expressed in (7), however, can be obtained efficiently by selecting an appropriate subset $R = \{x_1, \dots, x_m\} \subset \mathbb{Z}_p$ and considering the oracle that samples αf_s on R, that is,

$$F_s^R(\alpha) = (\alpha f_s(x_1), \dots, \alpha f_s(x_m)).$$

Of course, we have $\alpha f_s = \beta f_s \Rightarrow F_s^R(\alpha) = F_s^R(\beta)$ regardless of R; the difficulty is finding a small set R for which $F_s^R(\alpha) = F_s^R(\beta) \Rightarrow \alpha f_s = \beta f_s$. We show below that a set of $O(\log p)$ elements selected uniformly at random from $\mathbb{Z}_p$ has this property with high probability.

Considering that $\alpha f_s(x) = \alpha \cdot (1, s) \cdot f(x)$, it suffices to show that if $\alpha f \neq \beta f$, then

$$\Pr_x[\alpha f(x) = \beta f(x)] \leq 1/2,$$

where x is selected uniformly at random in $\mathbb{Z}_p$. Note that for affine functions α and β and an element $x \in \mathbb{Z}_p$ for which $\beta^{-1}(x) \neq 0$,

$$\alpha f(x) = \beta f(x) \;\Leftrightarrow\; \frac{\alpha^{-1}(x)}{\beta^{-1}(x)} \in M.$$

The function $\alpha^{-1}(x)/\beta^{-1}(x)$ is a *fractional linear transform*, i.e., the ratio of two linear functions; such functions are the discrete analogues of the Möbius transformations in the complex plane. As in the complex case, the fractional linear transform $\gamma(x)/\delta(x)$ is a bijection on the projective space $\mathbb{Z}_p \cup \{\infty\}$ unless γ and δ share a root, or, equivalently, there is a scalar $z \in \mathbb{Z}_p^*$ such that $\gamma(x) = z\delta(x)$. If $\alpha^{-1}(x)/\beta^{-1}(x)$ is injective, we can immediately conclude that

$$\Pr_x[\alpha f(x) = \beta f(x)] \leq |M|/(p-1) = 1/r \leq 1/2.$$

Otherwise, $\alpha^{-1}(x)/\beta^{-1}(x) = z$ for some scalar z. Since $\alpha f \neq \beta f$, however, in this case we must have $z \in \mathbb{Z}_p^* \setminus M$. In particular, $f(zy) \neq f(y)$ for any $y \neq 0$, and so

$$\Pr_x[\alpha f(x) = \beta f(x)] = 1/p,$$

since this occurs only at the unique root x of $\alpha^{-1}(x) = 0$.

In either case, then, αf and βf differ on at least half the elements of $\mathbb{Z}_p$ whenever α and β belong to different cosets of H_q^s. It follows that if $R \subset \mathbb{Z}_p$ consists of m elements chosen independently and uniformly at random from $\mathbb{Z}_p$, we have

$$\Pr_R\left[\forall x \in R, \alpha f(x) = \beta f(x)\right] \leq 1/2^m$$

for any $\alpha, \beta \in A_p$ with $\alpha^{-1}\beta \notin H_q$. Taking a union bound over all pairs of left cosets of H_q,

$$\Pr_R\left[\exists \alpha, \beta \in A_p : \alpha^{-1}\beta \notin H_q \; \forall x \in R, \alpha f(x) = \beta f(x)\right] \leq \left(\frac{p(p-1)}{|H_q|}\right)^2 \frac{1}{2^m}.$$

Selecting $m = 5 \log p$ ensures that this probability is less than $1/p$.

Since we showed in section 3 that we can identify a hidden conjugate of H_q whenever H_q is of polylogarithmic index in $\mathbb{Z}_p^*$, this provides an efficient solution to the hidden shift problem so long as $p/q = \operatorname{polylog}(p)$.

8. Closure under extending small groups. In this section we show that for any polynomial-size group K and any H for which we can solve the HSP, we can also solve the HSP for any extension of K by H, i.e., any group G with $K \lhd G$ and $G/K \cong H$. (Note that this is more general than split extensions, i.e., semidirect products $H \ltimes K$.) This includes the case discussed in [13] of Hamiltonian groups, since all such groups are direct products (and hence extensions) by abelian groups of the quaternion group Q_8 [23]. It also includes the case discussed in [8] of groups with commutator subgroups of polynomial size, such as extra-special p-groups, since in that case $K = G'$ and $H \cong G/G'$ is abelian. Indeed, our proof is an easy generalization of that in [8].

Theorem 7. *Let H be a group for which hidden subgroups are fully reconstructible and K a group of polynomial size in* $\log |H|$. *Then hidden subgroups in any extension of K by H, i.e., any group G with $K \lhd G$ and $G/K \cong H$, are fully reconstructible.*

Proof. We assume that G and K are encoded in such a way that multiplication can be carried out in classical polynomial time. We fix some transversal $t(h)$ of the left cosets of K. First, note that any subgroup $L \subseteq G$ can be described in terms of (i) its intersection $L \cap K$, (ii) its projection $L_H = L/(L \cap K) \subseteq H$, and (iii) a representative $\eta(h) \in L \cap (t(h) \cdot K)$ for each $h \in L_H$. Then each element of L_H is associated with some left coset of $L \cap K$, i.e., $L = \bigcup_{h \in L_H} \eta(h) \cdot (L \cap K)$. Moreover, if S is a set of generators for $L \cap K$ and T is a set of generators for L_H, then $S \cup \eta(T)$ is a set of generators for L.

We can reconstruct S in classical polynomial time simply by querying the function h on all of K. Then $L \cap K$ is the set of all k such that $f(k) = f(1)$, and we construct S by adding elements of $L \cap K$ to it one at a time until they generate all of $L \cap K$.

To identify L_H, as in [8] we define a new function f' on H consisting of the unordered collection of the values of f on the corresponding left coset of K:

$$f'(h) = \{f(g) \mid g \in t(h) \cdot K\}.$$

Each query to f' consists of $|K| = \text{poly}(n)$ queries to f. The level sets of f' are clearly the cosets of L_H, and so we reconstruct L_H by solving the HSP on H. This yields a set T of generators for L_H.

It remains to find a representative $\eta(h)$ in $L \cap (t(h) \cdot K)$ for each $h \in T$. We simply query $f(g)$ for all $g \in t(h) \cdot K$ and set $\eta(h)$ to any g such that $f(g) = f(1)$. Since $|T| = O(\log |H|) = \text{poly}(n)$, this can be done in polynomial time, completing the proof. □

Unfortunately, we cannot iterate this construction more than a constant number of times, since doing so would require a superpolynomial number of queries to f for each query of f'. If K has superpolynomial size, it is not clear how to obtain $\eta(h)$, even when H has only two elements. Indeed, this is precisely the difficulty with the dihedral group.

9. Conclusion and directions for further work. We have shown that the "strong standard method," applied with adapted bases, solves certain nonabelian HSPs in quantum polynomial time that cannot be solved using measurements in random bases or "forgetful" abelian approaches.

While we are still very far from an algorithm for HSP in the symmetric group S_n or for Graph Automorphism, a global understanding of the power of strong Fourier sampling remains an important goal. Perhaps the next class of groups to try beyond the affine and q-hedral groups are matrix groups such as $\text{PSL}_2(p)$, whose maximal subgroups are isomorphic to A_p, and which include one of the infinite families of finite simple groups.

Appendix A. Notes on exponential sums. The basic *Gauss sum* bounds the inner products of additive and multiplicative characters of $\mathbb{F}_p$, the finite field of prime cardinality p. Definitive treatments appear in [19, section 5] and [17]. Considering $\mathbb{F}_p$ as an additive group with p elements, we have p additive characters $\chi_s : \mathbb{F}_p \to \mathbb{C}$, for $s \in \mathbb{F}_p$, given by $\chi_s : z \mapsto \omega_p^{sz}$, where, as above, $\omega_p = \mathrm{e}^{2\pi i/p}$ is a primitive pth root of unity. Likewise considering the elements of $\mathbb{F}_p^* = \mathbb{F}_p \setminus \{0\}$ as a multiplicative group, we have $p-1$ characters $\psi_t : \mathbb{F}_p^* \to \mathbb{C}$, for $t \in \mathbb{F}_p^*$, given by $\psi_t : g^z \mapsto \omega_{p-1}^{tz}$, where $\omega_{p-1} = \mathrm{e}^{2\pi i/(p-1)}$ is a primitive $(p-1)$th root of unity and g is a multiplicative generator for the (cyclic) group $\mathbb{F}_p^*$.

With this notation the basic Gauss sum is the following.

THEOREM 8. *Let χ_s be an additive character and ψ_t a multiplicative character of $\mathbb{F}_p$. If $s \neq 0$ and $t \neq 1$, then*

$$\Big| \sum_{z \in \mathbb{F}_p^*} \chi_s(z)\, \psi_t(z) \Big| = \sqrt{p}.$$

Otherwise

$$\sum_{z \in \mathbb{F}_p^*} \chi_s(z)\psi_t(z) = \begin{cases} p-1 & \textit{if } s = 0,\ t = 1, \\ -1 & \textit{if } s = 0,\ t \neq 1, \\ 0 & \textit{if } s \neq 0,\ t = 1. \end{cases}$$

See [19, section 5.11] for a proof.

This basic result has been spectacularly generalized. In the body of the paper we require bounds on additive characters taken over multiplicative subgroups of $\mathbb{F}_p^*$. Such sums are discussed in detail in [17]. The specific bound we require is the following.

THEOREM 9. *Let χ_t be a nontrivial additive character of $\mathbb{F}_p$ and $a \in \mathbb{F}_p^*$ an element of multiplicative order q. Then*

$$\sum_{z=0}^{q-1} \chi_t(a^z) = \begin{cases} O(p^{1/2}) & \textit{if } q \geq p^{2/3}, \\ O(p^{1/4}q^{3/8}) & \textit{if } p^{1/2} \leq q \leq p^{2/3}, \\ O(p^{1/8}q^{5/8}) & \textit{if } p^{1/3} \leq q \leq p^{1/2}. \end{cases}$$

See [17, section 2] for a proof.

Note that in the body of the paper, we use $\mathbb{Z}_p$ to denote the additive group of integers modulo p and $\mathbb{Z}_p^*$ to denote the multiplicative group of integers modulo p.

Acknowledgments. We are grateful to Wim van Dam, Julia Kempe, Greg Kuperberg, Frederic Magniez, Martin Rötteler, and Miklos Santha for helpful conversations and to Sally Milius and Tracy Conrad for their support.

REFERENCES

[1] D. BACON, A. CHILDS, AND W. VAN DAM, *From optimal measurement to efficient quantum algorithms for the hidden subgroup problem over semidirect product groups*, in Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, 2005, pp. 469–478.

[2] R. BEALS, *Quantum computation of Fourier transforms over symmetric groups*, in Proceedings of the 29th Annual ACM Symposium on the Theory of Computing, 1997, pp. 48–53.

[3] E. Bernstein and U. Vazirani, *Quantum complexity theory (preliminary abstract)*, in Proceedings of the 25th Annual ACM Symposium on the Theory of Computing, 1993, pp. 11–20.

[4] W. van Dam, S. Hallgren, and L. Ip, *Quantum algorithms for some hidden shift problems*, in Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2003, pp. 489–498.

[5] I. B. Damgård, *On the randomness of Legendre and Jacobi sequences*, in Advances in Cryptology—CRYPTO '88, Lecture Notes in Comput. Sci. 403, Springer-Verlag, Berlin, 1990, pp. 163–172.

[6] M. Ettinger and P. Høyer, *On quantum algorithms for noncommutative hidden subgroups*, Adv. in Appl. Math., 25 (2000), pp. 239–251.

[7] M. Ettinger, P. Høyer, and E. Knill, *The quantum query complexity of the hidden subgroup problem is polynomial*, Inform. Process. Lett. 91 (2004), pp. 43–48.

[8] K. Friedl, G. Ivanyos, F. Magniez, M. Santha, and P. Sen, *Hidden translation and orbit coset in quantum computing*, in Proceedings of the 35th Annual ACM Symposium on Theory of Computing, 2003, pp. 1–9.

[9] W. Fulton and J. Harris, *Representation Theory: A First Course*, Grad. Texts in Math. 129, Springer-Verlag, London, 1991.

[10] M. Grigni, L. J. Schulman, M. Vazirani, and U. Vazirani, *Quantum mechanical algorithms for the nonabelian hidden subgroup problem*, in Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, 2001, pp. 68–74.

[11] L. Hales and S. Hallgren, *Quantum Fourier sampling simplified*, in Proceedings of the 31st Annual ACM Symposium on Theory of Computing, 1999, pp. 330–338.

[12] L. Hales and S. Hallgren, *An improved quantum Fourier transform algorithm and applications*, in Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science, 2000, pp. 515–525.

[13] S. Hallgren, A. Russell, and A. Ta-Shma, *Normal subgroup reconstruction and quantum computation using group representations*, in Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, 2000, pp. 627–635.

[14] P. Høyer, *Efficient Quantum Transforms*, http://arxiv.org/abs/quant-ph/9702028 (1997).

[15] G. Ivanyos, F. Magniez, and M. Santha, *Efficient quantum algorithms for some instances of the non-abelian hidden subgroup problem*, Internat. J. Found. Comput. Sci., 14 (2003), pp. 723–740.

[16] R. Jozsa, *Quantum Factoring, Discrete Logarithms and the Hidden Subgroup Problem*, http://arxiv.org/quant-ph/0012084 (2000).

[17] S. V. Konyagin and I. E. Shparlinski, *Character Sums with Exponential Functions and Their Applications*, Cambridge Tracts in Math. 136, Cambridge University Press, Cambridge, UK, 1999.

[18] G. Kuperberg, *A subexponential-time quantum algorithm for the dihedral hidden subgroup problem*, SIAM J. Comput., 35 (2005), pp. 170–188.

[19] R. Lidl and H. Niederreiter, *Finite Fields*, Encyclopedia Math. Appl. 20, Cambridge University Press, Cambridge, UK, 1997.

[20] C. Moore, D. Rockmore, and A. Russell, *Generic quantum Fourier transforms*, in Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2004, pp. 771–780.

[21] C. Moore and A. Russell, *For distinguishing conjugate hidden subgroups, the pretty good measurement is as good as it gets*, Quantum Inf. Comput., to appear.

[22] M. Roetteler and T. Beth, *Polynomial-Time Solution to the Hidden Subgroup Problem for a Class of Non-Abelian Groups*, http://arxiv.org/abs/quant-ph/9812070 (1998).

[23] J. Rotman, *An Introduction to the Theory of Groups*, Grad. Texts in Math. 148, Springer-Verlag, London, 1994.

[24] J.-P. Serre, *Linear Representations of Finite Groups*, Grad. Texts in Math. 42, Springer-Verlag, London, 1977.

[25] P. W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM J. Comput., 26 (1997), pp. 1484–1509.

[26] D. R. Simon, *On the power of quantum computation*, SIAM J. Comput., 26 (1997), pp. 1474–1483.

SIAM J. COMPUT.
Vol. 37, No. 3, pp. 959–976

EFFICIENT TESTING OF BIPARTITE GRAPHS FOR FORBIDDEN INDUCED SUBGRAPHS*

NOGA ALON†, ELDAR FISCHER‡, AND ILAN NEWMAN§

Abstract. Alon et. al. [N. Alon, E. Fischer, M. Krivelevich, and M. Szegedy, *Combinatorica*, 20 (2000), pp. 451–476] showed that every property that is characterized by a finite collection of forbidden induced subgraphs is ϵ-testable. However, the complexity of the test is *double-tower* with respect to $1/\epsilon$, as the only tool known to construct such tests uses a variant of Szemerédi's regularity lemma. Here we show that any property of *bipartite* graphs that is characterized by a finite collection of forbidden induced subgraphs is ϵ-testable, with a number of queries that is polynomial in $1/\epsilon$. Our main tool is a new "conditional" version of the regularity lemma for binary matrices, which may be interesting on its own.

Key words. property testing, graph algorithms, approximation, regularity lemma

AMS subject classifications. 05C85, 05C35, 68Q10

DOI. 10.1137/050627915

1. Introduction.

Property testing, first started in [6] and [17], deals with the following general question: Given a property P and an input which is assumed to come in the form of an oracle, how many queries to the input are required to distinguish between an input which satisfies P and an input which is ϵ-far (in the normalized Hamming distance) from any input that satisfies P? Property testing in general, and the investigation of graph testing that was started in [14], in particular, has become an active research area in recent years (see, for example, [14, 3, 8, 15, 1, 4] and the surveys [16, 9]). In particular, it was shown in [3] that every property that is characterized by a finite collection of forbidden induced subgraphs is ϵ-testable, that is, one can distinguish between graphs that satisfy it and graphs that are ϵ-far from satisfying it, with a number of queries that is bounded by a function of ϵ only, and is independent of the size of the input graph. However, the complexity of the test is *double-tower* with respect to $1/\epsilon$, as the only tool known to prove this testability is a variant of Szemerédi's regularity lemma.

More recently, Alon and Shapira [1, 4] initiated a study of those graph properties that are characterized by forbidden subgraphs and can be tested "very efficiently" in the sense that they can be tested with only $poly(1/\epsilon)$ many queries. In [1] it is shown that the property of not containing a given subgraph (where the subgraph is not necessarily induced) is testable with a number of queries polynomial in $1/\epsilon$ if and

*Received by the editors March 29, 2005; accepted for publication (in revised form) February 20, 2007; published electronically August 29, 2007. A preliminary (and weaker) version of these results formed part of [10].

http://www.siam.org/journals/sicomp/37-3/62791.html

†Schools of Mathematics and Computer Science, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv 69978, Israel and School of Mathematics, Institute for Advanced Study, Princeton, NJ 08540 (nogaa@tau.ac.il). This author's research was supported in part by a grant from the Israel Science Foundation, by the Hermann Minkowski Minerva Center for Geometry at Tel Aviv University, and by the Von Neumann Fund.

‡Faculty of Computer Science, Technion—Israel Institute of Technology, Haifa 32000, Israel (eldar@cs.technion.ac.il). This author's research was supported in part by grant 55/03 from the Israel Science Foundation.

§Department of Computer Science, University of Haifa, Haifa 31905, Israel (ilan@cs.haifa.ac.il). This author's research was supported in part by grant 55/03 from the Israel Science Foundation.

only if the forbidden subgraph is bipartite. In the context of testing digraphs for a forbidden structure, [4] contains a similar (but more complex) classification. The only known upper bounds for the cases where the number of queries is not polynomial are the tower (or worse) functions that result from Szemerédi's regularity lemma and its variants.

Here we concentrate on graph properties that are characterized by a finite family of forbidden induced subgraphs. For general graphs, the only known upper bound is the tower of towers; it was obtained from the proof in [3] that this is testable at all. We consider here the special case of *bipartite* input graphs and show, in contrast to the above, that any property of bipartite graphs that is characterized by a finite collection of forbidden induced subgraphs is ϵ-testable with a number of queries that is polynomial in $1/\epsilon$.

Our main tool is a new "conditional" version of the regularity lemma for binary matrices (Lemma 1.6 below), which may be interesting on its own. We combine this with some methods similar to those of [11] to obtain the desired result ([11] is an expanded version of the results from [10] about matrix-poset properties, while this paper expands the results from [10] about testing of bipartite graphs; the original bounds in [10] for bipartite graphs, while better than the previously known tower of towers, were not polynomial in $1/\epsilon$).

Our results are stated for graphs that are already given with a bipartition of their vertices (with the definition of a forbidden subgraph also relating to subgraphs with a compatible bipartition). However, in the case of bipartite input graphs whose bipartition is not given in advance (and general induced forbidden subgraphs), we can first use the approximate bipartition oracle given in [14] to reduce that setting to our setting.

We now note that the study of such bipartite graph properties is an extension of the poset model studied in [11], in which the testability of properties is related to the logical complexity of their description (for the purpose here a *model* is the language in which the properties are expressed, so a model is essentially identifiable with its family of expressible properties). In this case the poset is the 2-dimensional $n \times n$ grid, which as a poset is the product of two n-size total orders (lines). The language (syntax) includes the poset relation, the label unary relation (being labeled "1"), and in addition, the relations $row(x_1, x_2)$ which state that x_1 is on the same row as x_2, and similarly $col(x_1, x_2)$ for columns. $\forall$-properties in this model are properties that can be described by a finite formula over a fixed number of variables with only $\forall$-quantifiers in prenex normal form. Such properties would then correspond to exactly the properties that are characterized by a finite collection of forbidden submatrices (in a manner similar to what was done in [11] for the $\forall$-poset model). We call this model the "submatrix model." The submatrix model is closely related to a submodel of the (not always testable) $\forall\exists$-poset model, defined in [11].

The model "submatrix" includes some interesting properties. In particular, the permutation-invariant properties in it are tightly connected to bipartite graph properties that are characterized by a collection of forbidden induced subgraphs.

Definition 1.1. *For a finite collection F of $0/1$ matrices, we denote by $\mathcal{S}_F$ all $0/1$-matrices that do not contain as a submatrix any row and/or column permutation of a member of F.*

Observation 1.2. *Every bipartite graph property (where a bipartite graph is identified with its adjacency matrix in the usual way) that is characterized by a finite collection of forbidden induced subgraphs is equivalent to a property $\mathcal{S}_F$ for some finite set F of matrices. In addition, every $\mathcal{S}_F$-property in the "submatrix" model is* *equivalent to a bipartite graph property as above.*

It is important to note that here we discuss forbidden *induced* subgraphs. Not having a forbidden subgraph (rather than induced subgraph) is a monotone decreasing property. In this case, the test for the property is trivial, by density. For a large enough density, a Zarankiewicz (see [21], [13]) type theorem asserts that the answer "No" is correct (as the graph will have a large enough complete bipartite graph), while if the density is low then the answer is trivially "Yes," as the graph is close to the empty (edgeless) one. A thorough treatment of this case is found in [1]. The main result in the present paper is the following.

THEOREM 1.3. *Let F be a fixed finite collection of $0/1$ matrices. Property $\mathcal{S}_F$ is $(\epsilon, poly(\frac{1}{\epsilon}))$-testable for every $\epsilon > 0$, by a 2-sided error algorithm.*

The test above, however, is not only 2-sided but also very computation-intensive (despite this computation using only a relatively small set of queries as data). Using some additional tools we then derive a 1-sided error test which is also efficient in terms of its running time.

THEOREM 1.4. *Let F be a fixed finite collection of $0/1$ matrices. Property $\mathcal{S}_F$ is $(\epsilon, poly(\frac{1}{\epsilon}))$-testable for every $\epsilon > 0$, by a one sided error algorithm whose running time is polynomial in the time it takes to make the queries.*

The derivation of Theorem 1.4 from the main tool used in Theorem 1.3 is done in two stages, in sections 5 and 6. To present the test proving Theorem 1.3, we will need some machinery.

Let M be a 0/1-labeled, $n \times n$ matrix (to simplify notation we restrict ourselves to square matrices, but all arguments and theorems in this paper hold word-for-word for rectangular $n \times m$ matrices as well). We denote by $R(M)$ and $C(M)$ the set of rows and the set of columns of M, respectively. For an integer r, an *r-partition* of M is a partition of the set $R(M)$ into $r' \leq r$ parts $\{R_1, \ldots, R_{r'}\}$ and a partition of the set $C(M)$ into $r'' \leq r$ parts $\{C_1, \ldots, C_{r''}\}$. Each submatrix of the form $R_i \times C_j$ will be called a block (note that the coordinate sets defining the blocks do not necessarily consist of consecutive matrix coordinates). The weight of the (i, j) block is defined as $\frac{1}{n^2}|R_i||C_j|$. We also define similar weights for the R_i's and C_j's, e.g., $w(R_i) = \frac{1}{n}|R_i|$.

For a block B of a 0/1-matrix M and $\delta \geq 0$, we say that B is *δ-homogeneous* if all but a δ-fraction of its values are identical. If B is δ-homogeneous we call the value that appears in at least a $1-\delta$ fraction of the places the *δ-dominant* value of B. Note that this value is also α-dominant for any $\delta < \alpha < 1/2$. We say that a value is the *dominant* value of B if it is simply the majority value in B.

DEFINITION 1.5. *Let $\mathcal{P} = \{R_1, \ldots R_{r'}\} \times \{C_1, \ldots C_{r''}\}$ be an r-partition of M, and let $\delta > 0$. We say that $\mathcal{P}$ is a (δ, r)*-partition *if the total weight of the δ-homogeneous blocks is at least $1 - \delta$.*

The key result is that an input that does not admit some (δ, r)-partition can be rejected easily, because it will then contain many copies of every possible $k \times k$ matrix (including the forbidden ones) as submatrices.

LEMMA 1.6. *Let k be fixed. For every $\delta > 0$ and an $n \times n$, $0/1$-matrix M with $n > (k/\delta)^{O(k)}$, either M has a (δ, r)-partition for $r = r(\delta, k) \leq (k/\delta)^{O(k)}$, or for every $0/1$-labeled $k \times k$ matrix B, a $(g(\delta, k) \geq (\delta/k)^{O(k^2)})$-fraction of the $k \times k$ submatrices of M are B.*

This lemma allows us to reduce the testing problem to matrices that admit a (δ, r)-partition for certain δ, r; as for matrices that do not admit such partitions, the lemma asserts that querying a random submatrix will find a counterexample with sufficiently high probability. We note that the lemma is essentially a conditional version of Szemerédi's regularity lemma ([19]; see also [7, Chapter 7]), as a (δ, r)-partition is in particular a regular partition in the sense of Szemerédi of the corresponding bipartite graph. The improvement over directly using the regularity lemma is achieved because of this conditioning. The proof of the lemma will be presented in section 4.

We then construct a test for matrices admitting a (δ, r)-partition. This test will be very similar to the 2-sided boolean matrix poset test in [11]. However, the situation in the poset test is that the partition can be fixed in advance, while in our case there is the problem of "learning" enough of the partition by sampling. The main tool for doing so is Lemma 2.3 below. For stating it we need some more definitions, which are described in section 2 along with the framework of the proof of Theorem 1.3.

The plan of the paper is as follows. Section 2 includes some preliminaries, as well as a proof of Theorem 1.3 from two main lemmas—Lemma 1.6 above and Lemma 2.3 which is stated there. The lemmas themselves are proven in sections 4 and 3, respectively. We then turn to proving Theorem 1.4. This is done in two stages. First, a special case is proven in section 5, and then this case is used as a lemma in section 6 to prove the full result. In both stages we need the main tool that was used in the proof of Theorem 1.3, namely, Lemma 1.6. Finally, section 7 contains some concluding open problems.

2. Partitions, signatures, and Theorem 1.3. Assume that M has a (δ, r)-partition. We have no hope, of course, of finding it using $O(1)$ many queries, as we cannot even sample a single point from every matrix row. Hence, we will need to define the "high-level features" of the (δ, r)-partitions of M that can be detected by sampling.

In the following, whenever we refer to a δ-fraction of the members of a *weighted* set Q, we mean a subset Q', the total weight of whose members is δ (where we assume that the total weight of the members of Q is normalized to be 1). Let M be a matrix with a (δ, r)-partition $\mathcal{P}$ defined by the row partition $\{R_1, \ldots, R_s\}$ and the column partition $\{C_1, \ldots, C_t\}$, $s, t \leq r$. Then $\mathcal{P}$ naturally defines a high-level pattern which is an $s \times t$ matrix of the dominant labels of the blocks.

Definition 2.1. *Let $\mathcal{P}$ be a partition as above, and let P be a $0/1$-labeled, $s \times t$ matrix. A block $R_i \times C_j$ is called* δ-good *with respect to P if it is δ-homogeneous and its dominant label is $P_{i,j}$. P is called a* δ-pattern *of $\mathcal{P}$ if all but at most a δ-fraction of the weighted blocks in $\mathcal{P}$ are δ-good with respect to P.*

It is immediate from the definition that if a partition has a δ-good pattern of size $s \times t$, then it is a (δ, r)-partition with $r = \max\{s, t\}$. Conversely, if $\mathcal{P}$ is a (δ, r)-partition, then it has an $r \times r$ δ-pattern (by possibly introducing empty blocks). As the block sizes of a (δ, r)-partition need not be fixed, we will also need information about the weights of R_i and C_j, $(i, j) \in [s] \times [t]$.

Definition 2.2. *Let M be an $n \times n$ matrix with a (δ, r)-partition $\mathcal{P}$ defined by the row partition $\{R_1, \ldots, R_s\}$ and the column partition $\{C_1, \ldots, C_t\}$. Then a* δ-signature *of $\mathcal{P}$ is an $s \times t$, $0/1$-labeled matrix P and two sequences $\{\alpha_i\}_1^s, \{\beta_i\}_1^t$, where P is a δ-pattern of $\mathcal{P}$, and in addition $\sum_{i=1}^s |\frac{|R_i|}{n} - \alpha_i| \leq \delta$ and $\sum_{j=1}^t |\frac{|R_j|}{n} - \beta_j| \leq \delta$.*

Note that the signature of a partition is closed under permutations of rows and columns; namely, any row/column permutation of P with the respective permutations of $\{\alpha_i\}_1^s$ and $\{\beta_i\}_1^t$ is also a δ-signature of any matrix for which P is a δ-signature. Moreover, a signature of M is also a signature of all row/column permutations of M.

The signature of a partition has sufficient properties for constructing a test as we shall see in the proof of Theorem 1.3. The following also asserts that it can be approximated by sampling.

Lemma 2.3. *Let $\delta < 1/81$ and assume that an $n \times n$, 0/1-matrix M has a (δ, r)-partition. By making $q = (r/\delta)^{O(1)}$ many queries, a $26\delta^{1/6}$-signature of a $(16\delta^{1/6}, 10r^2/(4\delta^{1/3}) + 1)$-partition can be found, with success probability $\frac{3}{4}$.*

We note that a test for a much closer approximation of the original (δ, r)-partition can also be deduced from [14], with exponentially worse running time and query complexity. The proof of Lemma 2.3 is given in section 3. We end the discussion by showing that together with Lemma 1.6 this indeed implies a 2-sided error test.

Proof of Theorem 1.3. Assume that we want to ϵ-test M for a permutation-invariant collection of forbidden induced $k \times k$ submatrices. Blocks will now correspond to partition-blocks: Let $\delta = (\frac{\epsilon}{300})^6$, and let $g = g(\delta, k)$, $r = r(\delta, k)$ be those of Lemma 1.6. For $4/g = (k/\epsilon)^{O(k^2)}$ iterations, independently, we choose k random rows and k random columns of M and query all k^2 points in the $k \times k$ matrix that is defined by them. If we find a counterexample in the queried points we answer "No" and terminate the algorithm, and otherwise we continue. Let E_1 denote the event that M has no (δ, r)-partition and yet the algorithm continues. For inputs with a (δ, r)-partition, this event (by definition) never happens, while for other inputs, by Lemma 1.6, the probability of this event is bounded by $\frac{1}{12}$.

We now work under the assumption that M has a (δ, r)-partition and use the algorithm given in Lemma 2.3 to try finding an $\frac{\epsilon}{8}$-signature of an $(\frac{\epsilon}{8}, 10r^2/4(\frac{\epsilon}{300})^2 + 1)$-partition by sampling $(r/\delta)^{O(1)} = (k/\epsilon)^{O(k)}$ queries. Let P with $\{\alpha_i\}_1^s$ and $\{\beta_i\}_1^t$ be the signature obtained by the algorithm, and let E_2 be the event that it is not an $\frac{\epsilon}{8}$-signature of an $(\frac{\epsilon}{8}, 10r^2/4(\frac{\epsilon}{300})^2 + 1)$-partition of M. If M in fact did not have a (δ, r)-partition, then this event has the same probability as E_1 (which is bounded by $\frac{1}{12}$), and otherwise by Lemma 2.3 the probability of E_2 is bounded by $\frac{1}{4}$.

We now form an $n \times n$ matrix M_Q that represents our knowledge of M: We partition the rows of M_Q into s parts of weights $\{\alpha_i\}_1^s$ and the columns into t parts of weights $\{\beta_i\}_1^t$. For every block of P, we set every entry of the corresponding block of M_Q to have the same label as in P. Now, let $\mathcal{M}_{Q,\epsilon}$ be the set of all matrices that can be obtained from M_Q by changing at most $\epsilon n^2/2$ entries in any possible way. We check if any of the members of $\mathcal{M}_{Q,\epsilon}$ has the property $\mathcal{S}_F$. If there is such a member, the algorithm answers "Yes." Otherwise, if every member $\mathcal{M}_{Q,\epsilon}$ contains a permutation of a forbidden submatrix, then the answer is "No." Note that this last phase of the algorithm involves no additional queries and is just a computation phase.

To see that the algorithm is correct we first note that if a counterexample is found in the first phase of the algorithm, then the input M does not have the property with probability 1. Hence the algorithm can err only in the second phase.

We claim that unless E_2 happened the following hold: (a) some row/column permutation of M is a member of $\mathcal{M}_{Q,\epsilon}$, and (b) every two members of $\mathcal{M}_{Q,\epsilon}$ are of distance at most ϵn^2. Indeed, assume that the signature that has been found is an $\frac{\epsilon}{8}$-signature of an $(\frac{\epsilon}{8}, 10r^2/4(\frac{\epsilon}{300})^2 + 1)$-partition of M. Then M_Q can be obtained from M by changing at most an $\frac{\epsilon}{8}$-fraction of the entries in each $\frac{\epsilon}{8}$-good block, followed by changing any of the entries in the non–$\frac{\epsilon}{8}$-homogeneous blocks, and finally changing entries that are in strips around every block to compensate for the inaccuracy of the size sequences of the signature (whose sizes sum up to no more than $\frac{\epsilon}{8}$ for the rows and $\frac{\epsilon}{8}$ for the columns). The first two types of changes contribute at most an $\frac{\epsilon}{8}$-fraction of changes to the whole matrix each, and the last type contributes at most an $\frac{\epsilon}{4}$-fraction of changes. Thus M is at most $\epsilon n^2/2$-far from M_Q, and, in particular, M is in $\mathcal{M}_{Q,\epsilon}$. This proves (a), while (b) follows automatically from the definition of $\mathcal{M}_{Q,\epsilon}$ and the triangle inequality.

Hence, we may assume that with probability at least $\frac{3}{4}$ (which is the lower bound on E_2 not happening), the $\frac{\epsilon}{8}$-signature is computed correctly and (a) and (b) above are satisfied. We conclude that if M has the property then certainly some member of $\mathcal{M}_{Q,\epsilon}$ will have the property (as M itself is such a member by (a)), and thus the algorithm will accept. On the other hand, if M is more than ϵn^2-far from having the property, then no member of $\mathcal{M}_{Q,\epsilon}$ can have the property by (b).

Clearly the query complexity of the test is $O(k/\epsilon)^{O(k^2)}$, which for a fixed family F (and hence a fixed k) is polynomial in ϵ. □

The above test, while using only a constant number of queries, has a bad dependence of the calculation time on the input size (this can be alleviated somewhat, but in light of the following we omit the details). Unfortunately, this dependence is such that the automatic conversion by Alon of 2-sided tests to 1-sided ones, described in [15, Appendix D], will not work here. Instead we will go on a different route to show that a (δ, r)-partition of the matrix not only contains the necessary information about its farness from our property, but also implies the existence of many witnesses. But first, we turn back to the proofs of Lemmas 2.3 and 1.6.

3. (δ, r)-partitions, row similarity, and the proof of Lemma 2.3.

Our goal here is to show that by sampling $(r/\delta)^{O(1)}$ entries in M, one can detect the signature of a (δ', r')-partition, if a (δ, r)-partition exists. For this we need a representation of a partition in a "local" way, which is asserted by Claims 3.2 and 3.3. To do this, we relate the notion of a (δ, r)-partition to relative distances between rows and columns. For the rest of this section we assume that δ is smaller than 1/81.

For two vectors $u, v \in \{0,1\}^m$ let $\mu(u,v) = \frac{1}{m}|\{i|\ u_i \neq v_i\}|$; namely, $\mu(u,v)$ is the normalized Hamming distance between the two vectors. We will use the following definitions.

DEFINITION 3.1. *Let M be an $n \times n$ matrix. We set $E^R(\mu(r_i, r_j))$ to be the expected value of $\mu(r_i, r_j)$, where r_i, r_j are two rows of M chosen at random. Similarly let $E^C(\mu(c_i, c_j))$ denote the respective quantity where c_i, c_j are two columns chosen at random.*

*Given a set of vectors V (usually either the set of rows or the set of columns of M) and a partition $V_0, \dots, V_s$ of V, we say that the partition is a (δ, r)-*clustering of *V if $s \le r$, $|V_0| \le \delta|V|$, and for every $1 \le i \le r$ and $u, v \in V_i$ we have $\mu(u,v) \le \delta$.*

Finally, for a partition block B and a row u that intersects B, let $u|_B$ be the restriction of u to the columns in B.

There is a close correlation between (δ, r)-partitions of M and (δ, r)-clusterings of its rows and columns, as the following two claims show.

CLAIM 3.2. *Let M be a $0/1$, $m \times m$ matrix, and assume that M has a (δ, r)-partition. Then there exist a $(4\delta^{1/3}, r)$-clustering of the rows of M as well as a $(4\delta^{1/3}, r)$-clustering of the columns of M.*

CLAIM 3.3. *Let M be a $0/1$, $m \times m$ matrix, and assume that $\{R_0, \dots, R_s\}$ and $\{C_0, \dots, C_t\}$ are (δ^2, r)-clusterings, for $r = \max\{s, t\}$, of the set of rows and the set of columns, respectively. Then these clusterings also form a $(4\delta, r+1)$-partition of M.*

Moreover, for the above $R_0, \dots, R_s$ and $C_0, \dots, C_t$, a 4δ-signature for the partition is given by the sequences $\alpha_i = w(R_i)$, $i = 0, \dots, s$, $\beta_i = w(C_i)$, $i = 0, \dots, t$, and the $s \times t$ matrix P, where the (i,j) entry of P corresponds to the block $R_i \times C_j$ and its label is the dominant label of this block.

Before we prove the two claims we need two simple observations that in some sense correspond to the case "$r = 1$" of the claims.

OBSERVATION 3.4. *Let A be a $0/1$ matrix. If A is δ-homogeneous, then $E^R(\mu(r_i, r_j)) \leq 2\delta$ and $E^C(\mu(r_i, r_j)) \leq 2\delta$.*

Proof. As A is δ-homogeneous, we may assume without loss of generality that A contains less than a δ fraction of 0's. Hence, choosing two rows at random and picking a random place i in both, the probability that they are not both "1" in this place is at most 2δ. Thus the expectation of the fraction of the number of places where they differ is bounded by 2δ, and this expectation is exactly $E^R(\mu(r_i, r_j))$. The proof for $E^C(\mu(r_i, r_j))$ is analogous. □

OBSERVATION 3.5. *If A is a $0/1$ matrix such that $E^R(\mu(r_i, r_j)) < \delta$ and $E^C(\mu(c_i, c_j)) < \delta$, then A is 4δ-homogeneous.*

Proof. Assume on the contrary that A is not 4δ-homogeneous. This implies that when choosing two points from A independently and uniformly at random, with probability at least 4δ, they will not have the same label. This is also a lower bound on the fraction of the 2×2 submatrices that contain both 0's and 1's, as any two points with different labels can be extended to such a submatrix. On the other hand, if $E^R(\mu(r_i, r_j)) < \delta$, then with probability more than $1 - 2\delta$ both rows of a uniformly random 2×2 submatrix are identical, as this matrix can be expressed as choosing two random places from two random rows. By the same token, if $E^R(\mu(c_i, c_j)) < \delta$, then with probability more than $1 - 2\delta$ the two columns of a random 2×2 matrix are identical. Together these would have implied that less than a 4δ fraction of the 2×2 submatrices have both 0's and 1's, which is a contradiction. □

Proof of Claim 3.2. Assume that M has a (δ, r)-partition defined by the row partition $R_1, \ldots, R_s$ and the column partition $C_1, \ldots, C_t$, $s, t \leq r$. Assume that B is a δ-homogeneous block that contains the rows of R_i. Then by Observation 3.4, $E^R(u|_B, v|_B) \leq 2\delta$ for two rows chosen at random from R_i. For a non$-$ δ-homogeneous block, this expression is at most 1. Let $w_i = w(R_i) = |R_i|/m$, $i = 1, \ldots, s$, and let $E_i(\mu(u, v))$ be the expectation of $\mu(u, v)$, where u, v are two rows chosen uniformly at random from R_i. Then the above implies that $\Sigma_{i=1}^{r} w_i E_i(\mu(u, v)) \leq (1-\delta)2\delta + \delta \cdot 1 \leq 3\delta$, as this sum goes over all blocks and there is at least a $(1-\delta)$ fraction of 0/1-blocks contributing at most 2δ each.

Now this implies that the total weight of the R_i's for which $E_i(\mu(u, v)) \geq \delta^{2/3}$ is at most $3\delta^{1/3}$. Let R_0 be the union of all these R_i's. Let $R_1, \ldots, R_{r'}$ be all other R_i's, after renumbering. For every $i = 1, \ldots, r'$, by our assumption, $E_i(\mu(u, v)) < \delta^{2/3}$ for randomly chosen u, v, so there is an $r_i \in R_i$ for which for at least a $(1 - \delta^{1/3})$ fraction of the v's in R_i, $\mu(r_i, v) < \delta^{1/3}$. Hence if we define for $1 \leq i \leq r'$ the set $R'_i = \{v \in R_i | \mu(v, r_i) < \delta^{1/3}\}$ and then define $R'_0 = \bigcup_{i=1}^{r'} (R_i \setminus R'_i) \cup R_0$, we obtain that $R'_0, \ldots, R'_{r'}$ is indeed a $(4\delta^{1/3}, r)$-clustering for the rows of M. The proof for the existence of a clustering of the columns is analogous. □

Proof of Claim 3.3. By the assumptions of the claim, $|R_0| < \delta^2 n$. Also, for any $i \geq 1$ and any two rows $u, v \in R_i$, $\mu(u, v) \leq \delta^2$. Thus for $i = 1, \ldots, s$, $E_i(\mu(u, v)) \leq \delta^2$, where E_i is the expectation when u, v are chosen at random from R_i. Hence for the above partition into rows, $\Sigma_{i=0}^{s} \frac{|R_i|}{m} E_i(\mu(u, v)) \leq 2\delta^2$ (as for each $i > 1$ the corresponding term in this average is at most δ^2, and for $i = 0$ the weight of the term is at most δ^2). Similarly we get the analogous inequality for columns. Let $\mathcal{P}$ be the partition of M into blocks that is defined by the cross product of the two partitions above.

Recall that $\frac{|R_i|}{m}$, $\frac{|C_i|}{m}$ are the weights $w(R_i), w(C_i)$ of the corresponding sets. Also, for a block B, let $E_R(\mu(u|_B, v|_B))$, respectively, $E_C(\mu(u|_B, v|_B))$, be the expectation of $\mu(\cdot, \cdot)$ for two rows u, v, respectively, columns, chosen at random from B. By the law of complete probability, $\Sigma_{i=0}^{s} w(R_i) \cdot E_i(\mu(u,v)) = E_B(E_R(\mu(u|_B, v|_B)))$, where in the right-hand side the outer expectation is on blocks of $\mathcal{P}$ chosen according to their weights, and the inner expectation is on rows chosen at random in the block. Hence, the fact that $\Sigma_{i=0}^{s} w(R_i) E_i(\mu(u,v)) \leq 2\delta^2$ implies that the total weight of all blocks B for which $E_R(\mu(u|_B, v|_B)) > \delta$ is bounded by 2δ. By the same argument, for at most a 2δ fraction of the blocks $E_C(\mu(u|_B, v|_B)) > \delta$. Hence, for at least a $1-4\delta$ fraction of the blocks (weighted by the block weights), both $E_R(\mu(u|_B, v|_B)) \leq \delta$ and $E_C(\mu(u|_B, v|_B)) \leq \delta$. However, by Observation 3.5 above, each such block is 4δ-homogeneous, and hence at most a 4δ fraction of the blocks (measured by weights) are not 4δ-homogeneous. This implies that $\mathcal{P}$ is a $(4\delta, r+1)$-partition. Also, by definition, a pattern for this partition is any one that has, for each block, the $(1-4\delta)$-dominant label of this block if there is one, or an arbitrary value otherwise. Moreover, as α_i, β_i are the exact weights of the parts in the partition, we get a 4δ-signature for it by definition. □

We are now ready to present the testing algorithm that yields Lemma 2.3. We start with a trivial observation about approximating distances.

Claim 3.6. *Let $u, v \in \{0,1\}^n$, $\gamma < 1$. Choose randomly and independently (with repetitions) m elements of $[n]$, naming the resulting (multi)set $L = \{l_1, \dots, l_m\}$. Let $\tilde{\mu}(u,v) = \frac{1}{m}\sum_{k=1}^{m} |u(l_k) - v(l_k)|$, where $u(i)$ and $v(i)$ are the ith coordinates of u and v, respectively. Then $|\mu(u,v) - \tilde{\mu}(u,v)| \leq \gamma$ with probability at least $1 - 2exp(-\gamma^2 m)$.*

Proof. The proof is immediate by a Chernoff-type inequality (see, e.g., [5, Corollary A.1.7]). □

We next construct a testing algorithm for an approximate notion of clustering. Testing algorithms for clustering were already investigated in [2]; here we will use a simple self-contained proof for an algorithm that gives an approximation in a very weak sense.

Lemma 3.7. *There exists an approximate oracle algorithm that makes $(r/\delta)^{O(1)}$ bit queries (queries of one coordinate of one vector) to a set V of vectors over $\{0,1\}^n$, such that if V has a (δ, r)-clustering then the algorithm provides a $(4\delta, 10r^2/\delta)$-clustering of V as follows.*

The algorithm makes $(r/\delta)^{O(1)}$ queries in a preprocessing step, and with probability at least 0.9 provides a clustering oracle for V in the following sense: There exists a $(4\delta, 10r^2/\delta)$-clustering $V'_0, \dots, V'_t$ of V, such that for every specified $v \in V$ the algorithm can make $(r/\delta)^{O(1)}$ additional queries to provide an index $0 \leq i_v \leq t$, where it is guaranteed that for at least a $(1-4\delta)$ fraction of the vectors $v \in V$ the provided i_v will satisfy $v \in V_{i_v}$.

Proof. Suppose that $V_0, \dots, V_s$ is a (δ, r)-clustering of V. The algorithm starts by selecting uniformly at random $r' = 10r^2/\delta$ vectors $v_1, \dots, v_{r'}$ from V. With probability at least 0.95 (assuming that r is large enough) the situation is that for every $1 \leq i \leq r$ for which $|V_i| \geq \delta|V|/r$, we have picked at least one vector from V_i.

We now pick uniformly at random (with repetitions) $l = (10r' \log r')/\delta$ coordinates from $1, \dots, n$, and let $\tilde{\mu}(\cdot,\cdot)$ denote the corresponding approximated distance. Claim 3.6 implies that for every $v, v' \in V$, the probability for $|\mu(v,v') - \tilde{\mu}(v,v')| > \frac{1}{2}\delta$ is bounded by $\delta/20r'$, and so with probability at least 0.95 the situation is that for at least a $(1-\delta)$ fraction of the vectors $v \in V$, $|\mu(v,v_i) - \tilde{\mu}(v,v_i)| \leq \frac{1}{2}\delta$ for every $1 \leq i \leq r'$.

Assuming that both of the above events occurred (which is the case with probability at least 0.9), we define $V'_0, \dots, V'_{r'}$ as follows. Every vector v that belongs to V_0, or that belongs to a V_i of size $|V_i| < \delta/r$, or such that there exists some v_i for which

$|\mu(v,v_i) - \tilde{\mu}(v,v_i)| > \frac{1}{2}\delta$, is placed in V'_0. For every other vector we let i be the index for which $\tilde{\mu}(v,v_i)$ is minimal (or the smallest such index if there exist several values that minimize $\tilde{\mu}(v,v_i)$), and define v to be in V'_i.

We claim that $V'_0,\dots,V'_{r'}$ is indeed a $(4\delta,r')$-clustering. First, it is easy to see that $|V'_0| \le 3\delta|V| < 4\delta|V|$ from the assumption on the size of V_0, and the guarantee that we have on the number of vectors for which the distance was not well approximated. Now, if $u,v \in V'_i$ for some $1 \le i \le r'$, then we first note that $\mu(u,v_i) \le 2\delta$. This is because if we denote by $1 \le j \le r$ the index for which $u \in V_j$, then we have $\mu(u,v_i) \le \tilde{\mu}(u,v_i) + \frac{1}{2}\delta \le \tilde{\mu}(u,v_j) + \frac{1}{2}\delta \le \mu(u,v_j) + \delta \le 2\delta$. The same goes for proving that $\mu(v,v_i) \le 2\delta$, and so by the triangle inequality $\mu(u,v) \le 4\delta$. This concludes the claim about $V'_0,\dots,V'_{r'}$.

We now describe the remainder of the algorithm: After choosing $v_1,\dots,v_{r'}$ and the l coordinates as above, the algorithm now queries each of these coordinates from each v_i, and by this concludes the preprocessing stage. For the oracle stage, given a vector $v \in V$ the algorithm queries all the l chosen coordinates of v, and then calculates $\tilde{\mu}(v,v_i)$ for every i. The algorithm then outputs the index i that minimizes this, or the smallest such index in case there is more than one. It is clear that the algorithm gives the correct index for every vector that is not in V'_0, whose size is bounded by 4δ, concluding the proof. □

We note here that we could also use the above to find an approximate oracle for a $(4\delta,r)$-clustering (instead of a $(4\delta,10r^2/\delta)$-clustering), by trying to get from the set of queried vectors a subset V' for which all but at most a 3δ fraction of the members of V are δ-close to a member of V' (and verifying the validity of V' using a polynomial number of additional queries). This would also improve the dependencies in Lemma 2.3, but we omit it as our proofs already ensure the polynomial dependence on ϵ without this improvement.

We are now ready to describe the algorithm that proves Lemma 2.3, by finding with probability $\frac{3}{4}$ a signature of a $(16\delta^{1/6}, 10r^2/(4\delta^{1/3})+1)$-partition of M, if M has a (δ,r)-partition.

Algorithm Sig.

- By Claim 3.2, there exists a $(4\delta^{1/3},r)$-clustering of the rows. We perform the preprocessing stage of the algorithm provided by Lemma 3.7 to obtain an approximate oracle for a $(16\delta^{1/3}, 10r^2/(4\delta^{1/3}))$-clustering of the set of rows of M; we denote it by $R'_0,\dots,R'_{r'}$ for $r' = 10r^2/(4\delta^{1/3})$. Similarly, we obtain an approximate oracle for a $(16\delta^{1/3},r')$-clustering $C'_0,\dots,C'_{r'}$ of the columns.
- We now choose uniformly and independently at random (with repetitions) a (multi)set R of $l = (100r'\log r')/\delta$ rows of M, and for each of these we use the clustering oracle for $R'_0,\dots,R'_{r'}$. For $1 \le i \le r'$, we set α_i to be the number of rows from R for which the oracle answered "i," divided by l. We do the analogous operation for a set C of l columns M that were uniformly and independently chosen (this time with respect to the oracle for $C'_0,\dots,C'_{r'}$), and use it to set β_i for $1 \le i \le r'$. Both α_0 and β_0 are set to 0, as the above oracles never correctly detect that a row is in R'_0 or a column is in C'_0.
- Finally, for every $1 \le i \le r'$ and $1 \le j \le r'$ we look at the intersections of all the rows in R which the oracle located in R'_i, and all the columns in C which the oracle located in C'_j. We query the entries of M at the intersections of the set of sampled rows R and the set of sampled columns C, and we set $P_{i,j}$ to be the value (0 or 1) that has the majority of appearances in these queries.

We now claim that this algorithm satisfies the assertion of Lemma 2.3. First, we note that with probability at least 0.8, the oracles for both the clustering of the rows and the clustering of the columns are valid, as guaranteed by Lemma 3.7. In turn this guarantees that $R'_0, \dots, R'_{r'}$ and $C'_0, \dots, C'_{r'}$ form a $(16\delta^{1/6}, r'+1)$-partition of M, by Claim 3.3. Also, each of the following occurs with probability at least 0.99:

- The difference between every α_i and the total fraction of the rows of M for which the oracle would output "i" is at most δ/r'. This implies that $\sum_{i=0}^{r'} |\frac{|R'_i|}{n} - \alpha_i| \le 2 \cdot 16\delta^{1/3} + r' \cdot \delta/r' < 33\delta^{1/3}$.
- Similarly to the above, $\sum_{i=0}^{r'} |\frac{|C'_i|}{n} - \beta_i| < 33\delta^{1/3}$. With the previous item this means that for all but at most a $10\delta^{1/6}$ fraction of the pairs (i,j), both $|\frac{|R'_i|}{n} - \alpha_i| \le 7\delta^{1/6}$ and $|\frac{|C'_j|}{n} - \beta_j| \le 7\delta^{1/6}$.
- The fraction of appearances of "1" in the values taken under consideration when calculating $P_{i,j}$ differs from the fraction of appearances in the intersections of all rows assigned to "i" and all columns assigned to "j" (by the oracles) by no more than δ. In addition, by the previous item for all but at most a $10\delta^{1/6}$ fraction of the pairs (i,j), the above fraction differs by no more than $14\delta^{1/6}$ from the fraction of appearances of "1" in $R'_i \times C'_j$, and so (if δ is small enough) for the $16\delta^{1/6}$-homogeneous blocks among these, $P_{i,j}$ will get the correct value. Hence, the (weighted) fraction of wrong $P_{i,j}$ labels is no more than $16\delta^{1/6} + 10\delta^{1/6} = 26\delta^{1/6}$.

Therefore, with probability at least $\frac{3}{4}$ all the above occurs (including the two oracles being valid), and a $26\delta^{1/6}$-signature of a $16\delta^{1/6}$-partition is obtained. □

As a final remark, the proof of Lemma 1.6, given in the next section, also uses an interim lemma about clusterings, Lemma 4.1 below. One could save further on the number of queries in the main theorem if the notion of (δ, r)-clustering would be used throughout instead of the notion of (δ, r)-partitions, but it would still be polynomial (not linear) in ϵ. However, the notion of (δ, r)-partitions is more intuitive and could have applications outside the scope of this work, so we use it instead.

4. Proof of Lemma 1.6. We use the same definition of a (δ, r)-clustering (for sets of rows or columns) as we used in the previous section. Claim 3.3, which was proved above, implies that if A has a $(\delta^2/16, t)$-clustering for both its rows and its columns, then A admits a $(\delta, t+1)$-partition. Therefore, the following lemma immediately implies Lemma 1.6. Moreover, it follows that Lemma 1.6 is true even if we insist on the forbidden submatrices also obeying the order of the rows and the columns of the input matrix (which is ignored for our use of a matrix as representing a bipartite graph).

LEMMA 4.1. *Let k be a fixed integer and let $\delta > 0$ be a small real. For every $n \times n$, 0/1-matrix A, with $n > (k/\delta)^{O(k)}$, either A admits (δ, r)-clusterings for both the rows and columns with $r \le (k/\delta)^{O(k)}$, or for every $k \times k$, 0/1 matrix F, at least a $(\delta/k)^{O(k^2)}$ fraction of the $k \times k$ (ordered) submatrices of A are copies of F.*

We should also note that the above estimate is essentially tight, as shown by a random $n \times n$ matrix A, where each entry is independently chosen to be 1 with probability 2δ, and 0 with probability $1 - 2\delta$. The expected number of copies of the $k \times k$ all 1 matrix in such a matrix is only a $(2\delta)^{k^2}$ fraction of the total number of $k \times k$ submatrices, and it is not difficult to check that with high probability A does not have a $(\delta, o(n))$-clustering for either its rows or its columns.

We will prove the lemma only for the clustering of the columns, because the proof for rows is virtually identical. We make no attempt to optimize the absolute constants and omit all floor and ceiling signs to simplify the presentation. In order to prove the above lemma, we first need the following simple corollary of Sauer's lemma [18, 20].

LEMMA 4.2. *For every $t > 10k$, every $t \times t^{2k-1}$ binary matrix M with no two identical columns contains every possible $k \times k$ binary matrix as a submatrix.*

Proof. By Sauer's lemma [18, 20], every set of $s = 1 + \sum_{i=0}^{k-1} \binom{t}{i}$ consecutive columns of M contains a $k \times 2^k$ submatrix that has no two identical columns (and so contains all 2^k possible binary vectors as columns). Note that $s < t^{k-1}$ and $s(1 + (k+1)\binom{t}{k}) \leq t^{2k-1}$. Thus M can be partitioned into at least $1 + (k+1)\binom{t}{k}$ blocks of size $t \times s$, each consisting of s consecutive columns. Considering these $1 + (k+1) \cdot \binom{t}{k}$ pairwise disjoint consecutive blocks, we now find in each of them a $k \times 2^k$ submatrix with no identical columns. Considering now the set of k rows in each such submatrix, we obtain by the pigeonhole principle k such submatrices of size $k \times 2^k$, all having the same set of rows, such that their column sets are contained in disjoint intervals (according to the column order of M), one following the other. This implies the desired result, as we can choose from each of the submatrices a desired column and thus construct any given $k \times k$ matrix. □

We now turn to the proof of Lemma 4.1. Fix δ and k, and suppose that n is large enough (as a function of δ and k, to be chosen later). Let t be the smallest integer for which $(1 - \frac{1}{2}\delta)^t t^{4k-2} < 0.1$. A simple computation shows that $t = O(\frac{k}{\delta}\log(\frac{k}{\delta}))$. Define $T = t^{2k-1}$ and suppose that A is an $n \times n$ matrix with 0/1 entries which does not have a δ-clustering of the columns of size T. We have to show that in this case A must contain many copies of every $k \times k$ matrix F.

Indeed, let S be a random set of columns of A obtained by choosing, randomly, uniformly, and independently (with repetitions) $\tau = 5T/\delta$ columns of A. We assume that $n > 10(\frac{5T}{\delta})^2$. Note that, in particular, for such an n, with probability at least 9/10 no column is chosen more than once.

CLAIM 4.3. *With probability at least* 0.9*, S contains a subset S' of T columns so that the Hamming distance between any pair of them is at least $\frac{1}{2}\delta n$.*

Proof. Let us choose the members of S one by one and construct, greedily, a subset S' of S consisting of columns so that the Hamming distance between any pair of them is at least $\frac{1}{2}\delta n$ as follows. The first member of S belongs to S', and for all $i > 1$, the ith chosen column of S is added to S' if its Hamming distance from every previous member of S' is at least $\frac{1}{2}\delta n$. Since, by assumption, there is no (δ, T)-clustering of the columns of A, as long as the cardinality of S' is smaller than T, the probability that the next chosen member of S will be added to S' is at least δ (given any history of the previous choices); otherwise it would mean that the balls of radius $\frac{1}{2}\delta n$ around the members of S' form a δ-clustering. It thus follows that the probability that by the end of the procedure the cardinality of S' will still be smaller than T is at most the probability that a binomial random variable with parameters $5T/\delta$ and δ will have value at most T. Hence this probability is smaller than 0.1, which implies the assertion of the claim. □

The usefulness of S' as above is shown by the following claim.

CLAIM 4.4. *Let S' be a fixed set of T columns of A for which the pairwise Hamming distance is at least $\frac{1}{2}\delta n$. Then, if we choose a random set R of t rows of A by choosing them independently and uniformly at random, with probability at least* 0.9 *all the projections of the members of S' on the rows in R are distinct.*

Proof. Let S' be a fixed set of T columns of A so that the Hamming distance between every pair is at least $\frac{1}{2}\delta n$. For any two fixed columns $c_1, c_2 \in S'$ and a random row r we have that the probability that $c_1[r] = c_2[r]$ is at most $1 - \frac{1}{2}\delta$, where $c[j]$ denotes the jth coordinate of c. Hence, the expected number of pairs of members of S' whose projections on R are identical is at most $\binom{T}{2}(1-\frac{1}{2}\delta)^t < 0.1$, where the last inequality follows from the choice of t. The desired result follows. □

We can now conclude the proof of Lemma 4.1 as follows. Fix F to be any $k \times k$, 0/1 matrix. Choosing a random $t \times \tau$ submatrix C of A is just like choosing a set R of t random rows and a set S of τ random columns. By Claim 4.3, with probability at least 0.9, the set S of τ columns contains a subset of the columns S' of size T that has pairwise distances at least $\frac{1}{2}\delta n$. Given that this happens, by Claim 4.4 with probability 0.9 all the t projections of S' on the t rows of C are distinct. Hence with probability at least 0.8 (the probability that both events above hold) Lemma 4.2 ensures that C contains F as a submatrix.

Now choosing a random $k \times k$ submatrix of A can be viewed as first choosing a random $t \times \tau$ matrix C as above and then choosing a random subset of k columns and k rows in C. Hence the probability that such a random $k \times k$ matrix will be identical to F is at least $0.8/(\binom{t}{k}\binom{\tau}{k}) = (\frac{\delta}{k})^{O(k^2)}$.

5. Unfoldable graphs and 1-sided testing.

To construct a 1-sided test that is polynomial in ϵ, one would like to use the following scheme. First, the case where there is no (δ, r)-partition (for the appropriate parameters) is covered also for 1-sided algorithms by Lemma 1.6. Now, assuming that M is ϵ-far from $\mathcal{S}_F$ and has a (δ, r)-partition, using Lemma 2.3, we can find a submatrix Q that has a (δ', r)-partition with a signature similar to a (δ', r)-partition of M. We would like to show that in this case Q contains a member of F which will provide a witness for rejecting M.

However, having a Q with the same signature as a matrix M that is ϵ-far from $\mathcal{S}_F$ still does not imply that Q contains a member of F, because some of the partition blocks of Q may not be homogeneous and so their behavior may depend on n (this was circumvented in the 2-sided algorithm by checking all $n \times n$ matrices that are compatible with the signature). One way to solve this would be to use a Ramsey-like lemma like the one used in [11] to get rid of nonhomogeneous blocks, but this would create an exponential blow-up in the number of queries.

Here we take a different approach. First, in this section we prove the existence of the test only for the case where it is enough for Q to have only one row and one column from every cluster of the partition of M, and so the issue of homogeneity becomes moot. Later, we will use this special case as a lemma to prove the general case.

Definition 5.1. *A matrix M is called* unfoldable *if it contains no two identical rows and no two identical columns. Equivalently, an unfoldable bipartite graph is one that has no two vertices (on the same side) with exactly the same set of neighbors.*

A family F of matrices is called unfoldable *if all its members are unfoldable.*

The main lemma that we will prove in this section essentially states that properties definable by unfoldable matrices are testable.

Lemma 5.2. *For every ϵ, k, and a family F of unfoldable $k \times k$ or smaller matrices, there exists $\delta = (\epsilon/k)^{O(k^2)}$ such that if an $n \times n$ matrix M, where $n > (k/\epsilon)^{O(k)}$, is ϵ-far from the property $\mathcal{S}_F$, then M contains at least δn^{2k} distinct submatrices containing members of F (up to permutations).*

What we will need to use for the general case is the following corollary. In the next section we will use it on the *signature* of M to avoid dealing at all with blocks of M that are not homogeneous.

Corollary 5.3. *For every ϵ, k, and a family F of unfoldable $k \times k$ or smaller matrices, there exists $\delta = (\epsilon/k)^{O(k^2)}$ such that if an $n \times n$ matrix M, where $n > (k/\epsilon)^{O(k)}$, is ϵ-far from the property $\mathcal{S}_F$, then for every set X of δn^2 entries, M*

contains a member of F (up to permutations) that does not include any entry from X.

Proof. Every set X can clearly intersect at most $|X| \cdot \binom{n-1}{k-1}^2 < |X| n^{2k-2}$ submatrices of M. Hence, if $|X| < \delta n^2$, then Lemma 5.2 implies that, in particular, there exists a copy of a forbidden submatrix which does not intersect X. □

To prove Lemma 5.2, and also for the next section, it is more convenient to work with partitions into equally sized blocks.

Definition 5.4. *An r-partition of an $n \times n$ matrix M is called an* r-equipartition *if the size of all the sets R_i and C_j lie between $\lfloor n/r \rfloor$ and $\lceil n/r \rceil$. In an analogous manner we define a* (δ, r)-equipartition.

Note that for (δ, r)-equipartitions, a δ-signature essentially holds no more information than the δ-pattern it includes. The conditional existence of (δ', r')-equipartitions follows from that of (δ, r)-partitions by the following simple lemma.

Lemma 5.5. *For $\delta < \frac{1}{4}$, if a matrix M admits a (δ, r)-partition, then it admits also a $(\sqrt{\delta} + 3\delta, r/\delta)$-equipartition.*

Proof. For simplicity we assume that $l = \delta n/r$ is an integer. We repartition the original (δ, r)-partition of M in the following manner. From every R_i whose size is at least l we randomly and uniformly pick $s = \lfloor |R_i|/l \rfloor$ disjoint subsets $R_{i,1}, \ldots, R_{i,s}$ of size l. We call the matrix rows not picked for any $R_{i,x}$ by this procedure *leftover rows.* We now arbitrarily partition the set of leftover rows into disjoint sets of size l. We then perform the analogous procedure for the columns of the matrix M.

Now for every i and j such that $R_i \times C_j$ was δ-homogeneous, every block $R_{i,p} \times C_{j,t}$ will be $\sqrt{\delta}$-homogeneous with probability at least $1 - \sqrt{\delta}$. To see this assume without loss of generality that $R_i \times C_j$ has at most a δ-fraction of 1's. Then, for any fixed p, t, a random submatrix $R_{i,p} \times C_{j,t}$ of $R_i \times C_j$ has the same expected average value of its entries as the average value for $R_i \times C_j$, which is at most δ. Hence, by the Markov inequality, the probability that $R_{i,p} \times C_{j,t}$ will have more than a $\sqrt{\delta}$ fraction of 1's is at most $\sqrt{\delta}$. This probability is, however, the failure probability of $R_{i,p} \times C_{j,t}$ being $\sqrt{\delta}$-homogeneous.

Thus, there is a choice of the repartitions above for which the number of blocks $R_{i,p} \times C_{j,t}$ that come from δ-homogeneous blocks $R_i \times C_j$ but are not themselves $\sqrt{\delta}$-homogeneous is not more than $\sqrt{\delta}(n/l)^2$.

Also, since the original partition was δ-homogeneous, there are no more than $\delta(n/l)^2$ blocks $R_{i,p} \times C_{j,t}$ that come from blocks of the original partition that are not δ-homogeneous. Finally, there are the blocks that are related to leftover rows and columns. From the procedure it follows that there are no more than $lr \le \delta n$ leftover rows and no more than lr leftover columns. Thus the total number of such blocks is no more than $2\delta(n/l)^2$.

Counting all the above we obtain a total of not more than $(\sqrt{\delta} + 3\delta)(n/l)^2$ blocks that are not $\sqrt{\delta}$-homogeneous, and so the same bound holds also for non-$(\sqrt{\delta} + 3\delta)$-homogeneous blocks. □

Lemma 5.6. *Let k be fixed. For every $0 < \delta < \frac{1}{4}$ and any $n \times n$, $0/1$-matrix M, with $n > (k/\delta)^{O(k)}$, either M has a (δ, t)-equipartition for $t = t(\delta, k) \le (k/\delta)^{O(k)}$, or for every $0/1$-labeled $k \times k$ matrix B, an $h(\delta, k) \ge (\delta/k)^{O(k^2)}$ fraction of the $k \times k$ submatrices of M are B.*

Proof. We set $h(\delta, k) = g(\delta^2/16, k)$ and $t(\delta, k) = 16r(\delta, k)/\delta^2$, where g and r are the functions of Lemma 1.6. If M does not contain an h fraction of $k \times k$ submatrices that are identical to B, then it admits a $(\delta^2/16, r)$-partition as per Lemma 1.6. But then this implies that M admits a (δ, t)-equipartition by Lemma 5.5. □

The following lemma is the main technical tool, showing that the existence of a (δ, r)-partition (for the appropriate parameters) implies a dichotomy between being close to $\mathcal{S}_F$ and containing many forbidden matrices from F.

Lemma 5.7. *Let F be an unfoldable family of $k \times k$ or smaller matrices. Furthermore, let M be a matrix, and let P be an $\epsilon/8$-pattern of an $(\epsilon/8, t)$-equipartition of M for $t > 4k^2$. If P is $\epsilon/2$-close to $\mathcal{S}_F$, then M itself is ϵ-close to $\mathcal{S}_F$, while if P is $\epsilon/2$-far from $\mathcal{S}_F$, then M contains at least $\Omega(n/t)^{2k}$ distinct $k \times k$ matrices containing members of F (up to permutations).*

Proof. Let $R_1, \ldots, R_t$ and $C_1, \ldots, C_t$ be the $(\epsilon/8, t)$-equipartition of M, and let P be the corresponding $(\epsilon/8)$-pattern. If P is indeed $\epsilon/2$-close to $\mathcal{S}_F$, then let P' be the $\epsilon/2$-close matrix containing no members of F. Now modify M by setting every entry of M to be identical to the entry of P' corresponding to its block in the $(\epsilon/8, t)$-equipartition. Denote the modified matrix by M'. M' is ϵ-close to M, because the modified entries can only correspond to either entries where P and P' differed (a total of at most $\epsilon/2n^2$ entries), or entries that correspond to blocks that are not good with respect to P (at most $\epsilon/8n^2$), or entries that correspond to good blocks (at most $\epsilon/8n^2$, as in every good block the corresponding entry of P is $\epsilon/8$-dominant). Now since F is unfoldable, M' cannot contain members of F unless all their rows are in distinct R_i and all their columns are in distinct C_j. But then because P' contains no member of F, neither does M'.

We now assume that P is $\epsilon/2$-far from containing no member of F, and calculate the probability that a uniformly random $k \times k$ submatrix A of M is not a member of F. For simplicity we assume that t divides n. Recalling that $t > 4k^2$ we first note that with probability at least $\frac{1}{2}$ this matrix has no two rows in the same R_i and no two columns in the same C_j. Now, we condition the distribution of A on this event and note that it is identical to the one resulting from the following procedure: First choose uniformly, randomly, and independently a row $r_i \in R_i$ for every $1 \leq i \leq t$ and a column $c_j \in C_j$ for every $1 \leq j \leq t$. Denoting this matrix by Q, now let A be a uniformly random $k \times k$ submatrix of Q.

Because P is an $(\epsilon/8)$-pattern of the equipartition, no more than an $\epsilon/8$ fraction of the entries of M that make up Q come from blocks which are not $\epsilon/8$-good with respect to P. For an entry $Q_{i,j}$ of Q that does come from an $\epsilon/8$-good block $R_i \times C_j$, with probability at least $1 - \epsilon/8$ the value of $Q_{i,j}$ is identical to $P_{i,j}$. This implies that for the random set of entries of M that makes up Q, the expectation of the fraction of entries $Q_{i,j}$ that are consistent with the corresponding $P_{i,j}$ is at least $1 - \epsilon/4$. Hence, with probability at least $\frac{1}{2}$ the matrix Q is $\epsilon/2$-close to P, and so contains a member of F. Now conditioned on this event, the probability that A contains the forbidden submatrix is at least t^{-2k}. Putting all the above together using Bayes's law, the unconditional probability that a uniformly random A contains a forbidden submatrix is at least $t^{-2k}/4$, completing the proof. □

We can now put together the proof of Lemma 5.2 that concludes this section.

Proof of Lemma 5.2. If M is ϵ-far from $\mathcal{S}_F$ (where F is unfoldable), then there are two possible cases for M. Either it contains an $(\epsilon/8, t)$-equipartition for $t(\epsilon/8, k)$ as in Lemma 5.6, or M does not contain such an equipartition.

In the second case, Lemma 5.6 ensures that an $(\epsilon/k)^{O(k^2)}$ fraction of the $k \times k$ matrices are identical to an arbitrary member of F, so we are done.

In the first case, let P be an $\epsilon/8$-pattern of the equipartition of M. By Lemma 5.7 P itself cannot be $\epsilon/2$-close to $\mathcal{S}_F$ (as this would contradict the assumption that

M is ϵ-far from $\mathcal{S}_F$), and so P is $\epsilon/2$-far from $\mathcal{S}_F$. But then Lemma 5.7 implies that there is at least an $\Omega(t^{-2k}) = (\epsilon/k)^{O(k^2)}$ fraction of the $k \times k$ submatrices of M, such that each of these $k \times k$ submatrices contains members from F, as required. □

6. 1-sided testing for general bipartite graphs.

Given a family F of forbidden submatrices that may contain foldable ones, we will first construct a family $\tilde{F}$ that is related to F and is unfoldable.

Definition 6.1. *For a matrix A, we define the* folding *of A as the matrix $\tilde{A}$ resulting from A after removing all duplicate rows and columns, keeping only one of each.*[1]

For a family of matrices F, we define the folding *of F as the family $\tilde{F}$ consisting of all the foldings of the members of A.*

The main technical tool here is proven similarly to Lemma 5.7, but here we actually use Corollary 5.3 for the signature first, to address the possibility of having some nonhomogeneous blocks in our equipartition.

Lemma 6.2. *Let F be a family of $k \times k$ or smaller matrices, and let $\tilde{F}$ be the folding of F. Furthermore, let M be a matrix, and let P be a δ-pattern of a (δ, t)-equipartition of M, for $t \geq (k/\epsilon)^{O(k)}$ and $\delta = (\epsilon/k)^{O(k^2)}$. If P is $\epsilon/2$-close to $\mathcal{S}_{\tilde{F}}$, then M itself is ϵ-close to $\mathcal{S}_F$, while if P is $\epsilon/2$-far from $\mathcal{S}_{\tilde{F}}$, then M contains at least $\Omega(n/kt)^{2k}$ distinct $k \times k$ matrices containing members of F (up to permutations).*

Proof. Let $R_1, \ldots, R_t$ and $C_1, \ldots, C_t$ be the (δ, t)-equipartition of M. If P is indeed $\epsilon/2$-close to $\mathcal{S}_{\tilde{F}}$, then let P' be the $\epsilon/2$-close matrix containing no members of $\tilde{F}$. Now modify M by setting every entry of M to be identical to the entry of P' corresponding to its block in the (δ, t)-equipartition. Denote the modified matrix by M'. As in the proof of Lemma 5.7, it is not hard to see that M' is ϵ-close to M. Now M' cannot contain a member of F (up to permutations) unless P' contains a folding of this member, which is a contradiction as $\tilde{F}$ is the folding of F.

We now assume that P is $\epsilon/2$-far from containing no member of $\tilde{F}$ and calculate the probability that a uniformly random $k \times k$ submatrix A of M is not a member of F. For simplicity we assume that t divides n. We note that the distribution of picking a uniformly random $k \times k$ submatrix A is identical to the distribution of the following procedure: First choose uniformly, randomly, and independently k distinct rows $r_{i,1}, \ldots, r_{i,k} \in R_i$ for every $1 \leq i \leq t$, and k distinct columns $c_{j,1}, \ldots, c_{j,k} \in C_j$ for every $1 \leq j \leq t$. Denoting this matrix by Q, we now let A be a uniformly random $k \times k$ submatrix of Q.

Since P is a δ-pattern of the equipartition, the probability that a random entry x in M is equal to $P_{i,j}$ given that $x \in R_i \times C_j$ and that $R_i \times C_j$ is δ-good is at least $1 - \delta$. Thus, for a δ-good block, with probability at most δ its intersection with Q is not a $k \times k$ matrix whose entries are all identical to the corresponding label of P. Because P is a δ-pattern of the equipartition, the expectation of the number of blocks $R_i \times C_j$ for which their intersection with Q is not a $k \times k$ matrix whose entries are all identical to the corresponding label of P is no more than $2k^2\delta t^2$. We let X denote the set of entries of P corresponding to all such bad blocks. Let E be the event that $|X| \leq 8k^2\delta t^2$. Clearly E occurs with probability at least 3/4.

By Corollary 5.3, for X as above and the matrix P, there is a member of $\tilde{F}$ in P whose entries are disjoint from X (for an appropriate choice of the coefficient

[1] Note that if we remove one of two or more identical rows, the identity relations between columns remain exactly the same, and conversely the identity relations between rows remain exactly the same if we remove duplicate columns. Hence, the order in which we remove duplicates does not affect $\tilde{A}$ apart from a possible permutation in its rows and columns.

in the O notation in the expression of δ, and in the lower bound condition on t). However, if P contains a copy of a member $\tilde{B}$ of $\tilde{F}$ whose entries are disjoint from X, then Q contains the member B of F whose folding is $\tilde{B}$. Now conditioned on the event E, the probability that A contains the forbidden submatrix is at least $(kt)^{-2k}$. Putting all of the above together using Bayes's law, the unconditional probability that a uniformly random A contains a forbidden submatrix is at least $(kt)^{-2k}/4$, completing the proof. □

This allows us to conclude with the lemma yielding the 1-sided test.

Lemma 6.3. *For every ϵ and k there exists $\eta = (\epsilon/k)^{O(k^4)}$ such that if an $n \times n$ matrix M where $n > (k/\epsilon)^{O(k^3)}$ is ϵ-far from the property $\mathcal{S}_F$, where F is a family of $k \times k$ or smaller matrices, then M contains at least ηn^{2k} distinct submatrices containing members of F (up to permutations).*

Proof. We set $\delta = (\epsilon/k)^{O(k^2)}$ as required from Lemma 6.2 and set $t = t(\delta, k) = (k/\epsilon)^{O(k^3)}$ as per Lemma 5.6. Now if M is ϵ-far from $\mathcal{S}_F$, then either M contains a (δ, t)-equipartition or it does not.

In the second case, Lemma 5.6 ensures that there is a $(\delta/k)^{O(k^2)} = (\epsilon/k)^{O(k^4)}$ fraction of the $k \times k$ matrices, such that each of these matrices is identical to an arbitrary member of F, so we are done.

In the first case, let P be a δ-pattern of the equipartition of M. By Lemma 6.2 P itself cannot be $\epsilon/2$-close to $\mathcal{S}_{\tilde{F}}$ (as this would contradict the assumption that M is ϵ-far from $\mathcal{S}_F$), and so P is $\epsilon/2$-far from $\mathcal{S}_{\tilde{F}}$. But then Lemma 6.2 implies that M contains at least an $\Omega((tk)^{-2k}) \geq (\epsilon/k)^{O(k^4)}$ fraction of the $k \times k$ submatrices of M that contain members from F, as required. □

Corollary 6.4. *The property $\mathcal{S}_F$ is ϵ-testable with $(\epsilon/k)^{O(k^4)}$ many queries.*

Proof. Using the η of Lemma 6.3, select independently $3/\eta$ uniformly random $k \times k$ submatrices of M, and for each of them, check whether it contains a member of F. □

7. Open problems.

More general combinatorial structures. A long standing question in graph property testing is that of whether there exists a test for the property of a (general) graph being triangle-free, whose number of queries is less than a tower function in ϵ. Noting the "conditional regularity" nature of Lemma 1.6 here, one would hope for an analogue that will work for triangles. However, formulating such an analogue is not as simple as it seems: Gowers [12] constructed a bipartite (hence triangle-free) graph in which there is a tower lower bound on the size of the smallest regular partition. Hence, the only hope would be of finding a partition in which most of the nonregular pairs are somehow labeled as "irrelevant" for the existence of a triangle in the graph. This still remains open; we already know, however, by [1] that, unlike the case of bipartite graphs, a polynomial dependency (in $1/\epsilon$) is not possible for this case.

Another interesting open question would be to formulate a lemma in the spirit of Lemma 1.6 for higher dimensional matrices that would in turn correspond to r-partite r-uniform hypergraphs. Here too there is probably no avoiding the existence of "irrelevant" portions for which there is no regularity. Take, for example, any 3-dimensional matrix which is constant along the last dimension; it does not contain, for example, the $2 \times 2 \times 2$ matrix that is all zero apart from exactly one entry, while it may still not admit any relatively small regular partition.

Matrices with row and column order. This direction seems at the moment more accessible than those outlined above. It would be interesting to test a matrix for the property of not containing a member of a forbidden family of submatrices, with the same row and column orders (i.e., containing a nontrivial row or column permutation of a forbidden matrix is now allowed). Lemma 1.6 also holds for this framework, so the missing part would be "untangling" the sets of rows and columns in the resulting partition, in order to prove from this partition that one need only consider a set of possible input matrices that can be calculated from a small sample (as in the proof of Theorem 1.3).

Nonbinary matrices. It would also be interesting to prove the result for matrices that are not binary. It is enough to look at matrices with a fixed finite alphabet, because one does not need to distinguish between the different labels that do not appear in the finite set of forbidden matrices F.

Again "full conditional regularity" cannot be guaranteed, but this problem might be a little more accessible (though perhaps with a no longer polynomial dependence of the number of queries on ϵ). A possible course of attack could be to start by partitioning into blocks, each containing less than the full set of labels, and continue by recursively classifying each block as either "repartitionable" or "homogeneous" in a way somewhat reminiscent of what was done (more easily) in [11, 10] for poset properties.

Acknowledgment. We wish to thank Eyal Rozenberg for the discussion concerning an inaccuracy in an earlier version of the proof of Theorem 1.4. We also wish to thank two anonymous referees for their thoughtful comments.

REFERENCES

[1] N. Alon, *Testing subgraphs in large graphs*, Random Structures Algorithms, 21 (2002), pp. 359–370.

[2] N. Alon, S. Dar, M. Parnas, and D. Ron, *Testing of clustering*, SIAM J. Discrete Math., 16 (2003), pp. 393–417.

[3] N. Alon, E. Fischer, M. Krivelevich, and M. Szegedy, *Efficient testing of large graphs*, Combinatorica, 20 (2000), pp. 451–476.

[4] N. Alon and A. Shapira, *Testing subgraphs in directed graphs*, J. Comput. System Sci., 69 (2004), pp. 354–382.

[5] N. Alon and J. H. Spencer, *The Probabilistic Method*, 2nd ed., John Wiley, New York, 2000.

[6] M. Blum, M. Luby, and R. Rubinfeld, *Self-testing/correcting with applications to numerical problems*, J. Comput. System Sci., 47 (1993), pp. 549–595.

[7] R. Diestel, *Graph Theory*, 2nd ed., Springer-Verlag, New York, 2000.

[8] E. Fischer, *Testing graphs for colorability properties*, Random Structures Algorithms, 26 (2005), pp. 289–309.

[9] E. Fischer, *The art of uninformed decisions: A primer to property testing*, Bull. Eur. Assoc. Theor. Comput. Sci. EATCS, 75 (2001), pp. 97–126.

[10] E. Fischer and I. Newman, *Testing of matrix properties*, in Proceedings of the 33rd ACM Annual Symposium on Theory of Computing, ACM, New York, 2001, pp. 286–295.

[11] E. Fischer and I. Newman, *Testing of matrix-poset properties*, Combinatorica, to appear.

[12] W. T. Gowers, *Lower bounds of tower type for Szemerédi's Uniformity Lemma*, Geom. Funct. Anal., 7 (1997), pp. 322–337.

[13] T. Kövary, V.T. Sós, and P. Turán, *On a problem of K. Zarankiewicz*, Colloq. Math., 3 (1954), pp. 50–57.

[14] O. Goldreich, S. Goldwasser, and D. Ron, *Property testing and its connection to learning and approximation*, J. ACM, 45 (1998), pp. 653–750.

[15] O. Goldreich and L. Trevisan, *Three theorems regarding testing graph properties*, Random Structures Algorithms, 23 (2003), pp. 23–57.

[16] D. Ron, *Property testing (a tutorial)*, in Handbook of Randomized Computing, Vol. II, S. Rajasekaran, P. M. Pardalos, J. H. Reif, and J. D. P. Rolim, eds., Kluwer Academic Publishers, Boston, 2001, pp. 597–649.

[17] R. Rubinfeld and M. Sudan, *Robust characterizations of polynomials with applications to program testing*, SIAM J. Comput., 25 (1996), pp. 252–271.

[18] N. Sauer, *On the density of families of sets*, J. Combin. Theory Ser. A, 13 (1972), pp. 145–147.

[19] E. Szemerédi, *Regular partitions of graphs*, in Problems combinatoires et théorie des graphes, Colloq. Internat. CNRS 260, J. C. Bermond, J. C. Fournier, M. Las Vergnas, and D. Sotteau, eds., CNRS, Paris, 1978, pp. 399–401.

[20] S. Shelah, *A combinatorial problem: Stability and order for models and theories in infinitary languages*, Pacific J. Math., 41 (1972), pp. 247–261.

[21] K. Zarankiewicz, *Problem P* 101, Colloq. Math., 2 (1951), pp. 116–131.

SIAM J. COMPUT.
Vol. 37, No. 4, pp. 977–1013

OBSERVING BRANCHING STRUCTURE THROUGH PROBABILISTIC CONTEXTS*

NANCY LYNCH†, ROBERTO SEGALA‡, AND FRITS VAANDRAGER§

Abstract. Probabilistic automata (PAs) constitute a general framework for modeling and analyzing discrete event systems that exhibit both nondeterministic and probabilistic behavior, such as distributed algorithms and network protocols. The behavior of PAs is commonly defined using schedulers (also called adversaries or strategies), which resolve all nondeterministic choices based on past history. From the resulting purely probabilistic structures, trace distributions can be extracted, whose intent is to capture the observable behavior of a PA. However, when PAs are composed via an (asynchronous) parallel composition operator, a global scheduler may establish strong correlations between the behavior of system components and, for example, resolve nondeterministic choices in one PA based on the outcome of probabilistic choices in the other. It is well known that, as a result of this, the (linear-time) trace distribution precongruence is not compositional for PAs. In his 1995 Ph.D. thesis, Segala has shown that the (branching-time) probabilistic simulation preorder is compositional for PAs. In this paper, we establish that the simulation preorder is, in fact, the coarsest refinement of the trace distribution preorder that is compositional. We prove our characterization result by providing (1) a context of a given PA $\mathcal{A}$, called the *tester*, which may announce the state of $\mathcal{A}$ to the outside world, and (2) a specific global scheduler, called the *observer*, which ensures that the state information that is announced is actually correct. Now when another PA $\mathcal{B}$ is composed with the tester, it may generate the same external behavior as the observer only when it is able to simulate $\mathcal{A}$ in the sense that whenever $\mathcal{A}$ goes to some state s, $\mathcal{B}$ can go to a corresponding state u, from which it may generate the same external behavior. Our result shows that probabilistic contexts together with global schedulers are able to exhibit the branching structure of PAs.

Key words. probabilistic automata, trace distributions, compositionality, simulation preorder, concurrency theory

AMS subject classifications. 68Q05, 68Q10, 68Q85

DOI. 10.1137/S0097539704446487

1. Introduction.

Labeled transition systems (automata) are studied extensively within concurrency theory as underlying operational models of concurrent systems [27]: a system is described as a state machine whose transitions are labeled by *actions*, where each action describes potential communication with the external environment. An important aspect of concurrency theory is the study of relationships between systems, namely equivalence and preorder relations, with the objective of understanding whether a system can be used in place of another one or as an implementation of some

*Received by the editors November 8, 2004; accepted for publication (in revised form) December 2, 2006, published electronically September 14, 2007. A preliminary version of this paper appeared as [24].

http://www.siam.org/journals/sicomp/37-4/44648.html

†Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA 02139 (lynch@theory.csail.mit.edu). The work of this author was supported by AFOSR contract F49620-00-1-0097, AFRL award FA9550-04-1-0121, NSF grants CCR-0121277 and CCR-0326277, NSF award CNS-0614414, NSF/ITR grants CCR-0121277 and CCR-0326277, USAF/AFRL award FA9550-04-1-0121, and DARPA/AFOSR MURI F49620-02-1-0325.

‡Dipartimento di Informatica, Università di Verona, Italy (roberto.segala@univr.it). This author was supported by MURST projects MEFISTO and CoVer, MIUR project AIDA, and by Inria project ProNoBis.

§Institute for Computing and Information Sciences, Radboud University Nijmegen, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands (F.Vaandrager@cs.ru.nl). This author was supported by PROGRESS project TES4999: Verification of Hard and Softly Timed Systems (HaaST) and by DFG/NWO bilateral cooperation project Validation of Stochastic Systems (VOSS and VOSS2).

more abstract description. Several relations are studied in the literature, but the most important classes of relations are represented by simulations and bisimulations [27] and by language (trace) inclusion and equivalence [17]. An extensive classification of existing relations appears in [12], where, in particular, relations are classified as either *branching*, which observe the places where nondeterminism is resolved, or *linear*, which are insensitive to the actual places where nondeterminism is resolved. For instance, language inclusion and language equivalence are linear relations, while simulations and bisimulations are branching relations.

During the last fifteen years there has been a growing interest in the extension of concurrent models with probabilities, mainly motivated by the fact that several applications included randomized behaviors. Some of the most relevant proposals of operational models with probability and nondeterminism are reactive, generative, and stratified systems [13], concurrent labeled Markov chains [15], alternating automata [40, 29], probabilistic automata (PAs) [32], and probabilistic reactive modules [10]. Extensive comparative studies that include these models appear in [36, 4, 35].

Simulation, bisimulation, and language inclusion relations have been extended to the probabilistic case as well. In particular, [22] defines strong bisimulation on reactive systems, [34] defines strong and weak simulation and bisimulation relations on PAs, including a notion of branching bisimulation, [15] defines strong bisimulation on labeled concurrent Markov chains, [29] defines strong and weak bisimulation on alternating automata, and [2] defines branching bisimulation on alternating automata. Although the above definitions are quite different, it turns out that they can all be seen in a uniform way by viewing reactive systems, labeled concurrent Markov chains, and alternating automata as special cases of PAs [35]. For extensive comparative studies we refer the reader again to [36, 4, 35].

In this paper we are interested in extensions of language inclusion to the probabilistic case. On ordinary nondeterministic automata the resolution of nondeterminism produces sequences of alternating states and actions called *executions*; then, by restricting those sequences to visible actions, we obtain the so-called *traces*. Implementation and equivalence of nondeterministic automata can be defined in terms of inclusion and equality of sets of traces. This approach was first proposed in the context of process algebras [17] and is used extensively in the area of I/O automata [25].

An attempt to extend language inclusion to PAs appears in [31], where it is proposed that the probabilistic extension of a trace should be a probability measure over traces. Indeed, the resolution of nondeterminism on PAs produces a stochastic process that induces a probability measure over executions (a *probabilistic execution*), and the restriction of a probabilistic execution to the externally visible actions leads to a probability measure over traces (a *trace distribution*). Then, the proposal of [31] is to compare PAs based on inclusion and equality of sets of trace distributions. This is consistent with ordinary nondeterministic automata since an execution can be seen as a probabilistic execution that assigns probability 1 to a single element, and similarly a trace can be seen as a trace distribution that assigns probability 1 to a single element.

There are several arguments in favor of the point of view that probabilistic executions in PAs should play the role that executions play in ordinary nondeterministic automata, and thus in favor of the notion of trace distribution as well. One element is that this point of view leads to the definition of *weak transitions*, used to extend weak simulations and bisimulation to the probabilistic case. Another element of evidence comes from the area of distributed algorithms, where the probability of termination of an algorithm is studied under any scheduling policy: in this context a scheduler is the entity that resolves nondeterminism, for example, by choosing the order in which processes take steps, and a probabilistic execution is the natural object where the probability of termination can be computed, as demonstrated by several case studies [23, 30, 1, 39, 20, 28, 6] and by the ongoing research on automatic verification tools for probabilistic systems [16]. Finally, again in the area of distributed algorithms, the approach of [31] to language inclusion turns out to be useful for the modular analysis of complex algorithms [30].

An important requirement for an implementation relation on systems is *compositionality*, that is, the relation is preserved by parallel composition. For labeled transition systems, the trace, simulation, and bisimulation preorders are all compositional [17, 27]. For PAs, various simulation and bisimulation preorders are known to be compositional [34]. A problem with the trace-based relations proposed in [31] is that they are not compositional; that is, they are not preserved by parallel composition. A typical solution to the problem, followed by [31], is to define a notion of *trace distribution precongruence* as the coarsest precongruence included in the trace distribution inclusion. Unfortunately, such implicit definition does not provide much insight about the structure of the relation. For this reason, there have been several attempts to characterize it in more concrete terms. In [32] trace distribution precongruence is characterized in terms of the set of trace distributions observable in a certain *principal context*—a rudimentary PA that makes very limited nondeterministic and probabilistic choices; in [33] a testing scenario is proposed. However, these indirect characterizations still do not provide much insight into the structure of trace distribution precongruence; for example, they do not explain its branching structure. Indeed, trace distribution precongruence is not a linear relation since it distinguishes ordinary nondeterministic automata that are trace equivalent.

In this paper, we provide an explicit characterization of the trace distribution precongruence, $\leq_{DC}$, for PAs, which completely explains its branching structure. Namely, we show that $\mathcal{P}_1 \leq_{DC} \mathcal{P}_2$ iff there exists a *weak probabilistic (forward) simulation relation* from $\mathcal{P}_1$ to $\mathcal{P}_2$. Moreover, we provide a similar characterization of $\leq_{DC}$ for nondeterministic automata in terms of the existence of a weak (nonprobabilistic) simulation relation. It was previously known that simulation relations are sound for $\leq_{DC}$ [32], for both nondeterministic and probabilistic automata; we show the surprising fact that they are also *complete*. That is, we show that, for both nondeterministic and probabilistic automata, probabilistic contexts can observe all the distinctions that can be expressed using simulation relations.

Our proofs of completeness rely on special contexts for PAs, called *testers*. The tester of a PA $\mathcal{P}$, under the action of an appropriate scheduler, can reveal the branching structure of $\mathcal{P}$ via a trace distribution. Such a scheduler is called an *observer scheduler*. Informally, the tester $\mathcal{C}$ of a PA $\mathcal{P}$ announces the outcome of each probabilistic choice of $\mathcal{P}$ by performing an action with the name of the state reached, and flips coins to propose and announce how $\mathcal{P}$ should resolve its nondeterministic choices. The ability of another PA $\mathcal{P}'$ to comply with the requirements of $\mathcal{C}$, that is, that the trace distribution induced by the observer scheduler is also a trace distribution of $\mathcal{P}'\|\mathcal{C}$ (the parallel composition of $\mathcal{P}'$ and $\mathcal{C}$), reveals whether $\mathcal{P}'$ has at least the same possibilities for solving nondeterministic choices as $\mathcal{P}$. If $\mathcal{P} \leq_{DC} \mathcal{P}'$, then we extract a probabilistic forward simulation from $\mathcal{P}$ to $\mathcal{P}'$ by observing how $\mathcal{P}'\|\mathcal{C}$ produces the trace distribution induced by the observer scheduler.

An interesting observation about tester automata is that probabilistic choices of a PA are observed via nondeterministic choices of the tester automaton, while nondeterministic choices of a PA are observed via probabilistic choices of the tester automaton. Thus, the branching structure of a PA is observed via a probabilistic context.

The rest of the paper is structured as follows. Sections 2 and 3 contain basic definitions and results for nondeterministic and probabilistic automata, respectively, and for the preorders we consider. These sections contain no new material, but recall definitions and theorems from the literature. For a more leisurely introductions see [25, 26, 38, 36]. Section 4 introduces the concept of tester automaton and the scheduler for a PA $\mathcal{P}$ and its tester that reveals the structure of $\mathcal{P}$. Sections 5 and 6 contain, respectively, our characterization results for nondeterministic and probabilistic automata. Since the proof of the characterization result for the general case of PAs with internal actions is highly complex, we first present a proof for the special case of nondeterministic automata without internal actions (section 5.1). Then we successively show how we can also handle internal actions (section 5.2) and probabilistic choice (section 6.1) before dealing with the general case of PAs with internal actions (section 6.2). Section 7 contains our conclusions.

2. Definitions and basic results for nondeterministic automata.

In this section we recall definitions and basic results for nondeterministic automata. We impose a few restrictions to avoid confusion and unnecessary complications in the rest of the paper. For more information the reader is referred to [25, 26].

2.1. Nondeterministic automata, executions, and traces.

A *nondeterministic automaton* is a tuple $\mathcal{A} = (Q, \bar{q}, E, H, D)$, where

- Q is a countable set of *states*,
- $\bar{q} \in Q$ is a *start state*,
- E is a countable set of *external actions*,
- H is a countable set of *internal (hidden) actions* with $E \cap H = \emptyset$, and
- $D \subseteq Q \times (E \cup H) \times Q$ is a *transition relation*.

We denote $E \cup H$ by A, and we refer to it as the set of *actions*. We denote a transition (q, a, q') of D by $q \xrightarrow{a} q'$. We write $q \rightarrow q'$ if $q \xrightarrow{a} q'$ for some a, and we write $q \rightarrow$ if $q \rightarrow q'$ for some q'.

We assume finite branching: for each state q the number of pairs (a, q') such that $q \xrightarrow{a} q'$ is finite. We denote the elements of a nondeterministic automaton $\mathcal{A}$ by $Q_{\mathcal{A}}, \bar{q}_{\mathcal{A}}, E_{\mathcal{A}}, H_{\mathcal{A}}, D_{\mathcal{A}}, A_{\mathcal{A}}, \xrightarrow{a}_{\mathcal{A}}$. Often we use the name $\mathcal{A}$ for a generic nondeterministic automaton; in this case, we usually omit the subscripts, writing simply Q, $\bar{q}$, E, H, D, A, and $\xrightarrow{a}$. We extend this convention to allow indices and primes as well; thus, the set of states of a nondeterministic automaton $\mathcal{A}'_i$ is denoted by Q'_i.

Remark 2.1. In the definition of nondeterministic automaton above, we have imposed some restrictions that are not strictly necessary for this paper but rather avoid unnecessary complications. The restriction on the cardinality of the sets of states and actions is imposed to ensure that a nondeterministic automaton has at most countably many finite execution fragments (see definition later), which simplifies the use of measure theory later. The finite branching restriction is imposed to simplify the construction of the tester automaton in section 4; however, the results of this paper generalize to countable branching at the cost of adding complexity to the proofs (cf. Remark 4.5). We have also chosen to define nondeterministic automata with a single initial state rather than a set of initial states. Sets of initial states do not add any technical insight, but they complicate notation slightly.

An *execution fragment* of a nondeterministic automaton $\mathcal{A}$ is a finite or infinite sequence $\alpha = q_0a_1q_1a_2q_2\cdots$ of alternating states and actions, starting with a state and, if the sequence is finite, ending in a state, where each $(q_i, a_{i+1}, q_{i+1}) \in D$. State q_0, the first state of α, is denoted by *fstate*(α). If α is a finite sequence, then the last state of α is denoted by *lstate*(α). An *execution* of $\mathcal{A}$ is an execution fragment whose first state is the start state $\bar{q}$. We let *frags*$(\mathcal{A})$ denote the set of execution fragments of $\mathcal{A}$ and *frags*$^*(\mathcal{A})$ the set of finite execution fragments. Similarly, we let *execs*$(\mathcal{A})$ denote the set of executions of $\mathcal{A}$ and *execs*$^*(\mathcal{A})$ the set of finite executions.

Execution fragment α is a *prefix* of execution fragment α', denoted by $\alpha \leq \alpha'$, if sequence α is a prefix of sequence α'. Finite execution fragment $\alpha_1 = q_0a_1q_1\cdots a_kq_k$ and execution fragment α_2 can be concatenated if *fstate*$(\alpha_2) = q_k$. In this case the *concatenation* of α_1 and α_2, $\alpha_1 \frown \alpha_2$, is the execution fragment $q_0a_1q_1\cdots a_k\alpha_2$. Given an execution fragment α and a finite prefix α', $\alpha \rhd \alpha'$ (read as "α after α'") is defined to be the unique execution fragment α'' such that $\alpha = \alpha' \frown \alpha''$.

The *trace* of an execution fragment α of a nondeterministic automaton $\mathcal{A}$, written *trace*$_{\mathcal{A}}(\alpha)$, or just *trace*(α) when $\mathcal{A}$ is clear from context, is the sequence obtained by restricting α to the set of external actions of $\mathcal{A}$. For a set S of executions of a nondeterministic automaton $\mathcal{A}$, *traces*$_{\mathcal{A}}(S)$, or just *traces*(S) when $\mathcal{A}$ is clear from context, is the set of traces of the executions in S. We say that β is a trace of a nondeterministic automaton $\mathcal{A}$ if there is an execution α of $\mathcal{A}$ with *trace*$(\alpha) = \beta$. Let *traces*$(\mathcal{A})$ denote the set of traces of $\mathcal{A}$. We define the *trace preorder* relation on nondeterministic automata as follows: $\mathcal{A}_1 \leq_T \mathcal{A}_2$ iff $E_1 = E_2$ and *traces*$(\mathcal{A}_1) \subseteq$ *traces*$(\mathcal{A}_2)$. We use $\equiv_T$ to denote the kernel of $\leq_T$. That is, $\mathcal{A}_1 \equiv_T \mathcal{A}_2$ iff $\mathcal{A}_1 \leq_T \mathcal{A}_2$ and $\mathcal{A}_2 \leq_T \mathcal{A}_1$. A similar convention will be adopted to denote the kernels of other preorder relations used in the paper.

If $\beta \in A^*$, then $q \overset{\beta}{\Longrightarrow} q'$ iff there exists an execution fragment α such that *fstate*$(\alpha) = q$, *lstate*$(\alpha) = q'$, and *trace*$(\alpha) =$ *trace*(β). (Here and elsewhere, we abuse notation slightly by extending the *trace* function to arbitrary sequences.) We call $q \overset{\beta}{\Longrightarrow} q'$ a *weak transition*. If β is the empty sequence, then we write alternatively $q \Longrightarrow q'$. Observe that the definition of $q \overset{\beta}{\Longrightarrow} q'$ depends only on the external actions that occur in β. We have chosen to define weak transitions for any sequence β, including internal actions as well, for notational convenience in later definitions.

We let *tr* range over either transitions or weak transitions. For a transition $tr = (q, a, q')$, we denote q by *source*(tr) and q' by *target*(tr).

2.2. Composition. We define composition of nondeterministic automata by synchronizing them on common external actions. There are several ways to do this, but the simplest approach that is followed in several papers is to synchronize nondeterministic automata on common actions and impose the restriction that no internal action of a component is an action of the other component as well. This restriction can easily be eliminated, for example, by renaming internal actions if necessary.

Nondeterministic automata $\mathcal{A}_1$ and $\mathcal{A}_2$ are *compatible* if $H_1 \cap A_2 = A_1 \cap H_2 = \emptyset$. The *(parallel) composition* of compatible nondeterministic automata $\mathcal{A}_1$ and $\mathcal{A}_2$, denoted by $\mathcal{A}_1 \| \mathcal{A}_2$, is the nondeterministic automaton $\mathcal{A} \triangleq (Q_1 \times Q_2, (\bar{q}_1, \bar{q}_2), E_1 \cup E_2, H_1 \cup H_2, D)$, where D is the set of triples (q, a, q') such that, for $i \in \{1, 2\}$,

$$a \in A_i \Rightarrow (\pi_i(q), a, \pi_i(q')) \in D_i \quad \text{and} \quad a \notin A_i \Rightarrow \pi_i(q) = \pi_i(q'),$$

where π_i is the projection function on states of $\mathcal{A}$ defined by $\pi_i(q_1, q_2) = q_i$.

Let α be an execution fragment of $\mathcal{A}_1 \| \mathcal{A}_2$, $i \in \{1, 2\}$. Then $\pi_i(\alpha)$, the ith projection of α, is the sequence obtained from α by projecting each state onto its

ith component, and removing each action not in A_i together with its following state. Sometimes we denote this projection by $\alpha\lceil\mathcal{A}_i$.

PROPOSITION 2.2. *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be nondeterministic automata, with $\mathcal{A}_1 \leq_T \mathcal{A}_2$. Then, for each nondeterministic automaton $\mathcal{C}$ compatible with both $\mathcal{A}_1$ and $\mathcal{A}_2$, $\mathcal{A}_1\|\mathcal{C} \leq_T \mathcal{A}_2\|\mathcal{C}$.*

2.3. Simulation relations. We define two kinds of simulation relations: forward simulations, which provide a step-by-step correspondence, and weak forward simulations, which are insensitive to the occurrence of internal steps. Namely, relation $R \subseteq Q_1 \times Q_2$ is a *forward simulation* (resp., *weak forward simulation*) from $\mathcal{A}_1$ to $\mathcal{A}_2$ iff $E_1 = E_2$ and both of the following hold:

1. $\bar{q}_1 \; R \; \bar{q}_2$.
2. If $q_1 \; R \; q_2$ and $q_1 \xrightarrow{a} q_1'$, then there exists q_2' such that $q_2 \xrightarrow{a} q_2'$ (resp., $q_2 \overset{a}{\Longrightarrow} q_2'$) and $q_1' \; R \; q_2'$.

We write $\mathcal{A}_1 \leq_F \mathcal{A}_2$ (resp., $\mathcal{A}_1 \leq_{wF} \mathcal{A}_2$) when there is a forward simulation (resp., a weak forward simulation) from $\mathcal{A}_1$ to $\mathcal{A}_2$. It is easy to prove that both $\leq_F$ and $\leq_{wF}$ are preorders, that is, reflexive and transitive. Since all simulation relations in this paper are forward simulations, we often omit the word "forward."

PROPOSITION 2.3. *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be nondeterministic automata. Then*

1. *if $\mathcal{A}_1 \leq_F \mathcal{A}_2$, then $\mathcal{A}_1 \leq_{wF} \mathcal{A}_2$;*
2. *if $H_1 = H_2 = \emptyset$, then $\mathcal{A}_1 \leq_F \mathcal{A}_2$ iff $\mathcal{A}_1 \leq_{wF} \mathcal{A}_2$;*
3. *if $\mathcal{A}_1 \leq_{wF} \mathcal{A}_2$, then $\mathcal{A}_1 \leq_T \mathcal{A}_2$.*

Proof. The proof is standard; for instance, see [26]. ☐

2.4. Tree-structured automata. We say that a nondeterministic automaton is *tree-structured* if each state is reached via (i.e., occurs as a final state of) a unique execution.

The *unfolding* of nondeterministic automaton $\mathcal{A}$, denoted by *Unfold*$(\mathcal{A})$, is the tree-structured nondeterministic automaton $\mathcal{B}$ obtained from $\mathcal{A}$ by unfolding its transition graph into a tree. Formally,

- $Q_{\mathcal{B}} = \mathit{execs}^*(\mathcal{A})$,
- $\bar{q}_{\mathcal{B}} = \bar{q}_{\mathcal{A}}$,
- $E_{\mathcal{B}} = E_{\mathcal{A}}$,
- $H_{\mathcal{B}} = H_{\mathcal{A}}$, and
- $D_{\mathcal{B}} = \{(\alpha, a, \alpha a q) \mid (\mathit{lstate}(\alpha), a, q) \in D_{\mathcal{A}}\}$.

PROPOSITION 2.4. $\mathcal{A} \equiv_F \mathit{Unfold}(\mathcal{A})$.

Proof. See [26]. It is easy to check that the relation R, where $\alpha \; R \; q$ iff $\mathit{lstate}(\alpha) = q$, is a forward simulation from *Unfold*$(\mathcal{A})$ to $\mathcal{A}$ and that the inverse relation of R is a forward simulation from $\mathcal{A}$ to *Unfold*$(\mathcal{A})$. ☐

PROPOSITION 2.5. $\mathcal{A} \equiv_T \mathit{Unfold}(\mathcal{A})$.

Proof. The proof follows by Proposition 2.4 and Proposition 2.3, parts 1 and 3. ☐

3. Definitions and basic results for probabilistic automata.

3.1. Preliminaries and notation on measure theory. We recall a few basic definitions and notation for measure theory that can be retrieved from any standard book on the subject (e.g., [11]).

A *σ-field* over a set X is a set $\mathcal{F} \subseteq 2^X$ that contains the empty set and is closed under complement and countable union. A pair $(X, \mathcal{F})$, where $\mathcal{F}$ is a σ-field over X, is called a *measurable space.* A *measure* on a measurable space $(X, \mathcal{F})$ is a function $\mu : \mathcal{F} \to [0, \infty]$ such that $\mu(\emptyset) = 0$ and μ is countably additive: for each countable family $\{C_i\}_i$ of pairwise disjoint elements of $\mathcal{F}$, $\mu(\cup_i C_i) = \sum_i \mu(C_i)$. A *probability measure* on $(X, \mathcal{F})$ is a measure μ on $(X, \mathcal{F})$ such that $\mu(X) = 1$. A *subprobability measure* on $(X, \mathcal{F})$ is a measure μ on $(X, \mathcal{F})$ such that $\mu(X) \leq 1$. A *discrete probability measure* on a set X is a probability measure μ on $(X, 2^X)$. A *discrete subprobability measure* on X is a subprobability measure μ on $(X, 2^X)$. We denote the set of discrete probability measures and discrete subprobability measures on X by $\mathit{Disc}(X)$ and $\mathit{SubDisc}(X)$, respectively. We denote the *support* of a discrete measure μ, that is, the set of elements of X that have nonzero measure, by $\mathit{supp}(\mu)$. We let $\delta(q)$ denote the *Dirac measure* for q, the discrete probability measure that assigns probability 1 to $\{q\}$. Finally, if X is nonempty and finite, then $\mathcal{U}(X)$ denotes the *uniform distribution* over X, the discrete measure that assigns probability $1/|X|$ to each element of X. Given two discrete probability measures μ_1, μ_2 on $(X, 2^X)$ and $(Y, 2^Y)$, respectively, we denote by $\mu_1 \times \mu_2$ the *product measure*, that is, the measure on $(X \times Y, 2^{(X \times Y)})$ such that $\mu_1 \times \mu_2((x, y)) = \mu_1(x)\mu_2(y)$ for each $x \in X, y \in Y$.

Sometimes it is useful to know the probability μ of some event C, knowing that some other event C' takes place. We call this the measure of C *conditional* on C' and denote it by $\mu(C \mid C')$. Such probability is defined to be 0 if $\mu(C') = 0$, and $\mu(C \cap C')/\mu(C')$ otherwise.

A function $f : X \to Y$ is said to be *measurable* from $(X, \mathcal{F}_X)$ to $(Y, \mathcal{F}_Y)$ if the inverse image of each element of $\mathcal{F}_Y$ is an element of $\mathcal{F}_X$, that is, for each $C \in F_Y$, $f^{-1}(C) \in \mathcal{F}_X$. In such a case, given a measure μ on $(X, \mathcal{F}_X)$, the function $f(\mu)$ defined on $\mathcal{F}_Y$ by $f(\mu)(C) = \mu(f^{-1}(C))$ for each $C \in \mathcal{F}_Y$ is a measure on $(Y, \mathcal{F}_Y)$ and is called the *image measure* of μ under f.

Given a countable collection of measures $\{\mu_i\}_i$ on $(X, \mathcal{F}_X)$ and a countable collection $\{p_i\}_i$ of real numbers in $[0, \infty)$, denote by $\sum_i p_i \mu_i$ a new function μ such that, for each element $C \in \mathcal{F}_X$, $\mu(C) = \sum_i p_i \mu_i(C)$. We state a few elementary properties.

PROPOSITION 3.1. *The following hold:*

1. $\sum_i p_i \mu_i$ *is a measure on* $(X, \mathcal{F}_X)$.
2. *If each* μ_i *is a (sub)probability measure and* $\sum_i p_i = 1$*, then* $\sum_i p_i \mu_i$ *is a (sub)probability measure.*
3. *If* f *is a measurable function from* $(X, \mathcal{F}_X)$ *to* $(Y, \mathcal{F}_Y)$*, then* $f(\sum_i p_i \mu_i) = \sum_i p_i f(\mu_i)$.

3.2. PAs, executions, and traces. A *probabilistic automaton (PA)* is a tuple $\mathcal{P} = (Q, \bar{q}, E, H, D)$, where all components are exactly as for nondeterministic automata, except that the following holds:

- D, the *transition relation*, is a subset of $Q \times (E \cup H) \times \mathit{Disc}(Q)$.

We define A as before. Also, we use the name $\mathcal{P}$ for a generic PA, and we refer to its components by writing simply Q, $\bar{q}$, E, H, D, A, and $\xrightarrow{a}$. We extend this convention to allow indices and primes as well; thus, the set of states of a PA $\mathcal{P}'_i$ is denoted by Q'_i. We denote a transition (q, a, μ) by $q \xrightarrow{a} \mu$. We assume finite branching: for each state q the number of pairs (a, μ) such that $q \xrightarrow{a} \mu$ is finite. Given a transition $tr = (q, a, \mu)$, we denote q by $\mathit{source}(tr)$ and μ either by $\mathit{target}(tr)$ or by μ_{tr}.

Thus, a PA differs from a nondeterministic automaton in that a transition leads to a probability measure over states rather than to a single state. A nondeterministic automaton can be viewed as a special case of a PA, where the last component of each transition is a Dirac measure. Conversely, we can associate a nondeterministic automaton with each PA by replacing transition relation D by the relation D' given by

$$(q, a, q') \in D' \Leftrightarrow (\exists \mu)[(q, a, \mu) \in D \wedge \mu(q') > 0].$$

Using this correspondence, notions such as execution fragments and traces carry over from nondeterministic automata to PAs.[1] For instance, an execution fragment of a PA is simply an execution fragment of its associated nondeterministic automaton. Along the same lines we write $q \overset{a}{\to} q'$ whenever there exists a measure μ such that $q \overset{a}{\to} \mu$ and $q' \in \mathit{supp}(\mu)$.

An execution fragment of a PA is the result of resolving nondeterministic as well as probabilistic choices; however we are interested also in the outcome of the resolution of nondeterministic choices only. We can think of resolving nondeterminism by unfolding the transition relation of a PA and then choosing only one transition at each point. From the formal point of view it is more convenient to define a function, called a *scheduler*, that chooses transitions based on the past history (i.e., the current position in the unfolding of the transition relation).

A *scheduler* for a PA $\mathcal{P}$ is a function $\sigma : \mathit{frags}^*(\mathcal{P}) \to \mathit{SubDisc}(D)$ such that $\mathit{tr} \in \mathit{supp}(\sigma(\alpha))$ implies $\mathit{source}(\mathit{tr}) = \mathit{lstate}(\alpha)$. A scheduler σ is said to be *deterministic* if for each finite execution fragment α either $\sigma(\alpha)(D) = 0$ or else $\sigma(\alpha) = \delta(\mathit{tr})$ (the Dirac measure for tr) for some $\mathit{tr} \in D$. A scheduler σ is *memoryless* if it depends only on the last state of its argument, that is, for each pair α_1, α_2 of finite execution fragments, if $\mathit{lstate}(\alpha_1) = \mathit{lstate}(\alpha_2)$, then $\sigma(\alpha_1) = \sigma(\alpha_2)$.

Informally, $\sigma(\alpha)$ describes the rule for choosing a transition after α has occurred. The rule itself may be randomized. Since $\sigma(\alpha)$ is a subprobability measure, it is possible that with some nonzero probability no transition is chosen, which corresponds to terminating the computation (with what in nondeterministic automata is called a finite execution fragment). Deterministic schedulers are not allowed to use randomization in their choices, while memoryless schedulers are not allowed to look at the past history in their choices. Deterministic and memoryless schedulers are easier to analyze compared to general schedulers, and several properties (e.g., reachability) can be studied by referring to deterministic memoryless schedulers only. Note that a deterministic memoryless scheduler can be represented alternatively as a partial function from Q to D.

A scheduler σ and a discrete probability measure over states μ induce a measure ϵ on the σ-field generated by cones of execution fragments as follows. If α is a finite execution fragment, then the *cone* of α is defined by $C_\alpha = \{\alpha' \in \mathit{frags}(\mathcal{P}) \mid \alpha \leq \alpha'\}$. The measure ϵ of a cone C_α is defined to be $\mu(q)$ if $\alpha = q$ for some state $q \in Q$, and if α is of the form $\alpha' a' q'$, it is defined by the recursive equation

$$\epsilon(C_\alpha) = \epsilon(C_{\alpha'}) \sum_{\mathit{tr} \in D(a')} \sigma(\alpha')(\mathit{tr}) \mu_{\mathit{tr}}(q'), \tag{1}$$

where $D(a')$ denotes the set of transitions of D that are labeled by a'. Roughly speaking, the measure of a cone C_α equals the probability of doing α when using σ to resolve nondeterminism. Standard measure theoretical arguments ensure that ϵ is well defined. We call the measure ϵ a *probabilistic execution fragment* of $\mathcal{P}$, and we say that ϵ is *generated* by σ and μ. We also denote by $\epsilon_{\sigma,\mu}$ the probabilistic execution fragment generated by σ and μ.

[1] The correspondence between nondeterministic automata and PAs is worked out in great detail in [4].

Proposition 3.2. *Let σ be a scheduler and μ be a discrete probability measure over states. Then fstate$(\epsilon_{\sigma,\mu}) = \mu$.*

Proof. The proof follows immediately by definition of $\epsilon_{\sigma,\mu}$ after observing that the inverse image under *fstate* of a state q is the set C_q. □

We call the measure *fstate*(ϵ) the *first state* of ϵ. If *fstate*(ϵ) is the Dirac measure over the start state $\bar{q}$, then ϵ is called a *probabilistic execution.* We often write $\epsilon_{\sigma,q}$ for $\epsilon_{\sigma,\delta(q)}$, and we say that $\epsilon_{\sigma,q}$ is generated by σ and q.

Example 3.1 (the cone construction is rich). The cone construction produces a very rich set of measurable events. The event "action a occurs at least once," that is, the set of execution fragments where an action a occurs at least once, is measurable since it can be expressed as a union of cones and there are at most countably many cones in a PA. Similarly the event "action a occurs at least n times" is measurable for any natural number n. The event "action a occurs exactly n times" is measurable since it is the intersection of "action a occurs at least n times" with the complement of "action a occurs at least $n + 1$ times." Also the event "action a occurs finitely (infinitely) many times" is measurable since it is the countable union (intersection) of "action a occurs exactly (at least) n times." Similar arguments hold for occurrences of states rather than actions.

Any singleton set is measurable since for an infinite execution fragment α the set $\{\alpha\}$ is the intersection of the cones of all its finite prefixes, while for a finite execution fragment α the set $\{\alpha\}$ is the intersection of C_α with the complement of the union of the cones of the extensions of α. Thus, also the set of finite execution fragments is measurable, and the set of infinite execution fragments is measurable as well.

Observe that the probability of a finite execution fragment α is the probability that α occurs and then the computation terminates. Thus, the probability of the set of finite execution fragments represents the probability of termination in a probabilistic execution fragment. This leads to the idea that a probabilistic execution fragment should be called *finite* if the probability of the set of finite execution fragments is 1.

We now show how to obtain a probability measure over traces from a probabilistic execution fragment. The measurable space is the pair $(E^* \cup E^\omega, \mathcal{F})$, where $\mathcal{F}$ is the σ-field generated by cones of traces. More precisely, the cone of a finite trace β is defined by $C_\beta = \{\beta' \in E^* \cup E^\omega \mid \beta \leq \beta'\}$, where $\leq$ denotes the prefix ordering on sequences. It is easy to check that the *trace* function is measurable since the inverse image of a cone C_β is a union of cones, specifically those cones C_α such that $\beta \leq \mathit{trace}(\alpha)$, and since there are countably many finite execution fragments in a PA.

Given a probabilistic execution fragment ϵ, we define the *trace distribution* of ϵ, *tdist*(ϵ), to be the image measure of ϵ under *trace*. We denote the set of trace distributions of probabilistic executions of a PA $\mathcal{P}$ by *tdists*$(\mathcal{P})$. We define the *trace distribution preorder* relation on PAs by $\mathcal{P}_1 \leq_D \mathcal{P}_2$ iff $E_1 = E_2$ and *tdists*$(\mathcal{P}_1) \subseteq$ *tdists*$(\mathcal{P}_2)$.

An example of a measurable set of traces that is used extensively throughout the paper is the set $E^*a(E^* \cup E^\omega)$ of traces in which a specific action a occurs. We denote this set by $\diamond a$. The inverse image under *trace* of $\diamond a$ can be expressed as a disjoint union of cones of executions, namely the cones of the minimal executions with trace in $\diamond a$. Thus, we have the following proposition, whose elementary proof is omitted.

Proposition 3.3. *Let η be the trace distribution of a probabilistic execution ϵ of a PA $\mathcal{P}$, and let Θ_a be the set of finite executions of $\mathcal{P}$ with a single occurrence of* *action a whose last transition is labeled by a. Then*

$$\eta(\diamond a) = \sum_{\alpha \in \Theta_a} \epsilon(C_\alpha). \tag{2}$$

3.3. Combined, weak, and hyper transitions. We define three new kinds of transitions that play crucial roles in the paper. Informally, a combined transition is a convex combination of transitions that are labeled by the same action, a weak combined transition abstracts from internal computation and is obtained by performing several, possibly zero, combined transitions, while a hyper-transition is a generalization of combined transitions and weak combined transitions where the starting point is a measure over states rather than a single state.

3.3.1. Combined transitions. Let $\{q \xrightarrow{a} \mu_i\}_{i\in I}$ be a collection of transitions of a PA $\mathcal{P}$, and let $\{p_i\}_{i\in I}$ be a collection of probabilities such that $\sum_{i\in I} p_i = 1$. Then the triple $(q, a, \sum_{i\in I} p_i\mu_i)$ is called a *combined transition* of $\mathcal{P}$.

3.3.2. Weak transitions. Consider a probabilistic execution fragment ϵ of a PA $\mathcal{P}$, with first state $\delta(q)$, that assigns probability 1 to the set of all finite execution fragments with trace *trace*(β) for some $\beta \in A^*$. Let μ be the discrete measure on Q defined by $\mu(q') = \epsilon(\{\alpha \mid \mathit{lstate}(\alpha) = q'\})$. Then $q \overset{\beta}{\Longrightarrow} \mu$ is a *weak combined transition* of $\mathcal{P}$. We refer to ϵ as a *representation* of $q \overset{\beta}{\Longrightarrow} \mu$. Observe that the measure μ can be seen alternatively as the image measure of ϵ under *lstate*. This is an abuse of notation because *lstate* is not defined for infinite execution fragments; however, since ϵ assigns measure 1 to the set of finite execution fragments, we can extend the definition of *lstate* to infinite execution fragments for this purpose: for instance, we define the last state of any infinite execution fragment to be $\bar{q}$.

The notion of weak combined transition that we have just defined for PAs is a conservative extension of the corresponding notion defined for nondeterministic automata. Indeed, it is routine to check that whenever $q \overset{\beta}{\Longrightarrow} q'$ is a weak transition of a nondeterministic automaton $\mathcal{A}$, then $q \overset{\beta}{\Longrightarrow} \delta(q')$ is a weak combined transition of $\mathcal{A}$ viewed as a PA.

PROPOSITION 3.4. *Let $\{tr_i = (q, a, \mu_i)\}_{i\in I}$ be a collection of weak combined transitions of a PA $\mathcal{P}$, and let $\{p_i\}_{i\in I}$ be probabilities such that $\sum_{i\in I} p_i = 1$. Then $(q, a, \sum_{i\in I} p_i\mu_i)$, written $\sum_{i\in I} p_i tr_i$, is a weak combined transition of $\mathcal{P}$.*

Proof. For each $i \in I$, let ϵ_i be a representation of tr_i, and σ_i be a scheduler that, together with state q, induces ϵ_i. We omit the index set I in the rest of the proof. For each finite execution fragment α, let $N(\alpha) \triangleq \sum_i p_i\epsilon_i(C_\alpha)$. Define a new scheduler σ as follows:

$$\sigma(\alpha) = \begin{cases} \displaystyle\sum_i \frac{p_i\epsilon_i(C_\alpha)}{N(\alpha)}\sigma_i(\alpha) & \text{if } N(\alpha) > 0, \\ \text{arbitrary} & \text{otherwise.} \end{cases}$$

Informally, the weight that $\sigma(\alpha)$ gives to the choice $\sigma_i(\alpha)$ is the normalized probability with which σ_i contributes to the generation of α. Let ϵ be the probabilistic execution fragment induced by σ and q. Let α be a finite execution fragment of $\mathcal{P}$. We first prove by induction on the length of α that $\epsilon(C_\alpha) = N(\alpha)$. The base case is trivial since $\epsilon(C_q) = 1$ and for each i, $\epsilon_i(C_q) = 1$, which implies $N(q) = \sum_i p_i\epsilon_i(C_q) = 1$; similarly, for each state $q' \neq q$, $\epsilon(C_{q'}) = 0$ and for each i, $\epsilon_i(C_{q'}) = 0$. For the inductive step, let $\alpha = \alpha'a'q'$. If $\epsilon(C_{\alpha'}) = 0$, then, by induction, $N(\alpha') = \sum_i p_i\epsilon_i(C_{\alpha'}) = 0$, which implies that for each i, $p_i\epsilon_i(C_{\alpha'}) = 0$. By the definition of the measure of a cone, in (1), $\epsilon(C_\alpha) = 0$. Furthermore, for each i, if $p_i = 0$, then $p_i\epsilon_i(C_\alpha) = 0$ trivially, and if $p_i > 0$, then $\epsilon_i(C_{\alpha'}) = 0$, and by definition of measure of a cone, in (1), $\epsilon_i(C_\alpha) = 0$, which implies $p_i\epsilon_i(C_\alpha) = 0$. Thus, $N(\alpha) = 0$ as needed. If $\epsilon(C_{\alpha'}) > 0$, then, by the definition of the measure of a cone, in (1),

$$\epsilon(C_\alpha) = \epsilon(C_{\alpha'}) \sum_{tr \in D(a')} \sigma(\alpha')(tr)\mu_{tr}(q').$$

By the induction hypothesis, $N(\alpha') > 0$. Thus, by expanding $\sigma(\alpha')(tr)$ with the definition of σ, we obtain

$$\epsilon(C_\alpha) = \epsilon(C_{\alpha'}) \sum_{tr \in D(a')} \left(\sum_i \frac{p_i\epsilon_i(C_{\alpha'})}{N(\alpha')}\sigma_i(\alpha')(tr) \right) \mu_{tr}(q').$$

By standard algebraic manipulations (exchanges of sums and rearrangements of constants) we obtain

$$\epsilon(C_\alpha) = \frac{\epsilon(C_{\alpha'})}{N(\alpha')} \sum_i \sum_{tr \in D(a')} p_i\epsilon_i(C_{\alpha'})\sigma_i(\alpha')(tr)\mu_{tr}(q').$$

By induction, $\epsilon(C_{\alpha'}) = N(\alpha')$. Thus, by simplifying (removing) the leftmost term and rearranging constants, we obtain

$$\epsilon(C_\alpha) = \sum_i p_i \left(\epsilon_i(C_{\alpha'}) \sum_{tr \in D(a')} \sigma_i(\alpha')(tr)\mu_{tr}(q') \right).$$

Finally, by the definition of the measure of a cone, in (1), we get the desired equation

$$\epsilon(C_\alpha) = N(\alpha) = \sum_i p_i\epsilon_i(C_\alpha).$$

Thus, $\epsilon = \sum_i p_i\epsilon_i$. Since each ϵ_i assigns probability 1 to the set of finite execution fragments of $\mathcal{P}$ with trace *trace*(a), then so does ϵ. Furthermore, by Proposition 3.1.3, *lstate*$(\epsilon) = \sum_i p_i$ *lstate*(ϵ_i). That is, ϵ is a representation of a weak combined transition $(q, a, \sum_i p_i \textit{ lstate}(\epsilon_i))$, which, since *lstate*$(\epsilon_i) = \mu_i$, is the triplet $(q, a, \sum_i p_i \mu_i)$. Hence, $\sum_i p_i tr_i$ is a weak combined transition of $\mathcal{P}$. □

3.3.3. Hyper-transitions. Let $\mathcal{P}$ be a PA with $a \in A$, and let $\mu \in \textit{Disc}(Q)$. For each $q \in \textit{supp}(\mu)$, suppose that $q \xrightarrow{a} \mu_q$ is a combined transition of $\mathcal{P}$. Let μ' be $\sum_{q \in \textit{supp}(\mu)} \mu(q)\mu_q$. Then $\mu \xrightarrow{a} \mu'$ is called a *hyper-transition* of $\mathcal{P}$. Also, let $\beta \in A^*$, and for each $q \in \textit{supp}(\mu)$, suppose that $q \overset{\beta}{\Longrightarrow} \mu_q$ is a weak combined transition of $\mathcal{P}$. Let μ' be $\sum_{q \in \textit{supp}(\mu)} \mu(q)\mu_q$. Then $\mu \overset{\beta}{\Longrightarrow} \mu'$ is called a *weak hyper-transition* of $\mathcal{P}$.

We now prove two technical properties of weak hyper-transitions. The first property gives an alternative definition of weak hyper-transition and is used to prove the second property; the second property states that weak hyper-transitions can be concatenated. It will be used in section 6.2.

PROPOSITION 3.5. *There is a weak hyper-transition $\mu \overset{\beta}{\Longrightarrow} \mu'$ iff there is a scheduler σ such that $\epsilon_{\sigma,\mu}$ assigns probability 1 to the set of finite execution fragments with trace β, and lstate$(\epsilon_{\sigma,\mu}) = \mu'$. We say that $\epsilon_{\sigma,\mu}$ represents $\mu \overset{\beta}{\Longrightarrow} \mu'$.*

Proof. Let $\{q_i\}_I$ be an enumeration of the states in *supp*(μ). We prove the two implications separately.

$\Rightarrow$ For each i let σ_i be a scheduler such that ϵ_{σ_i,q_i} represents $q_i \overset{\beta}{\Longrightarrow} \mu_i$ and $\mu' = \sum_I \mu(q_i)\mu_i$. Let σ be a new scheduler defined as follows:

$$\sigma(\alpha) = \begin{cases} \sigma_i(\alpha) & \text{if } \mathit{fstate}(\alpha) = q_i \text{ for some } i \in I, \\ 0 & \text{otherwise.} \end{cases}$$

We first prove that $\epsilon_{\sigma,\mu} = \sum_i \mu(q_i)\epsilon_{\sigma_i,q_i}$ by demonstrating that $\epsilon_{\sigma,\mu}(C_\alpha) = \sum_i \mu(q_i)\epsilon_{\sigma_i,q_i}(C_\alpha)$ for each finite execution fragment α. The proof is by induction on the length of α. For the base case, let $\alpha = q$ for some state q. By the definition of the measure of a cone, $\epsilon_{\sigma,\mu}(C_\alpha) = \mu(q)$, and, for each i, $\epsilon_{\sigma_i,q_i}(C_\alpha)$ is 1 if $q = q_i$ and 0 otherwise. Thus, $\epsilon_{\sigma,\mu}(C_q) = \sum_i \mu(q_i)\epsilon_{\sigma_i,q_i}(C_q)$ trivially. For the inductive step, let α be $\alpha' a q$. If $\mathit{fstate}(\alpha') \notin \mathit{supp}(\mu)$, then trivially $\epsilon_{\sigma,\mu}(C_\alpha) = 0$ and, for each i, $\epsilon_{\sigma_i,q_i}(C_\alpha) = 0$. Thus, $\epsilon_{\sigma,\mu}(C_\alpha) = \sum_i \mu(q_i)\epsilon_{\sigma_i,q_i}(C_\alpha)$. If $\mathit{fstate}(\alpha') \in \mathit{supp}(\mu)$, then let j be the index of $\mathit{fstate}(\alpha')$. By the definition of the measure of a cone, $\epsilon_{\sigma,\mu}(C_\alpha) = \epsilon_{\sigma,\mu}(C_{\alpha'}) \sum_{tr\in D(a)} \sigma(\alpha')(tr)\mu_{tr}(q)$. By induction and the definition of σ,

$$\epsilon_{\sigma,\mu}(C_\alpha) = \sum_i \mu(i)\epsilon_{\sigma_i,q_i}(C_{\alpha'}) \sum_{tr\in D(a)} \sigma_j(\alpha')(tr)\mu_{tr}(q).$$

Since only $\epsilon_{\sigma_j,q_j}(C_{\alpha'})$ may be different from 0, we get

$$\epsilon_{\sigma,\mu}(C_\alpha) = \mu(q_j)\epsilon_{\sigma_j,q_j}(C_{\alpha'}) \sum_{tr\in D(a)} \sigma_j(\alpha')(tr)\mu_{tr}(q) = \mu(q_j)\epsilon_{\sigma_j,q_j}(C_\alpha).$$

For the same reason, $\sum_i \mu(q_i)\epsilon_{\sigma_i,q_i}(C_\alpha) = \mu(q_j)\epsilon_{\sigma_j,q_j}(C_\alpha)$. Thus, $\epsilon_{\sigma,\mu}(C_\alpha) = \sum_i \mu(i)\epsilon_{\sigma_i,q_i}(C_\alpha)$, as needed.

Since each ϵ_{σ_i,q_i} assigns probability 1 to the set of finite execution fragments with trace β, then also $\epsilon_{\sigma,\mu}$ assigns probability 1 to the set of finite execution fragments with trace β. Furthermore, by Proposition 3.1.3, $\mathit{lstate}(\epsilon_{\sigma,\mu}) = \sum_i \mu(q_i)\mathit{lstate}(\epsilon_{\sigma_i,q_i}) = \mu'$. Thus, $\mathit{lstate}(\epsilon_{\sigma,\mu})$ represents $\mu \overset{\beta}{\Longrightarrow} \mu'$.

$\Leftarrow$ Let σ be a scheduler that represents $\mu \overset{\beta}{\Longrightarrow} \mu'$. For each i, let $\sigma_i = \sigma$. Observe that for each finite execution fragment α,

$$\epsilon_{\sigma,\mu}(C_\alpha) = \mu(\mathit{fstate}(\alpha))\epsilon_{\sigma,\mathit{fstate}(\alpha)}(C_\alpha).$$

Thus, as in the previous case, $\epsilon_{\sigma,\mu} = \sum_i \mu(i)\epsilon_{\sigma_i,q_i}$. Let $q_i \overset{\beta}{\Longrightarrow} \mu_i$ be the weak transition represented by ϵ_{σ_i,q_i}. Since $\mu' = \mathit{lstate}(\epsilon_{\sigma,\mu})$ and for each i, $\mu_i = \mathit{lstate}(\epsilon_{\sigma_i,q_i})$, by Proposition 3.1.3, $\mu' = \sum_i \mu(q_i)\mu_i$. This suffices. □

PROPOSITION 3.6. *Suppose that* $\mu_1 \overset{\beta_1}{\Longrightarrow} \mu_2$ *and* $\mu_2 \overset{\beta_2}{\Longrightarrow} \mu_3$ *are weak hyper-transitions of a PA* $\mathcal{P}$. *Then* $\mu_1 \overset{\beta_1\beta_2}{\Longrightarrow} \mu_3$ *is a weak hyper-transition of* $\mathcal{P}$.

Proof. Let ϵ_1 and ϵ_2 be the probabilistic execution fragments that represent $\mu_1 \overset{\beta_1}{\Longrightarrow} \mu_2$ and $\mu_2 \overset{\beta_2}{\Longrightarrow} \mu_3$, respectively, and let σ_1 and σ_2 be the schedulers that generate ϵ_1 and ϵ_2, respectively. Let $N(\alpha) \triangleq \epsilon_1(C_\alpha) + \sum_{\alpha' < \alpha} \epsilon_1(\alpha')\epsilon_2(C_{\alpha \triangleright \alpha'} \mid C_{\alpha' \triangleright \alpha'})$, where we recall that $\epsilon_2(C_{\alpha \triangleright \alpha'} \mid C_{\alpha' \triangleright \alpha'})$ denotes the probability of $C_{\alpha \triangleright \alpha'}$ conditional on $C_{\alpha' \triangleright \alpha'}$. Define a new scheduler σ as follows:

$$\sigma(\alpha) = \begin{cases} \frac{\epsilon_1(C_\alpha)\sigma_1(\alpha) + \sum_{\alpha' \leq \alpha} \epsilon_1(\alpha')\epsilon_2(C_{\alpha \triangleright \alpha'} | C_{\alpha' \triangleright \alpha'})\sigma_2(\alpha \triangleright \alpha')}{N(\alpha)} & \text{if } N(\alpha) > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Informally, σ is a scheduler that represents a concatenation of ϵ_1 and ϵ_2. A finite execution fragment α can be reached in the concatenation of ϵ_1 and ϵ_2 either because ϵ_1 reaches it or because ϵ_1 terminates at a prefix of α from which ϵ_2 continues. The scheduler σ from α should behave according to σ_1 whenever α is reached in ϵ_1, and according to σ_2, with the appropriate argument, whenever α is reached in ϵ_2. The schedules from α must be weighted by the probabilities with which ϵ_1 and ϵ_2 lead to α. The term $N(\alpha)$ is a normalization factor whose computation is just technical. In the formal proof we have to show that σ is indeed a scheduler (in other words, the value of N is correct), and that σ generates the representation of $\mu_1 \overset{\beta_1\beta_2}{\Longrightarrow} \mu_3$. These proofs are mainly detailed algebraic manipulations. An interesting point of the proof is (4), which expresses the probability of a finite execution fragment in the concatenation of ϵ_1 and ϵ_2 in terms of the probabilities of finite execution fragments of ϵ_1 and ϵ_2.

We show that σ is a scheduler. That is, for each finite execution fragment α, $\sigma(\alpha)(D) \le 1$, or equivalently, $\epsilon_1(C_\alpha)\sigma_1(\alpha)(D)+\sum_{\alpha' \le \alpha}\epsilon_1(\alpha')\epsilon_2(C_{\alpha\triangleright\alpha'} \mid C_{\alpha'\triangleright\alpha'})\sigma_2(\alpha\triangleright\alpha')(D) \le N(\alpha)$, where the left-hand term is the numerator of the definition of σ applied to D. Since σ_2 is a scheduler, we know that $\sigma_2(\alpha\triangleright\alpha')(D) \le 1$. Also, observe that $\epsilon_1(C_\alpha)\sigma_1(\alpha)(D) = \epsilon_1(C_\alpha - \{\alpha\})$. Thus, it suffices to show that $\epsilon_1(C_\alpha - \{\alpha\}) + \sum_{\alpha' \le \alpha}\epsilon_1(\alpha')\epsilon_2(C_{\alpha\triangleright\alpha'} \mid C_{\alpha'\triangleright\alpha'}) \le N(\alpha)$. We separate from the sum the term with $\alpha' = \alpha$ and observe that $\epsilon_2(C_{\alpha\triangleright\alpha} \mid C_{\alpha\triangleright\alpha}) \le 1$. The new inequality, which suffices for our purposes, is $\epsilon_1(C_\alpha - \{\alpha\}) + \epsilon_1(\{\alpha\}) + \sum_{\alpha' < \alpha}\epsilon_1(\alpha')\epsilon_2(C_{\alpha\triangleright\alpha'} \mid C_{\alpha'\triangleright\alpha'}) \le N(\alpha)$. However, since $\epsilon_1(C_\alpha - \{\alpha\}) + \epsilon_1(\{\alpha\}) = \epsilon_1(C_\alpha)$, the inequality above is $N(\alpha) \le N(\alpha)$, which is trivially true.

Let ϵ be the probabilistic execution fragment generated by σ and μ_1. We show that ϵ represents $\mu_1 \overset{\beta_1\beta_2}{\Longrightarrow} \mu_3$ in three steps. First we show by induction on the length of a finite execution fragment α that $\epsilon(C_\alpha) = N(\alpha)$. The base case is trivial since $N(q) = \epsilon_1(C_q)$ by definition of N, and $\epsilon_1(C_q) = \epsilon(C_q) = \mu_1(q)$ by definition of ϵ_1 and ϵ since both measures are generated by μ_1. For the inductive step, let $\alpha = \alpha' a' q'$. If $\epsilon(C_{\alpha'}) = 0$, then by the definition of the measure of a cone, in (1), $\epsilon(C_\alpha) = 0$. We show that $N(\alpha) = 0$ as well. By induction, since $\epsilon(C_{\alpha'}) = 0$, $N(\alpha') = 0$. By definition of N, $\epsilon_1(C_{\alpha'}) = 0$, and $\epsilon_1(\alpha'')\epsilon_2(C_{\alpha'\triangleright\alpha''} \mid C_{\alpha''\triangleright\alpha''}) = 0$ for each $\alpha'' < \alpha'$. Then, by the definition of a conditional measure and of a measure of a cone, $\epsilon_1(C_\alpha) = 0$ and $\epsilon_1(\alpha'')\epsilon_2(C_{\alpha\triangleright\alpha''} \mid C_{\alpha''\triangleright\alpha''}) = 0$ for each $\alpha'' < \alpha'$. By $\epsilon_1(C_{\alpha'}) = 0$, also $\epsilon_1(\alpha') = 0$, and hence $\epsilon_1(\alpha')\epsilon_2(C_{\alpha\triangleright\alpha'} \mid C_{\alpha'\triangleright\alpha'}) = 0$. We have shown that all the terms of the definition of $N(\alpha)$ are 0, and thus $N(\alpha) = 0$ as needed. If $\epsilon(C_{\alpha'}) > 0$, then by expanding σ with its definition in (1), the definition of the measure of a cone, we get that $\epsilon(C_\alpha)/\epsilon(C_{\alpha'})$ equals

$$\sum_{tr \in D(a)} \frac{\epsilon_1(C_{\alpha'})\sigma_1(\alpha')(tr) + \sum_{\alpha'' \le \alpha'}\epsilon_1(\alpha'')\epsilon_2(C_{\alpha'\triangleright\alpha''} \mid C_{\alpha''\triangleright\alpha''})\sigma_2(\alpha' \triangleright \alpha'')(tr)}{N(\alpha')}\mu_{tr}(q').$$

By induction, $\epsilon(C_{\alpha'}) = N(\alpha')$, and thus the two terms can be simplified in the equation above. Then, by rearranging terms algebraically, we get

$$\epsilon(C_\alpha) = \sum_{tr \in D(a)} \epsilon_1(C_{\alpha'})\sigma_1(\alpha')(tr)\mu_{tr}(q')$$

$$+ \sum_{\alpha'' \le \alpha'} \epsilon_1(\alpha'') \sum_{tr \in D(a)} \epsilon_2(C_{\alpha'\triangleright\alpha''} \mid C_{\alpha''\triangleright\alpha''})\sigma_2(\alpha' \triangleright \alpha'')(tr)\mu_{tr}(q').$$

By the definition of the measure of a cone, the first term above is $\epsilon_1(C_\alpha)$. With a similar argument, after applying the definition of a conditional measure and distinguishing the cases where $\epsilon_2(C_{\alpha'' \rhd \alpha''}) = 0$, the second term above is $\sum_{\alpha'' \leq \alpha'} \epsilon_1(\alpha'')\epsilon_2(C_{\alpha \rhd \alpha''} \mid C_{\alpha'' \rhd \alpha''})$. Thus, we get

$$\epsilon(C_\alpha) = \epsilon_1(C_\alpha) + \sum_{\alpha'' \leq \alpha'} \epsilon_1(\alpha'')\epsilon_2(C_{\alpha \rhd \alpha''} \mid C_{\alpha'' \rhd \alpha''}).$$

Then it is enough to observe that the right-hand side of the equation above is $N(\alpha)$ since $\alpha'' \leq \alpha'$ iff $\alpha'' < \alpha$.

Second we show that for each finite execution fragment α,

$$\epsilon(\alpha) = \sum_{\alpha' \leq \alpha} \epsilon_1(\alpha')\epsilon_2(\alpha \rhd \alpha' \mid C_{\alpha' \rhd \alpha'}). \tag{4}$$

Observe that, by the definition of the cone σ-field, $\epsilon(\alpha) = \epsilon(C_\alpha) - \sum_{a \in A, q \in Q} \epsilon(C_{\alpha a q})$. By replacing the ϵ measures of cones with the definition of N in the equation above, we get

$$\begin{aligned}\epsilon(\alpha) = \epsilon_1(C_\alpha) &+ \sum_{\alpha' < \alpha} \epsilon_1(\alpha')\epsilon_2(C_{\alpha \rhd \alpha'} \mid C_{\alpha' \rhd \alpha'}) \\ &- \sum_{a \in A, q \in Q} \left(\epsilon_1(C_{\alpha a q}) + \sum_{\alpha' < \alpha a q} \epsilon_1(\alpha')\epsilon_2(C_{\alpha a q \rhd \alpha'} \mid C_{\alpha' \rhd \alpha'}) \right).\end{aligned}$$

We now use the terms $\epsilon_1(C_\alpha)$ and $-\sum_{a \in A, q \in Q} \epsilon_1(C_{\alpha a q})$ in the equation above to derive $\epsilon_1(\alpha)$, and similarly we use part of the other two terms to derive the following:

$$\epsilon(\alpha) = \epsilon_1(\alpha) + \sum_{\alpha' < \alpha} \epsilon_1(\alpha')\epsilon_2(\alpha \rhd \alpha' \mid C_{\alpha' \rhd \alpha'}) - \sum_{a \in A, q \in Q} \epsilon_1(\alpha)\epsilon_2(C_{\alpha a q \rhd \alpha} \mid C_{\alpha \rhd \alpha}).$$

Observe that the third term in the equation above is $\epsilon_1(\alpha)\epsilon_2((C_{\alpha \rhd \alpha} - \{\alpha \rhd \alpha\}) \mid C_{\alpha \rhd \alpha})$. Thus, by adding and subtracting the term $\epsilon_1(\alpha)\epsilon_2(\alpha \rhd \alpha \mid C_{\alpha \rhd \alpha})$ and rearranging algebraically, we get

$$\epsilon(\alpha) = \sum_{\alpha' \leq \alpha} \epsilon_1(\alpha')\epsilon_2(\alpha \rhd \alpha' \mid C_{\alpha' \rhd \alpha'}) + \epsilon_1(\alpha)\left(1 - \epsilon_2(C_{\alpha \rhd \alpha} \mid C_{\alpha \rhd \alpha})\right).$$

If $\epsilon_1(\alpha) = 0$, then (4) follows trivially. If $\epsilon_1(\alpha) > 0$, then, since ϵ_1 represents $\mu_1 \overset{\beta_1}{\Longrightarrow} \mu_2$, $\mu_2(q) > 0$, where q is *lstate*(α). Since ϵ_2 is generated by σ_2 and μ_2, $\epsilon_2(C_q) = \mu_2(q) > 0$. By definition of $\rhd$, $\alpha \rhd \alpha = q$, and thus $\epsilon_2(C_{\alpha \rhd \alpha}) > 0$. By the definition of the conditional measure, $\epsilon_2(C_{\alpha \rhd \alpha} \mid C_{\alpha \rhd \alpha}) = 1$, which leads again to (4).

Third we show that ϵ represents $\mu_1 \overset{\beta_1\beta_2}{\Longrightarrow} \mu_3$. Let α be such that $\epsilon(\alpha) > 0$. By (4) there exists a prefix α' of α such that $\epsilon_1(\alpha') > 0$ and $\epsilon_2(\alpha' \rhd \alpha) > 0$. Then *trace*$(\alpha') = \beta_1$ and *trace*$(\alpha \rhd \alpha') = \beta_2$. Thus, *trace*$(\alpha) = \beta_1\beta_2$. Let μ be *lstate*(ϵ). We are left to show that $\mu = \mu_3$. Consider a state q of Q. By the definition of an image measure and (4), $\mu(q) = \sum_{\alpha \mid lstate(\alpha) = q} \sum_{\alpha' \leq \alpha} \epsilon_1(\alpha')\epsilon_2(\alpha \rhd \alpha' \mid C_{\alpha' \rhd \alpha'})$. The sum above can be restructured as follows:

$$\mu(q) = \sum_{\alpha'} \sum_{\alpha \mid fstate(\alpha) = lstate(\alpha'), lstate(\alpha) = q} \epsilon_1(\alpha')\epsilon_2(\alpha \mid C_{\alpha' \rhd \alpha'}).$$

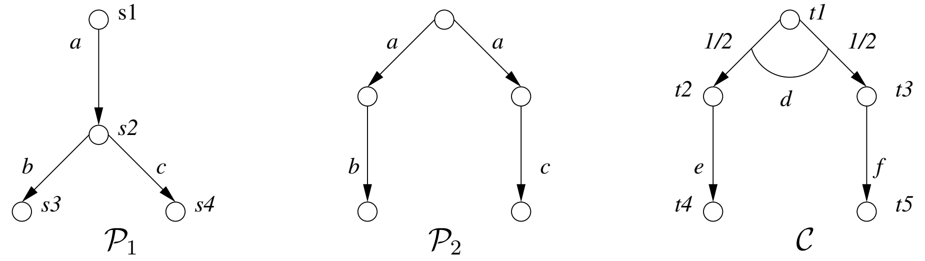


FIG. 3.1. *Trace distribution inclusion is not preserved by composition (without communication).*

We further partition the sums over $lstate(\alpha')$, thus getting

$$\mu(q) = \sum_{q' \in Q} \sum_{\alpha' | lstate(\alpha') = q'} \sum_{\alpha | fstate(\alpha) = q', lstate(\alpha) = q} \epsilon_1(\alpha')\epsilon_2(\alpha \mid C_{q'}).$$

Observe that, since $\mu_2 = lstate(\epsilon_1)$, if $\mu_2(q') = 0$, then there is no α' with last state q' such that $\epsilon_1(\alpha') > 0$. Thus we can restrict the first sum to those q' such that $\mu_2(q') > 0$. Also, if $\mu_2(q') > 0$, we have already concluded before that $\epsilon_2(C_{\alpha' \rhd \alpha'}) = \mu_2(q')$. Thus, we get

$$\mu(q) = \sum_{q' \in Q | \mu_2(q') > 0} \sum_{\alpha' | lstate(\alpha') = q'} \sum_{\alpha | fstate(\alpha) = q', lstate(\alpha) = q} \epsilon_1(\alpha')\epsilon_2(\alpha)/\mu_2(q').$$

Observe that the two inner sums can be exchanged. By definition of μ_2 it follows that $\sum_{\alpha' | lstate(\alpha') = q'} \epsilon_1(\alpha') = \mu_2(q')$. Thus, we get

$$\mu(q) = \sum_{q' \in Q | \mu_2(q') > 0} \sum_{\alpha | fstate(\alpha) = q', lstate(\alpha) = q} \epsilon_2(\alpha).$$

Following the same argument that we used to restrict the sum over q', we can remove such restriction, and thus we can remove the most external sum, leading to $\mu(q) = \sum_{\alpha | lstate(\alpha) = q} \epsilon_2(\alpha)$, which is the definition of $\mu_3(q)$. □

3.4. Composition. Two PAs, $\mathcal{P}_1$ and $\mathcal{P}_2$, are *compatible* if $H_1 \cap A_2 = A_1 \cap H_2 = \emptyset$. The *(parallel) composition* of two compatible PAs $\mathcal{P}_1$ and $\mathcal{P}_2$, denoted by $\mathcal{P}_1 \| \mathcal{P}_2$, is the PA $\mathcal{P} = (Q_1 \times Q_2, (\bar{q}_1, \bar{q}_2), E_1 \cup E_2, H_1 \cup H_2, D)$, where D is the set of triples $(q, a, \mu_1 \times \mu_2)$ such that, for $i \in \{1, 2\}$,

$$a \in A_i \Rightarrow (\pi_i(q), a, \mu_i) \in D_i \quad \text{and} \quad a \notin A_i \Rightarrow \mu_i = \delta(\pi_i(q)).$$

Let ϵ be a probabilistic execution (fragment) of $\mathcal{P}_1 \| \mathcal{P}_2$ and let $i \in \{1, 2\}$. Define $\pi_i(\epsilon)$, the ith projection of ϵ, to be the image measure under π_i of ϵ. It is easy to verify that the projection function is measurable. When convenient, we denote a projection by $\epsilon \lceil \mathcal{P}_i$, where $\mathcal{P}_i$ is the PA that appears in the ith position.

PROPOSITION 3.7. *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be compatible PAs, and let ϵ be a probabilistic execution (fragment) of $\mathcal{P}_1 \| \mathcal{P}_2$. Then for each $i \in \{1, 2\}$, $\pi_i(\epsilon)$ is a probabilistic execution (fragment) of $\mathcal{P}_i$.*

Proof. The proof follows by Propositions 4.3.4 and 4.3.5 of [32]. □

The trace distribution preorder is not preserved by composition [34, 37], as is shown by the following example.

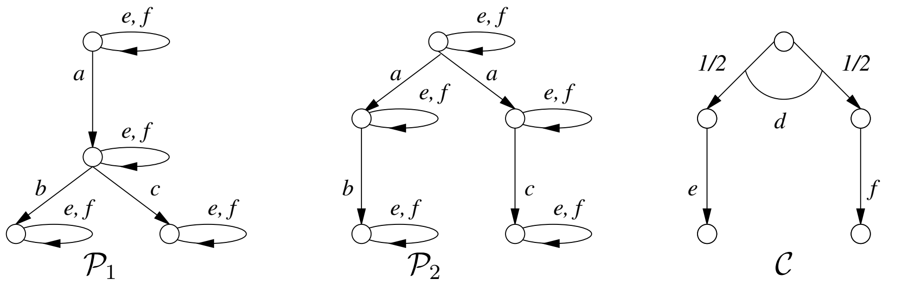


FIG. 3.2. *Trace distribution inclusion is not preserved by composition (with communication).*

Example 3.2 (failure of compositionality). Consider the two (nondeterministic) automata $\mathcal{P}_1$ and $\mathcal{P}_2$ of Figure 3.1. The two automata are trace equivalent, and it is easy to see that they are also trace distribution equivalent. Now consider the compositions $\mathcal{P}_1\|\mathcal{C}$ and $\mathcal{P}_2\|\mathcal{C}$, where $\mathcal{C}$ is the PA of Figure 3.1 and we assume that the actions of $\mathcal{C}$ are not shared with $\mathcal{P}_1$ and $\mathcal{P}_2$. It is possible to build a probabilistic execution of $\mathcal{P}_1\|\mathcal{C}$ as follows: first a is scheduled followed by d; then e or f is scheduled, depending on the outcome state of the transition labeled by d; finally, b or c is scheduled, depending on whether e or f was scheduled. Formally, we consider the probabilistic execution induced by the deterministic memoryless scheduler specified by the following partial function, where a transition is denoted by the unique action labeling it:

Q_1	$Q_\mathcal{C}$	$D_{\mathcal{P}_1\|\mathcal{C}}$
s_1	t_1	a
s_2	t_1	d
s_2	t_2	e
s_2	t_3	f
s_2	t_4	b
s_2	t_5	c

Thus, in the resulting trace distribution there is a total correlation between e, b and f, c, respectively. The same trace distribution cannot be obtained from $\mathcal{P}_2\|\mathcal{C}$ because after scheduling the transition labeled by a we are already bound to b or c, and thus the occurrence of b or c cannot be correlated to e or f in this case.

Example 3.2 may appear pathological since, in the probabilistic execution of $\mathcal{P}_1\|\mathcal{C}$ that correlates the choices between e and f and between b and c, a nondeterministic choice of $\mathcal{P}_1$ is resolved based on information that is not available to $\mathcal{P}_1$. This may lead us to propose a naive solution to the nonpreservation of trace distribution inclusion by parallel composition, where we require that each PA in a parallel composition be able to resolve its nondeterministic choices based on local knowledge only. However, a more elaborate example shows that this naive idea also does not work.

Example 3.3 (failure of compositionality). Consider the two automata $\mathcal{P}_1$ and $\mathcal{P}_2$ of Figure 3.2, which are essentially the automata of Example 3.2 where self-loop transitions labeled by e and f are added to each state. In this case the context $\mathcal{C}$ synchronizes with $\mathcal{P}_1$ and $\mathcal{P}_2$ on actions e and f, and $\mathcal{P}_1$ is able to learn which of e or f occurs, thus determining the correlation with b and c based on local knowledge only.

The solution of resolving nondeterminism based on local knowledge is adopted in [10] for a probabilistic extension of reactive modules; however, the idea of [10] cannot be extended easily to PAs because of key structural differences in the models: in PAs there is a total interleaving of the transitions taken by different PAs in a parallel composition, while in probabilistic reactive modules there are several independent *atoms* that are not forced to interleave. A direct adaptation of the idea of [10] to PAs would require drastic modifications of the model that go beyond the scope of this paper: transitions would have to be labeled by sets of actions and be structured in such a way that each action affects different parts of the state.

An alternative approach, followed in [32] and adopted in this paper, consists of defining a new *trace distribution precongruence* relation, denoted by $\leq_{DC}$, as the coarsest precongruence (for parallel composition) that is included in the trace distribution preorder $\leq_D$, and finding alternative characterizations of $\leq_{DC}$. It is known from [32] that there exists a simple context, called the *principal context*, that is sufficiently powerful to distinguish all PAs that are not in the trace distribution precongruence relation; alternatively, a testing scenario is proposed in [33].

In this paper we characterize $\leq_{DC}$ in terms of probabilistic simulation relations. Another simple alternative characterization of $\leq_{DC}$ that is useful for our study is given by the following proposition.

PROPOSITION 3.8. *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be PAs. Then $\mathcal{P}_1 \leq_{DC} \mathcal{P}_2$ iff for every PA $\mathcal{C}$ that is compatible with both $\mathcal{P}_1$ and $\mathcal{P}_2$, $\mathcal{P}_1\|\mathcal{C} \leq_D \mathcal{P}_2\|\mathcal{C}$.*

Proof. Define relation $\sqsubseteq$ such that $\mathcal{P}_1 \sqsubseteq \mathcal{P}_2$ iff for every PA $\mathcal{C}$ that is compatible with both $\mathcal{P}_1$ and $\mathcal{P}_2$, $\mathcal{P}_1\|\mathcal{C} \leq_D \mathcal{P}_2\|\mathcal{C}$.

Let $\mathcal{P}_1 \leq_{DC} \mathcal{P}_2$ and let $\mathcal{C}$ be a PA compatible with both $\mathcal{P}_1$ and $\mathcal{P}_2$. Since $\leq_{DC}$ is a precongruence by definition, then $\mathcal{P}_1\|\mathcal{C} \leq_{DC} \mathcal{P}_2\|\mathcal{C}$. Since, again by definition, $\leq_{DC}$ is included in $\leq_D$, then $\mathcal{P}_1\|\mathcal{C} \leq_D \mathcal{P}_2\|\mathcal{C}$. Thus, $\mathcal{P}_1 \sqsubseteq \mathcal{P}_2$, which implies that $\leq_{DC}$ is included in $\sqsubseteq$.

Conversely, observe that $\sqsubseteq$ is reflexive and transitive, and thus a preorder relation. Observe also that, by using a trivial context $\mathcal{C}$ with no external actions and no transitions, $\sqsubseteq$ is included in $\leq_D$. Finally, using the associativity of parallel composition, observe that $\sqsubseteq$ is preserved by parallel composition, and thus is a precongruence. This means that $\sqsubseteq$ is a precongruence included in $\leq_D$. Since $\leq_{DC}$ is the coarsest precongruence included in $\leq_D$, we get that $\sqsubseteq$ is included in $\leq_{DC}$. □

3.5. Simulation relations. The definitions of forward simulation and weak forward simulation in section 2 can be extended naturally to PAs [34]. However, Segala has shown [31] that the resulting simulations are not complete for $\leq_{DC}$, and has defined new candidate simulations. These new simulations relate states to probability measures on states.

In order to define the new simulations formally, we need two new concepts. First we show how to *lift* a relation between sets to a relation between measures over sets [18]. Let $R \subseteq X \times Y$. The *lifting* of R is a relation $R' \subseteq \mathit{Disc}(X) \times \mathit{Disc}(Y)$ such that $\mu_X \; R' \; \mu_Y$ iff there is a function $w : X \times Y \to [0,1]$ that satisfies the following:

1. If $w(x,y) > 0$, then $x \; R \; y$.
2. For each $x \in X$, $\sum_{y \in Y} w(x,y) = \mu_X(x)$.
3. For each $y \in Y$, $\sum_{x \in X} w(x,y) = \mu_Y(y)$.

We abuse notation slightly and denote the lifting of a relation R by R as well. Second, we define a *flattening* operation that converts a measure μ in $\mathit{Disc}(\mathit{Disc}(X))$ into a measure $\mathit{flatten}(\mu)$ in $\mathit{Disc}(X)$. Namely, we define $\mathit{flatten}(\mu) = \sum_{\rho \in \mathit{supp}(\mu)} \mu(\rho)\rho$.

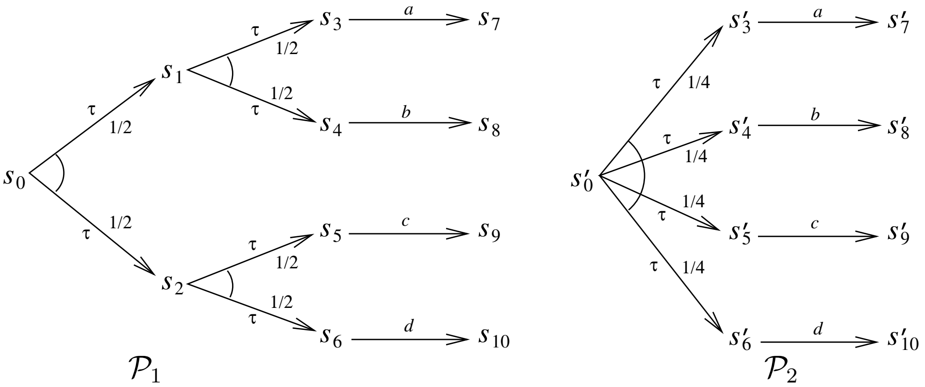


FIG. 3.3. *A forward probabilistic simulation between two PAs.*

We now define simulations for PAs. A relation $R \subseteq Q_1 \times Disc(Q_2)$ is a *probabilistic forward simulation* (resp., *weak probabilistic forward simulation*) from PA $\mathcal{P}_1$ to PA $\mathcal{P}_2$ iff $E_1 = E_2$ and both of the following hold:

1. $\bar{q}_1 \; R \; \delta(\bar{q}_2)$.
2. For each pair q_1, μ_2 such that $q_1 \; R \; \mu_2$ and each transition $q_1 \xrightarrow{a} \mu'_1$, there exists a measure $\xi'_2 \in Disc(Disc(Q_2))$ such that $\mu'_1 \; R \; \xi'_2$ and such that $\mu_2 \xrightarrow{a} \mathit{flatten}(\xi'_2)$ (resp., $\mu_2 \overset{a}{\Longrightarrow} \mathit{flatten}(\xi'_2)$) is a hyper-transition (resp., a weak hyper-transition) of $\mathcal{P}_2$.

We write $\mathcal{P}_1 \leq_{PF} \mathcal{P}_2$ (resp., $\mathcal{P}_1 \leq_{wPF} \mathcal{P}_2$) whenever there is a probabilistic forward simulation (resp., a weak probabilistic forward simulation) from $\mathcal{P}_1$ to $\mathcal{P}_2$.

Example 3.4 (forward probabilistic simulation). Figure 3.3 gives an example of two PAs that are in the kernel of probabilistic forward simulation. However, there would be no simulation from $\mathcal{P}_1$ to $\mathcal{P}_2$ if we did not allow states to be related to measures over states. The probabilistic forward simulation R from $\mathcal{P}_1$ to $\mathcal{P}_2$ relates each state of $\mathcal{P}_1$ with the Dirac measure over its primed version of $\mathcal{P}_2$, relates s_1 with the uniform measure over s'_3 and s'_4, and relates s_2 with the uniform measure over s'_5 and s'_6. The transition from s_0 can be simulated from s'_0 by scheduling the only transition enabled. Indeed, the target measure $\mathcal{U}(s'_3, s'_4, s'_5, s'_6)$ is the flattening of $\mathcal{U}(\mathcal{U}(s'_3, s'_4), \mathcal{U}(s'_5, s'_6))$, and it is easy to check that $\mathcal{U}(s_1, s_2) \; R \; \mathcal{U}(\mathcal{U}(s'_3, s'_4), \mathcal{U}(s'_5, s'_6))$.

Note that a forward simulation between nondeterministic automata is a probabilistic forward simulation between the two automata viewed as PAs, as described next.

PROPOSITION 3.9. *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be nondeterministic automata. Then*

1. *$\mathcal{A}_1 \leq_F \mathcal{A}_2$ iff $\mathcal{A}_1 \leq_{PF} \mathcal{A}_2$, and*
2. *$\mathcal{A}_1 \leq_{wF} \mathcal{A}_2$ iff $\mathcal{A}_1 \leq_{wPF} \mathcal{A}_2$.*

Proof. The left-to-right inclusions are easy since, given a (weak) forward simulation R from $\mathcal{A}_1$ to $\mathcal{A}_2$, it is routine to check that the relation $R' \triangleq \{(q_1, \delta(q_2)) \mid q_1 \; R \; q_2\}$ is a (weak) probabilistic forward simulation from $\mathcal{A}_1$ to $\mathcal{A}_2$.

For the converse implication, let R be a (weak) probabilistic forward simulation from $\mathcal{A}_1$ to $\mathcal{A}_2$. Define a relation $R' \triangleq \{(q_1, q_2) \mid \exists_\mu q_1 \; R \; \mu, q_2 \in supp(\mu)\}$. We show that R' is a (weak) forward simulation from $\mathcal{A}_1$ to $\mathcal{A}_2$.

The start condition is trivial since $\bar{q}_1 \; R \; \delta(\bar{q}_2)$, and thus $\bar{q}_1 \; R' \; \bar{q}_2$. For the step condition, let $q_1 \; R' \; q_2$, and let $q_1 \xrightarrow{a} q'_1$. By the definition of R', there exists a measure

μ such that $q_1 \; R \; \mu$ and $q_2 \in \mathit{supp}(\mu)$. Since R is a (weak) forward simulation, there exists a hyper-transition $\mu \xrightarrow{a} \mu'$ (a weak hyper-transition $\mu \overset{a}{\Longrightarrow} \mu'$), where μ' is the flattening of some measure μ'' such that $\delta(q_1') \; R \; \mu''$. By the definition of hyper-transition, there is a combined transition $q_2 \xrightarrow{a} \mu_2$ (a weak combined transition $q_2 \overset{a}{\Longrightarrow} \mu_2$) such that $\mathit{supp}(\mu_2) \subseteq \mathit{supp}(\mu')$. For the strong case, let $q_2 \xrightarrow{a} q_2'$ be one of the transitions of D_2 that are combined in $q_2 \xrightarrow{a} \mu_2$. Then, $q_2' \in \mathit{supp}(\mu_2)$. For the weak case, consider a scheduler σ that generates $q_2 \overset{a}{\Longrightarrow} \mu_2$, and build a new scheduler σ' that on input α stops (does not return any transition) if $\sigma(\alpha)$ stops with some nonzero probability and chooses any transition in $\mathit{supp}(\sigma(\alpha))$ that reduces the distance from a stopping point otherwise. This leads to a weak transition $q_2 \overset{a}{\Longrightarrow} q_2'$, where $q_2' \in \mathit{supp}(\mu_2)$. We now show that $q_1' \; R' \; q_2'$, which suffices. Since $q_2' \in \mathit{supp}(\mu_2)$, and since $\mathit{supp}(\mu_2) \subseteq \mathit{supp}(\mu')$, then $q_2' \in \mathit{supp}(\mu')$. Since $\mu' = \mathit{flatten}(\mu'')$, then q_2' is also in the support of some measure $\rho \in \mathit{supp}(\mu'')$. Thus, $q_1' \; R \; \rho$, and, by the definition of R', $q_1' \; R' \; q_2'$, as needed. □

PROPOSITION 3.10. *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be PAs. Then the following hold:*

1. *If $\mathcal{P}_1 \leq_{PF} \mathcal{P}_2$, then $\mathcal{P}_1 \leq_{wPF} \mathcal{P}_2$.*
2. *If $H_1 = H_2 = \emptyset$, then $\mathcal{P}_1 \leq_{PF} \mathcal{P}_2$ iff $\mathcal{P}_1 \leq_{wPF} \mathcal{P}_2$.*
3. *If $\mathcal{P}_1 \leq_{wPF} \mathcal{P}_2$, then $\mathcal{P}_1 \leq_{DC} \mathcal{P}_2$.*

Proof. The first item follows from the fact that a combined transition is a special case of a weak combined transition; the second item follows from the fact that in the absence of internal actions a weak combined transition is a combined transition. The proof of the third item is quite involved, and we refer the reader to Proposition 8.7.1 of [32]. The main idea is to use the weak probabilistic forward simulation from $\mathcal{P}_1$ to $\mathcal{P}_2$ to build, for each probabilistic execution of $\mathcal{P}_1$, a corresponding probabilistic execution of $\mathcal{P}_2$ with the same trace distribution. □

3.6. Tree-structured PAs. A *path* of a PA $\mathcal{P}$ is a finite sequence $\gamma = q_0 a_1 \mu_1 q_1 a_2 \mu_2 q_2 \ldots q_n$ of alternating states, actions, and distribution over states, starting with the start state of $\mathcal{P}$ such that for each nonfinal i, $q_i \xrightarrow{a_{i+1}} \mu_{i+1}$ and $q_{i+1} \in \mathit{supp}(\mu_{i+1})$. We write $\mathit{lstate}(\gamma)$ to denote q_n and $\mathit{paths}(\mathcal{P})$ for the set of all paths of $\mathcal{P}$. We say that $\mathcal{P}$ is *tree-structured* if each state is reached via a unique path. Tree-structured PAs are characterized uniquely by the property that all states are reachable, the start state does not occur in the target of any transition, and each of the other states occurs in the target of exactly one transition. Tree-structured nondeterministic automata are also characterized uniquely by this property, albeit for a different notion of transition.

If a PA is tree-structured, then its underlying nondeterministic automaton is also tree-structured. The following example shows that the converse does not hold.

Example 3.5 (non–tree-structured PAs). Figure 3.4 shows a PA that is not tree-structured, as state q' can be reached via two different paths. The underlying nondeterministic automaton is tree-structured, however, since the only way to reach state q' is via the execution qaq'.

The *unfolding* of a PA $\mathcal{P}$, denoted by $\mathit{Unfold}(\mathcal{P})$, is the tree-structured PA $\mathcal{Q}$ obtained from $\mathcal{P}$ by unfolding its transition graph into a tree. Formally,

- $Q_{\mathcal{Q}} = \mathit{paths}(\mathcal{P})$,
- $\bar{q}_{\mathcal{Q}} = \bar{q}_{\mathcal{P}}$,
- $E_{\mathcal{Q}} = E_{\mathcal{P}}$,
- $H_{\mathcal{Q}} = H_{\mathcal{P}}$, and
- $D_{\mathcal{Q}} = \{(\gamma, a, \mu) \mid (\exists \mu')[(\mathit{lstate}(\gamma), a, \mu') \in D_{\mathcal{P}} \wedge (\forall q \in \mathit{supp}(\mu'))[\mu'(q) = \mu(\gamma a \mu' q)]]\}$.

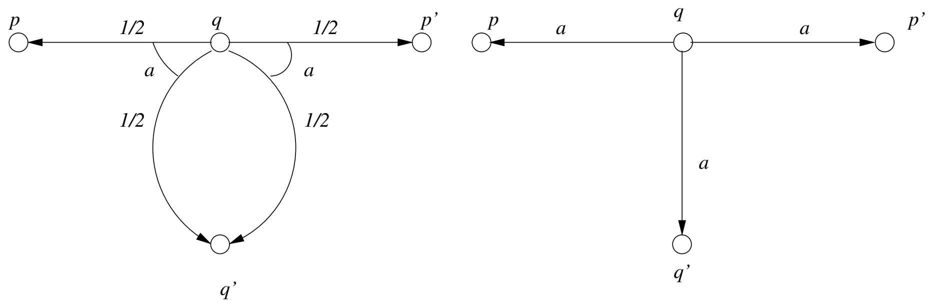


FIG. 3.4. *The PA on the left is not tree-structured even though its underlying nondeterministic automaton on the right is.*

PROPOSITION 3.11. $\mathcal{P} \equiv_{PF}$ *Unfold*$(\mathcal{P})$.

Proof. It is easy to check that the relation R, where $\alpha \; R \; \delta(q)$ iff *lstate*$(\alpha) = q$ is a probabilistic forward simulation from *Unfold*$(\mathcal{P})$ to $\mathcal{P}$ and that the "inverse" of R, that is, the relation R' such that $q \; R' \; \delta(\alpha)$ iff *lstate*$(\alpha) = q$, is a probabilistic forward simulation from $\mathcal{P}$ to *Unfold*$(\mathcal{P})$. □

PROPOSITION 3.12. $\mathcal{P} \equiv_{DC}$ *Unfold*$(\mathcal{P})$.

Proof. The proof follows by Proposition 3.11 and Proposition 3.10, parts 1 and 3. □

3.7. Truncations and continuations. We now define two simple constructions on probabilistic execution fragments that will be useful for our proofs. Specifically, we define the truncation of a probabilistic execution fragment, which is the result of stopping the computation at some designated points, and the continuation of a probabilistic execution fragment, which represents the rest of a probabilistic execution fragment after some finite execution fragment has occurred.

Let ϵ be a probabilistic execution fragment of a PA $\mathcal{P}$, generated by some scheduler σ, and let Θ be a set of finite execution fragments of $\mathcal{P}$. Define the *truncation of* ϵ *at* Θ to be the same as ϵ except that no transition is scheduled from all the Θ places, that is, the probabilistic execution fragment ϵ', with the same start state as ϵ, generated by a new scheduler σ' such that $\sigma'(\alpha) = \sigma(\alpha)$ if $\alpha \notin \Theta$ and $\sigma'(\alpha)(D) = 0$ if $\alpha \in \Theta$.

PROPOSITION 3.13. *The definition of truncation of a probabilistic execution fragment* ϵ *is independent of the choice of the inducing scheduler.*

Proof. Let μ be the first state of ϵ, and let σ_1, σ_2 be two schedulers that, together with μ, induce ϵ. Let Θ be a set of finite execution fragments of $\mathcal{P}$, and let σ'_1, σ'_2 be the schedulers built from σ_1, σ_2, respectively, according to the definition of truncation. Let ϵ_1, ϵ_2 be the induced probabilistic execution fragments, and suppose by contradiction that $\epsilon_1 \neq \epsilon_2$. Then there exists a finite execution α such that $\epsilon_1(C_\alpha) \neq \epsilon_2(C_\alpha)$. Consider such a finite execution α of minimum length. Observe that $|\alpha| > 0$ since $\epsilon(C_q) = \epsilon_1(C_q) = \epsilon_2(C_q) = \mu(q)$ for each state $q \in Q$. Thus, $\alpha = \alpha' a' q'$ for some α', a', q', where $\epsilon_1(C_{\alpha'}) = \epsilon_2(C_{\alpha'})$. We distinguish two cases.

Case 1. If $\alpha' \in \Theta$, then, by the definitions of σ'_1 and σ'_2, $\sigma'_1(\alpha')(D) = \sigma'_2(\alpha')(D) = 0$. Thus, $\epsilon_1(C_\alpha) = \epsilon_2(C_\alpha) = 0$, a contradiction.

Case 2. If $\alpha' \notin \Theta$, then, by the definitions of σ'_1 and σ'_2, $\sigma'_1(\alpha') = \sigma_1(\alpha')$ and

$\sigma_2'(\alpha') = \sigma_2(\alpha')$. Then,

$$\begin{aligned}
\epsilon_1(C_\alpha) &= \epsilon_1(C_{\alpha'}) \textstyle\sum_{tr \in D(a')} \sigma_1'(\alpha')(tr)\mu_{tr}(q') \quad \text{(by (1))}\\
&= \epsilon_2(C_{\alpha'}) \textstyle\sum_{tr \in D(a')} \sigma_1(\alpha')(tr)\mu_{tr}(q')\\
&\qquad \text{(by } \sigma_1'(\alpha') = \sigma_1(\alpha'),\ \epsilon_1(C_{\alpha'}) = \epsilon_2(C_{\alpha'})\text{)}\\
&= \epsilon_2(C_{\alpha'}) \textstyle\sum_{tr \in D(a')} \sigma_2(\alpha')(tr)\mu_{tr}(q') \quad \text{(by } \sigma_1 \text{ and } \sigma_2 \text{ induce } \epsilon\text{)}\\
&= \epsilon_2(C_{\alpha'}) \textstyle\sum_{tr \in D(a')} \sigma_2'(\alpha')(tr)\mu_{tr}(q') \quad \text{(by } \sigma_2'(\alpha') = \sigma_2(\alpha')\text{)}\\
&= \epsilon_2(C_\alpha) \quad \text{(by (1))},
\end{aligned}$$

again a contradiction. □

Let ϵ be a probabilistic execution fragment of a PA $\mathcal{P}$, generated by a scheduler σ, and let α be a finite execution fragment with $\mathit{fstate}(\alpha) \in \mathit{supp}(\mathit{fstate}(\epsilon))$. Define $\epsilon \rhd \alpha$, the *continuation of ϵ after prefix α*, to be the probabilistic execution fragment generated by the following scheduler σ' from $\mathit{lstate}(\alpha)$:

$$\sigma'(\alpha') = \begin{cases} \sigma(\alpha \frown \alpha') & \text{if } \mathit{fstate}(\alpha') = \mathit{lstate}(\alpha),\\ 0 & \text{otherwise,} \end{cases}$$

where by 0 we denote the identically 0 function.

PROPOSITION 3.14. *The definition of $\epsilon \rhd \alpha$ is independent of the choice of the inducing scheduler.*

Proof. The proof proceeds in a manner similar to that of Proposition 3.13. Let μ be the first state of ϵ, and let σ_1, σ_2 be two schedulers that, together with μ, induce ϵ. Let q' be $\mathit{lstate}(\alpha)$. Let σ_1', σ_2' be the schedulers built from σ_1, σ_2, respectively, according to the definition of $\epsilon \rhd \alpha$. Let ϵ_1, ϵ_2 be the induced probabilistic execution fragments from q', and suppose by contradiction that $\epsilon_1 \neq \epsilon_2$. Then there exists a finite execution α' such that $\epsilon_1(C_{\alpha'}) \neq \epsilon_2(C_{\alpha'})$. Consider such a finite execution α' of minimum length. Observe that $|\alpha'| > 0$ since $\epsilon(C_{q'}) = \epsilon_1(C_{q'}) = \epsilon_2(C_{q'}) = 1$ and, for each state $q'' \neq q'$, $\epsilon(C_{q''}) = \epsilon_1(C_{q''}) = \epsilon_2(C_{q''}) = 0$. Thus, $\alpha' = \alpha'' a'' q''$ for some α'', a'', q'', where $\epsilon_1(C_{\alpha''}) = \epsilon_2(C_{\alpha''})$. We distinguish two cases.

Case 1. If $\mathit{fstate}(\alpha'') \neq q'$, then, by the definitions of ϵ_1 and ϵ_2, $\epsilon_1(C_{\alpha'}) = \epsilon_2(C_{\alpha'}) = 0$, a contradiction.

Case 2. If $\mathit{fstate}(\alpha'') = q'$, then, by the definitions of σ_1' and σ_2', $\sigma_1'(\alpha'') = \sigma_1(\alpha \frown \alpha'')$ and $\sigma_2'(\alpha'') = \sigma_2(\alpha \frown \alpha'')$. Then,

$$\begin{aligned}
\epsilon_1(C_{\alpha'}) &= \epsilon_1(C_{\alpha''}) \textstyle\sum_{tr \in D(a'')} \sigma_1'(\alpha'')(tr)\mu_{tr}(q'') \quad \text{(by (1))}\\
&= \epsilon_2(C_{\alpha''}) \textstyle\sum_{tr \in D(a'')} \sigma_1(\alpha \frown \alpha'')(tr)\mu_{tr}(q'')\\
&\qquad \text{(by } \sigma_1'(\alpha'') = \sigma_1(\alpha \frown \alpha'') \text{ and } \epsilon_1(C_{\alpha''}) = \epsilon_2(C_{\alpha''})\text{)}\\
&= \epsilon_2(C_{\alpha''}) \textstyle\sum_{tr \in D(a'')} \sigma_2(\alpha \frown \alpha'')(tr)\mu_{tr}(q'') \quad \text{(by } \sigma_1 \text{ and } \sigma_2 \text{ induce } \epsilon\text{)}\\
&= \epsilon_2(C_{\alpha''}) \textstyle\sum_{tr \in D(a'')} \sigma_2'(\alpha'')(tr)\mu_{tr}(q'') \quad \text{(by } \sigma_2'(\alpha'') = \sigma_2(\alpha \frown \alpha'')\text{)}\\
&= \epsilon_2(C_{\alpha'}) \quad \text{(by (1))},
\end{aligned}$$

again a contradiction. □

The following proposition relates the continuation of ϵ after some prefix α with ϵ itself. In practice it states that $\epsilon \rhd \alpha$ is closely related to $\epsilon \mid C_\alpha$.

PROPOSITION 3.15. *Let ϵ be a probabilistic execution fragment of a PA $\mathcal{P}$, and let α be a finite execution fragment of $\mathcal{P}$. Then, for each finite execution α' with $\mathit{lstate}(\alpha) = \mathit{fstate}(\alpha')$, $\epsilon(C_{\alpha \frown \alpha'}) = \epsilon(C_\alpha) \cdot (\epsilon \rhd \alpha)(C_{\alpha'})$.*

Proof. The proof follows easily by induction on the length of α' from the definition of the probability of a cone. □

4. Tester automata and observer schedulers.

The proofs of our completeness results rely on a special context for a PA, which we call its *tester PA*. The tester automaton, $\mathit{tester}(\mathcal{P})$, of a PA $\mathcal{P}$ can observe the states that $\mathcal{P}$ goes through and the transitions that are scheduled during a probabilistic execution. This information is revealed by means of externally visible transitions of $\mathit{tester}(\mathcal{P})$ with the help of a specific scheduler, called the *observer*, which synchronizes $\mathcal{P}$ with its tester. In this section we present the constructions of the tester and observer and prove some results about the resulting trace distributions.

Informally, the tester of a PA $\mathcal{P}$ is a PA $\mathcal{C}$ whose states include a distinguished start state, all the states of $\mathcal{P}$, and all the transitions of $\mathcal{P}$. Automaton $\mathcal{C}$ has a special transition from its own start state, $\bar{q}_{\mathcal{C}}$, to the start state of $\mathcal{P}$, $\bar{q}_{\mathcal{P}}$, labeled by $\bar{q}_{\mathcal{P}}$. Also, from every state q of $\mathcal{P}$, $\mathcal{C}$ has a uniform transition labeled by ch ("choose") to the set of transitions of $\mathcal{P}$ that begin in state q. Finally, for every transition tr of $\mathcal{P}$ and every state q in the support of μ_{tr}, $\mathcal{C}$ has a transition labeled by q from tr to q.

DEFINITION 4.1. *The* tester PA *of a PA $\mathcal{P}$, denoted by $\mathit{tester}(\mathcal{P})$, is a PA $\mathcal{C} = (Q_{\mathcal{C}}, \bar{q}_{\mathcal{C}}, E_{\mathcal{C}}, H_{\mathcal{C}}, D_{\mathcal{C}})$, where*

- $Q_{\mathcal{C}} = \{\bar{q}_{\mathcal{C}}\} \cup Q_{\mathcal{P}} \cup D_{\mathcal{P}}$,
- $E_{\mathcal{C}} = Q_{\mathcal{P}} \cup \{ch\}$,
- $H_{\mathcal{C}} = \emptyset$, *and*
- $D_{\mathcal{C}} = \{(\bar{q}_{\mathcal{C}}, \bar{q}_{\mathcal{P}}, \delta(\bar{q}_{\mathcal{P}}))\} \cup$
  $\{(q, ch, \mathcal{U}(\{tr \in D_{\mathcal{P}} \mid \mathit{source}(tr) = q\})) \mid q \in Q_{\mathcal{P}} \wedge q \rightarrow\} \cup$
  $\{(tr, q, \delta(q)) \mid tr \in D_{\mathcal{P}}, q \in \mathit{supp}(\mu_{tr})\}$.

Observe that the tester of an ordinary nondeterministic automaton enables at most one transition from each state, and dually, the tester of an automaton that enables at most one transition from each state is a nondeterministic automaton. This observation, together with the results that we prove later in the paper, imply that a fully probabilistic context is enough to observe the branching structure of a nondeterministic automaton.

PROPOSITION 4.2. *The following hold:*

1. *The tester of a nondeterministic automaton is fully probabilistic; that is, it enables at most one transition from each state.*
2. *The tester of a fully probabilistic automaton is a nondeterministic automaton; that is, it contains only transitions whose target measures are Dirac.*

Proof. For the first item, observe that the only states of $\mathit{tester}(\mathcal{P})$ that may enable more than one transition are of the form $tr \in D_{\mathcal{P}}$, which enable one transition for each state in $\mathit{supp}(\mu_{tr})$; however, the size of $\mathit{supp}(\mu_{tr})$ is 1 in a nondeterministic automaton.

For the second item, observe that the only states of $\mathit{tester}(\mathcal{P})$ that may enable non-Dirac transitions are of the form $q \in Q_{\mathcal{P}}$, which may enable a transition labeled by ch to a uniform measure over the set of transitions enabled from q in $\mathcal{P}$; however, there is at most one transition enabled from q in a fully probabilistic automaton. □

We assume without loss of generality that a PA $\mathcal{P}$ and its tester do not have any actions in common (otherwise we can simply rename states of $\mathcal{P}$ to achieve our goal), and thus $\mathcal{P}$ and its tester are compatible.

Since $\mathit{tester}(\mathcal{P})$ and $\mathcal{P}$ share no actions, merely composing $\mathit{tester}(\mathcal{P})$ with $\mathcal{P}$ does not ensure that $\mathit{tester}(\mathcal{P})$ faithfully emulates the behavior of $\mathcal{P}$. However, an appropriate scheduler can synchronize the two automata and ensure such an emulation, which will be sufficient for our purposes. Given a PA $\mathcal{P}$, we define a specific scheduler σ for $\mathcal{P}\|tester(\mathcal{P})$, called the *observer* of $\mathcal{P}$, that synchronizes the two automata so that the internal structure of $\mathcal{P}$ is visible in the trace. Specifically, the scheduler σ starts by scheduling the transition of $tester(\mathcal{P})$ from the start state of $tester(\mathcal{P})$ to the start state of $\mathcal{P}$, leading to state $(\bar{q}, \bar{q})$, which is of the form (q, q). Then σ repeats the following as long as $q \rightarrow$:

1. Schedule the *ch* transition of $tester(\mathcal{P})$, thus choosing a transition *tr* of $\mathcal{P}$.
2. Schedule transition *tr* of $\mathcal{P}$, leading $\mathcal{P}$ to a new state q'.
3. Schedule the transition of $tester(\mathcal{P})$ labeled by the state q', resulting in the state (q', q'), which is again of the form (q, q).

DEFINITION 4.3. *The* observer *of a PA $\mathcal{P}$, which we denote by observer$(\mathcal{P})$, is a deterministic (almost memoryless) scheduler for $\mathcal{P}\|tester(\mathcal{P})$ that bases its decisions on the last state and sometimes the last action of its argument according to the following table. Here, $\mathcal{C}$ denotes tester$(\mathcal{P})$, q is any state such that $q \rightarrow$, tr is any transition in $D_{\mathcal{P}}$, and q' is any state in μ_{tr}.*

$Q_{\mathcal{P}}$	$Q_{tester(\mathcal{P})}$	Last action	$D_{\mathcal{P}\|tester(\mathcal{P})}$
$\bar{q}_{\mathcal{P}}$	$\bar{q}_{tester(\mathcal{P})}$		$\bar{q}_{\mathcal{P}}$-labeled transition of $tester(\mathcal{P})$
q	q		*ch*-labeled transition of $tester(\mathcal{P})$
q	*tr*	*ch*	transition *tr* of $\mathcal{P}$
q'	*tr*	not *ch*	q'-labeled transition of $tester(\mathcal{P})$

Scheduler $observer(\mathcal{P})$ and start state $(\bar{q}_{\mathcal{P}}, \bar{q}_{tester(\mathcal{P})})$ induce a trace distribution for $\mathcal{P}\|tester(\mathcal{P})$ where all states and external actions of $\mathcal{P}$ appear explicitly.

DEFINITION 4.4. *The* observation *of a PA $\mathcal{P}$, denoted by observation$(\mathcal{P})$, is the trace distribution induced by observer$(\mathcal{P})$ and $(\bar{q}_{\mathcal{P}}, \bar{q}_{tester(\mathcal{P})})$.*

Remark 4.5. The *ch*-labeled transitions of a tester are defined to lead to uniform measures over states of $tester(\mathcal{P})$ that represent transitions of $\mathcal{P}$. This is well defined since we have assumed that nondeterministic and probabilistic automata are finite branching. From the technical point of view, however, the proofs of this paper rely on the fact that the transitions labeled by *ch* assign nonzero probability (not necessarily the same probability) to each one of the options that are available in a nondeterministic choice. Thus, it would be possible to remove the finite branching restriction from the definition of automata and modify the definition of a tester automaton so that, whenever there are countably many transitions from a state q, the corresponding *ch*-labeled transition of the tester assigns a nonuniform measure to the transitions enabled from q, for example, a Poisson distribution after enumerating all possible transitions enabled from q. We have chosen not to deal with countable branching automata in this paper because it would complicate proofs without adding much insight.

We state and prove some properties of $observation(\mathcal{P})$. The first property, given in (5), says that the cone of traces beginning with the starting state of $\mathcal{P}$ has probability 1. The second property, (6), says that for any state q of $\mathcal{P}$ from which some transition is enabled and for each finite trace β of $\mathcal{P}\|tester(\mathcal{P})$, the probability of the cone of traces beginning with βq is the same as the probability of the cone beginning with $\beta q\, ch$; that is, once βq occurs, the probability that *ch* follows is 1. The third property, (7), says that for any state q of $\mathcal{P}$ and for each finite trace β of $\mathcal{P}\|tester(\mathcal{P})$, the probability of the cone of traces beginning with $\beta q\, ch$ is the same as the sum of the probabilities of the cones beginning with $\beta q\, ch\, \beta'$, where β' represents one single step of $\mathcal{P}$ from q; that is, once *ch* occurs, one of the transitions of $\mathcal{P}$ that are enabled from q is exposed. The right-hand side of (7) consists of two parts dealing with external and internal transitions, respectively.

PROPOSITION 4.6. *The trace distribution $\eta = observation(\mathcal{P})$ induced by the observer of a PA $\mathcal{P}$ satisfies the following three properties, for all finite traces β of $\mathcal{P}\|tester(\mathcal{P})$ and for all states q of $\mathcal{P}$:*

$$\eta(C_{\bar{q}}) = 1, \tag{5}$$

$$q \rightarrow \quad \Longrightarrow \quad \eta(C_{\beta q}) = \eta(C_{\beta q\, ch}), \tag{6}$$

$$\eta(C_{\beta q\, ch}) = \sum_{(a,q') \mid a \in E, q \xrightarrow{a} q'} \eta(C_{\beta q\, ch\, aq'}) + \sum_{q' \mid (\exists a) a \in H, q \xrightarrow{a} q'} \eta(C_{\beta q\, ch\, q'}). \tag{7}$$

Proof. Equation (5) follows from the fact that $observer(\mathcal{P})$ schedules action $\bar{q}$ immediately. Equation (6) follows from the fact that, after scheduling action q, thus leading to a state of the form (q, q), $observer(\mathcal{P})$ immediately schedules action ch if q enables at least one transition. Equation (7) follows from the fact that, after scheduling ch, $observer(\mathcal{P})$ schedules one of the transitions of $\mathcal{P}$ that are enabled from q, say $q \xrightarrow{a} \mu$, followed by a transition of $tester(\mathcal{P})$ labeled by a state in $supp(\mu)$. □

The following technical properties will be needed in the proofs of section 6. The first property, in (8), says that the probability of observing a state q' reachable in a tree-structured PA $\mathcal{P}$ with a single transition, say tr, from another state q is the probability of observing q, divided by the number of transitions enabled from q, and multiplied by the probability of reaching q' in tr, the only transition that may lead to q' since $\mathcal{P}$ is tree-structured. Indeed, q' can be observed only if q is observed (probability of observing q), the transition tr is chosen (factor $1/k$ since transitions are chosen uniformly), and the chosen transition leads to q'. The second property, (9), is similar to the first one, where the probability of observing q and scheduling the transition tr is replaced by the probability of observing any state in the target of tr.

PROPOSITION 4.7. *Let $\mathcal{P}$ be a tree-structured PA, and let η be $observation(\mathcal{P})$. Let $tr = (q, a, \mu)$ be a transition of $\mathcal{P}$. Let k be the number of transitions that are enabled from q in $\mathcal{P}$, and let q' be a state in $supp(\mu)$. Then the following properties hold:*

$$\eta(\diamond q') = \frac{\eta(\diamond q)}{k} \mu(q'), \tag{8}$$

$$\eta(\diamond q') = \left(\sum_{q'' \in supp(\mu)} \eta(\diamond q'') \right) \mu(q'). \tag{9}$$

Proof. Let σ be the observer of $\mathcal{P}$, and let ϵ_σ be the probabilistic execution induced by σ. Since $\mathcal{P}$ is tree-structured, the set Θ_q contains a single execution α. Indeed, by the definition of tree-structured, there is only one execution in $\mathcal{P}$ ending with state q, and σ simply interleaves this execution with transitions labeled by ch, by the names of the transitions of $\mathcal{P}$ that are needed to reach q, and by the names of the states that are reached. Similarly, $\Theta_{q'}$ contains a single execution α'.

Once state q is reached, σ schedules action ch, reaching state tr of $tester(\mathcal{P})$ with probability $1/k$. Then, σ schedules transition tr, reaching state q' in $\mathcal{P}$ with probability $\mu(q')$, and finally σ schedules the transition of $tester(\mathcal{P})$ labeled by q'. Thus, $\epsilon_\sigma(C_{\alpha'}) = \epsilon_\sigma(C_\alpha)(1/k)\mu(q')$. Then (8) follows by (2).

By summing over *supp*(μ) in (8), we get

$$\sum_{q'' \in supp(\mu)} \eta(\diamond q'') = \frac{\eta(\diamond q)}{k} \sum_{q'' \in supp(\mu)} \mu(q''). \tag{10}$$

Observe that $\sum_{q'' \in supp(\mu)} \mu(q'') = 1$. Hence, (10) simplifies to

$$\sum_{q'' \in supp(\mu)} \eta(\diamond q'') = \frac{\eta(\diamond q)}{k}. \tag{11}$$

Substitution of (11) into (8) gives us (9), as needed. □

5. Characterizations of $\leq_{DC}$ for nondeterministic automata.

In this section, we present our characterization theorems for $\leq_{DC}$ for nondeterministic automata: Theorem 5.2 characterizes $\leq_{DC}$ in terms of $\leq_F$, for nondeterministic automata without internal actions, and Theorem 5.4 characterizes $\leq_{DC}$ in terms of $\leq_{wF}$, for arbitrary nondeterministic automata. In each case, we prove the result first for tree-structured nondeterministic automata and then extend it to the non–tree-structured case via unfolding. The interesting direction for each of these results is the completeness direction, showing that $\mathcal{A}_1 \leq_{DC} \mathcal{A}_2$ implies the existence of a simulation relation from $\mathcal{A}_1$ to $\mathcal{A}_2$.

The strategy that we use to prove our completeness results is also applied in many other full abstraction results; see, for example, [5, 14]. By Proposition 3.8, $\mathcal{A}_1 \leq_{DC} \mathcal{A}_2$ implies that $\mathcal{A}_1 \leq_D \mathcal{A}_2$ for all contexts $\mathcal{C}$. Thus it suffices to construct a specific context $\mathcal{C}$ with the property that the trace distributions of $\mathcal{A}_1 \| \mathcal{C}$ contain all information about $\mathcal{A}_1$ that is preserved by the simulation preorder. More specifically, we compose $\mathcal{A}_1$ with the context $\mathcal{C} = \mathit{tester}(\mathcal{A}_1)$ and consider just a single trace distribution of the composed system, namely $\mathit{observation}(\mathcal{A}_1)$, the one generated by $\mathit{observer}(\mathcal{A}_1)$. We show, for any other nondeterministic automaton $\mathcal{A}_2$, that if the composition $\mathcal{A}_2 \| \mathit{tester}(\mathcal{A}_1)$ generates the trace distribution $\mathit{observation}(\mathcal{A}_1)$, then $\mathcal{A}_2$ actually simulates $\mathcal{A}_1$ in a strong sense. Namely, whenever $\mathcal{A}_1$ reaches some state q_1, $\mathcal{A}_2$ can reach a corresponding state q_2 from which it generates the same trace distribution. The formalities of the proof are intricate, in part because states of $\mathcal{A}_1$ also show up as states of $\mathit{tester}(\mathcal{A}_1)$ and within the trace distribution of $\mathit{observation}(\mathcal{A}_1)$. In the proof we try to be very explicit about the roles of states of $\mathcal{A}_1$, but we also warn the reader to be alert to this potential source of confusion.

5.1. Nondeterministic automata without internal actions.

We begin by considering nondeterministic automata without internal actions. We first consider tree-structured nondeterministic automata.

PROPOSITION 5.1. *Let $\mathcal{A}_1$, $\mathcal{A}_2$ be nondeterministic automata without internal actions such that $\mathcal{A}_1$ is tree-structured. Then $\mathcal{A}_1 \leq_{DC} \mathcal{A}_2$ implies $\mathcal{A}_1 \leq_F \mathcal{A}_2$.*

Proof. Assume that $\mathcal{A}_1 \leq_{DC} \mathcal{A}_2$. Let $\mathcal{C}$ be $\mathit{tester}(\mathcal{A}_1)$ and η be $\mathit{observation}(\mathcal{A}_1)$, that is, the trace distribution of $\mathcal{A}_1 \| \mathcal{C}$ induced by the scheduler $\mathit{observer}(\mathcal{A}_1)$. Since $\mathcal{A}_1 \leq_{DC} \mathcal{A}_2$ implies $\mathcal{A}_1 \| \mathcal{C} \leq_D \mathcal{A}_2 \| \mathcal{C}$, Proposition 3.8 implies that η is also a trace distribution of $\mathcal{A}_2 \| \mathcal{C}$. That is, there exists a probabilistic execution ϵ of $\mathcal{A}_2 \| \mathcal{C}$, induced by some scheduler σ_2, such that $\mathit{tdist}(\epsilon) = \eta$.

For each state q_1 in Q_1, let Θ_{q_1} be the set of finite executions of $\mathcal{A}_2 \| \mathcal{C}$ whose last transition is labeled by q_1. For each state q_2 of $\mathcal{A}_2$, let Θ_{q_1,q_2} be the set of executions in Θ_{q_1} whose last state is the pair (q_2, q_1).

Define a relation R on $Q_1 \times Q_2$ as follows: $q_1 \; R \; q_2$ iff there exists a finite execution α in Θ_{q_1,q_2} such that $\epsilon(C_\alpha) > 0$. We claim that R is a forward simulation from $\mathcal{A}_1$ to $\mathcal{A}_2$.

For the start condition, we must show that $\bar{q}_1 \; R \; \bar{q}_2$. Consider the start state $(\bar{q}_2, \bar{q}_\mathcal{C})$ of $\mathcal{A}_2 \| \mathcal{C}$. Since there are no internal actions in $\mathcal{A}_2$ or $\mathcal{C}$, and since, by (5) from Proposition 4.6, $\eta(C_{\bar{q}_1}) = 1$, the only action that is scheduled initially by σ_2 is $\bar{q}_1$, leading to state $(\bar{q}_2, \bar{q}_1)$. Thus, the finite execution $\alpha = (\bar{q}_2, \bar{q}_\mathcal{C})\bar{q}_1(\bar{q}_2, \bar{q}_1)$ is an element of $\Theta_{\bar{q}_1,\bar{q}_2}$ such that $\epsilon(C_\alpha) > 0$, as needed.

For the step condition, assume $q_1 \; R \; q_2$ and let $q_1 \xrightarrow{a}_1 q_1'$ be a transition of $\mathcal{A}_1$, which we denote by *tr* for convenience. We exhibit a matching transition $q_2 \xrightarrow{a}_2 q_2'$.

By the definition of R, there exists a finite execution α in Θ_{q_1,q_2} such that $\epsilon(C_\alpha) > 0$. Since Θ_{q_1,q_2} is a subset of Θ_{q_1}, by definition of Θ_{q_1}, $\mathit{trace}(\alpha) = \beta q_1$ for some finite trace β. Therefore, $\eta(C_{\beta q_1}) > 0$. Since q_1 enables at least one transition in $\mathcal{A}_1$, specifically transition *tr*, (6) from Proposition 4.6 implies that $\eta(C_{\beta q_1 \, ch}) = \eta(C_{\beta q_1})$. Then, since $\mathcal{A}_2$ and $\mathcal{C}$ have no internal actions, σ_2 schedules action *ch* from α with probability 1.

By the definition of *tester*$(\mathcal{A}_1)$, the transition labeled by *ch* leaving from state q_1 of $\mathcal{C}$ leads to state *tr* with probability > 0. Hence, $\epsilon(C_{\alpha \, ch \, (q_2, tr)}) > 0$. By (7) from Proposition 4.6, where only the first term of the right-hand side is nonzero due to the absence of internal actions, $\eta(C_{\beta q_1 \, ch}) = \sum_{(a,q') | a \in E, q_1 \xrightarrow{a} q'} \eta(C_{\beta q_1 \, ch \, a q'})$. Hence, σ_2 must extend $\alpha \, ch \, (q_2, tr)$ with two steps labeled by an action and a state of $\mathcal{A}_1$, respectively, where the action and the state are compatible with one of the transitions of $\mathcal{A}_1$ that are enabled from q_1. Since state *tr* of $\mathcal{C}$ enables only action q_1', and since, by the tree-structure of $\mathcal{A}_1$, a is uniquely determined by q_1', the action and state scheduled by σ_2 are a and q_1'. Therefore, there exists a state q_2' of $\mathcal{A}_2$ such that the execution $\alpha' = \alpha \, ch \, (q_2, tr) a (q_2', tr) q_1' (q_2', q_1')$ is an execution in $\Theta_{q_1',q_2'}$ such that $\epsilon(C_{\alpha'}) > 0$. Then $q_1' \; R \; q_2'$ and $q_2 \xrightarrow{a} q_2'$, as needed. □

Now we present our result for general (non–tree-structured) nondeterministic automata without internal actions.

THEOREM 5.2. *Let $\mathcal{A}_1$, $\mathcal{A}_2$ be nondeterministic automata without internal actions. Then $\mathcal{A}_1 \leq_{DC} \mathcal{A}_2$ iff $\mathcal{A}_1 \leq_F \mathcal{A}_2$.*

Proof. First we prove soundness of forward simulations:

$$\begin{aligned} \mathcal{A}_1 \leq_F \mathcal{A}_2 \quad &\Rightarrow \quad \mathcal{A}_1 \leq_{PF} \mathcal{A}_2 \quad \text{(Proposition 3.9.1)} \\ &\Rightarrow \quad \mathcal{A}_1 \leq_{wPF} \mathcal{A}_2 \quad \text{(Proposition 3.10.1)} \\ &\Rightarrow \quad \mathcal{A}_1 \leq_{DC} \mathcal{A}_2 \quad \text{(Proposition 3.10.3).} \end{aligned}$$

Next we establish completeness:

$$\begin{aligned} \mathcal{A}_1 \leq_{DC} \mathcal{A}_2 \quad &\Rightarrow \quad \mathit{Unfold}(\mathcal{A}_1) \leq_F \mathcal{A}_1 \leq_{DC} \mathcal{A}_2 \quad \text{(Proposition 2.4)} \\ &\Rightarrow \quad \mathit{Unfold}(\mathcal{A}_1) \leq_{DC} \mathcal{A}_1 \leq_{DC} \mathcal{A}_2 \quad \text{(as in soundness proof)} \\ &\Rightarrow \quad \mathit{Unfold}(\mathcal{A}_1) \leq_{DC} \mathcal{A}_2 \quad (\leq_{DC} \text{ is transitive}) \\ &\Rightarrow \quad \mathit{Unfold}(\mathcal{A}_1) \leq_F \mathcal{A}_2 \quad \text{(Proposition 5.1)} \\ &\Rightarrow \quad \mathcal{A}_1 \leq_F \mathit{Unfold}(\mathcal{A}_1) \leq_F \mathcal{A}_2 \quad \text{(Proposition 2.4)} \\ &\Rightarrow \quad \mathcal{A}_1 \leq_F \mathcal{A}_2 \quad (\leq_F \text{ is transitive}). \qquad \square \end{aligned}$$

5.2. Nondeterministic automata with internal actions. Next we extend the results of section 5.1 to nondeterministic automata that may include internal actions. The proofs are analogous to those in section 5.1. The difference is that, in several places in the proof of Proposition 5.3, we need to reason about multistep extensions of executions instead of single-step extensions. Again, we begin with tree-structured nondeterministic automata.

Proposition 5.3. *Let $\mathcal{A}_1$, $\mathcal{A}_2$ be nondeterministic automata such that $\mathcal{A}_1$ is tree-structured. Then $\mathcal{A}_1 \leq_{DC} \mathcal{A}_2$ implies $\mathcal{A}_1 \leq_{wF} \mathcal{A}_2$.*

Proof. Assume that $\mathcal{A}_1 \leq_{DC} \mathcal{A}_2$. Let $\mathcal{C}$ be $\mathit{tester}(\mathcal{A}_1)$ and η be $\mathit{observation}(\mathcal{A}_1)$, that is, the trace distribution of $\mathcal{A}_1 \| \mathcal{C}$ induced by the scheduler $\mathit{observer}(\mathcal{A}_1)$. Define the scheduler σ_2, the probabilistic execution ϵ, and the Θ sets as in the proof of Proposition 5.1.

The definition of R is slightly different: $q_1 \; R \; q_2$ iff there exists a state q_2' such that $q_2 \implies q_2'$ and there exists $\alpha \in \Theta_{q_1,q_2'}$ such that $\epsilon(C_\alpha) > 0$. We claim that R is a weak forward simulation from $\mathcal{A}_1$ to $\mathcal{A}_2$.

For the start condition, we must show that $\bar{q}_1 \; R \; \bar{q}_2$. By Item 1 of Proposition 4.6, $\eta(C_{\bar{q}_1}) = 1$. This means that there exists a finite execution fragment α of $\mathcal{A}_2 \| \mathcal{C}$ with trace $\bar{q}_1$ that ends with action $\bar{q}_1$ such that $\epsilon(C_\alpha) > 0$. By definition of $\mathcal{C}$, the last state of α is $(q_2, \bar{q}_1)$ for some state q_2 satisfying $\bar{q}_2 \implies q_2$. By definition of R, $\bar{q}_1 \; R \; \bar{q}_2$, as needed.

For the step condition, assume $q_1 \; R \; q_2$ and let $q_1 \xrightarrow{a}_1 q_1'$ be a transition of $\mathcal{A}_1$, which we denote by tr. We exhibit a matching weak transition $q_2 \overset{a}{\implies}_2 q_2'$.

By definition of R, there exists a state q_2'' of $\mathcal{A}_2$ such that $q_2 \implies q_2''$, and there exists a finite execution α in $\Theta_{q_1,q_2''}$ such that $\epsilon(C_\alpha) > 0$. Since $\Theta_{q_1,q_2''}$ is a subset of Θ_{q_1}, by definition of Θ_{q_1}, $\mathit{trace}(\alpha) = \beta q_1$ for some finite trace β. Therefore, $\eta(C_{\beta q_1}) > 0$. Since q_1 enables at least one transition in $\mathcal{A}_1$, specifically transition tr, (6) from Proposition 4.6 implies that $\eta(C_{\beta q_1 \, ch}) = \eta(C_{\beta q_1})$. Thus, there exists an execution fragment α' of $\mathcal{A}_2 \| \mathcal{C}$ with trace ch such that $\epsilon(C_{\alpha \frown \alpha'}) > 0$. Furthermore, since, by definition of $\mathcal{C} = \mathit{tester}(\mathcal{A}_1)$, the transition of $\mathcal{C}$ labeled by ch that leaves from state q_1 leads to state tr with nonzero probability, we can assume that the last state of α' is of the form (q', tr) for some state q' of $\mathcal{A}_2$.

Recall from above that $\eta(C_{\beta q_1 \, ch}) > 0$. By (7) from Proposition 4.6, $\eta(C_{\beta q_1 \, ch}) = \sum_{(a,q') | a \in E, q \xrightarrow{a} q'} \eta(C_{\beta q_1 \, ch \, aq'}) + \sum_{q' | (\exists a) a \in H, q_1 \xrightarrow{a} q'} \eta(C_{\beta q_1 \, ch \, q'})$. Hence, σ_2 must extend $\alpha \frown \alpha'$ in such a way that the first or the first two external actions are compatible with one of the transitions of $\mathcal{A}_1$ that are enabled from q_1. (The number of external actions depends on whether the compatible transition of $\mathcal{A}_1$ is labeled by an internal or external action.) Since state tr of $\mathcal{C}$ enables only action q_1', and since, by the tree-structure of $\mathcal{A}_1$, a is uniquely determined by q_1', the first or first two external actions of $\mathcal{A}_2 \| \mathcal{C}$ scheduled by σ_2 are either q_1' or aq_1', depending on whether a is internal or external. Thus, there exists an execution fragment α'' of $\mathcal{A}_2 \| \mathcal{C}$, with trace $\mathit{trace}(aq_1')$, such that $\epsilon(C_{\alpha \frown \alpha' \frown \alpha''}) > 0$. Furthermore, we can assume that the last transition of α'' is labeled by q_1' (simply truncate α'' otherwise).

Let (q_2', q_1') be the last state of α''. Then, $\alpha \frown \alpha' \frown \alpha'' \in \Theta_{q_1',q_2'}$, thus showing that $q_1' \; R \; q_2'$. It remains to show that $q_2 \overset{a}{\implies} q_2'$. For this, it suffices to recall that $q_2 \implies q_2''$ and observe that $q_2'' \overset{a}{\implies} q_2'$ since the execution fragment $(\alpha' \frown \alpha'') \lceil \mathcal{A}_2$ has trace $\mathit{trace}(a)$, first state q_2'', and last state q_2'. ◻

Using the same approach as before, we may eliminate the assumption in Proposition 5.3 that $\mathcal{A}_1$ is tree-structured.

Theorem 5.4. *Let $\mathcal{A}_1$, $\mathcal{A}_2$ be nondeterministic automata. Then $\mathcal{A}_1 \leq_{DC} \mathcal{A}_2$ iff $\mathcal{A}_1 \leq_{wF} \mathcal{A}_2$.*

Proof. This proof is analogous to that of Theorem 5.2. First we prove soundness of weak forward simulations:

$$\begin{aligned} \mathcal{A}_1 \leq_{wF} \mathcal{A}_2 \quad &\Rightarrow \quad \mathcal{A}_1 \leq_{wPF} \mathcal{A}_2 \quad \text{(Proposition 3.9.2)} \\ &\Rightarrow \quad \mathcal{A}_1 \leq_{DC} \mathcal{A}_2 \quad \text{(Proposition 3.10.3).} \end{aligned}$$

Now we prove completeness:

$$\begin{aligned} \mathcal{A}_1 \leq_{DC} \mathcal{A}_2 \quad &\Rightarrow \quad \mathit{Unfold}(\mathcal{A}_1) \leq_F \mathcal{A}_1 \leq_{DC} \mathcal{A}_2 \quad \text{(Proposition 2.4)} \\ &\Rightarrow \quad \mathit{Unfold}(\mathcal{A}_1) \leq_{DC} \mathcal{A}_1 \leq_{DC} \mathcal{A}_2 \\ & \quad \text{(as in proof Theorem 5.2, soundness part)} \\ &\Rightarrow \quad \mathit{Unfold}(\mathcal{A}_1) \leq_{DC} \mathcal{A}_2 \quad (\leq_{DC} \text{ is transitive}) \\ &\Rightarrow \quad \mathit{Unfold}(\mathcal{A}_1) \leq_{wF} \mathcal{A}_2 \quad \text{(Proposition 5.3)} \\ &\Rightarrow \quad \mathcal{A}_1 \leq_F \mathit{Unfold}(\mathcal{A}_1) \leq_{wF} \mathcal{A}_2 \quad \text{(Proposition 2.4)} \\ &\Rightarrow \quad \mathcal{A}_1 \leq_{wF} \mathit{Unfold}(\mathcal{A}_1) \leq_{wF} \mathcal{A}_2 \quad \text{(Proposition 2.3.1)} \\ &\Rightarrow \quad \mathcal{A}_1 \leq_{wF} \mathcal{A}_2 \quad (\leq_{wF} \text{ is transitive}). \quad \square \end{aligned}$$

6. Characterizations of $\leq_{DC}$ for PAs.

Now we present our characterization theorems for $\leq_{DC}$ for PAs: Theorem 6.3 characterizes $\leq_{DC}$ in terms of $\leq_{PF}$, for PAs without internal actions, and Theorem 6.5 characterizes $\leq_{DC}$ in terms of $\leq_{wPF}$, for arbitrary PAs. Again, we give the results first for tree-structured PAs and extend them by unfolding. As before, the interesting direction is the completeness direction, showing that $\mathcal{P}_1 \leq_{DC} \mathcal{P}_2$ implies the existence of a simulation relation from $\mathcal{P}_1$ to $\mathcal{P}_2$. Our proofs of completeness for PAs are analogous to those for nondeterministic automata.

6.1. PAs without internal actions.

We first consider tree-structured PAs.

PROPOSITION 6.1. *Let $\mathcal{P}_1$, $\mathcal{P}_2$ be PAs without internal actions such that $\mathcal{P}_1$ is tree-structured. Then $\mathcal{P}_1 \leq_{DC} \mathcal{P}_2$ implies $\mathcal{P}_1 \leq_{PF} \mathcal{P}_2$.*

Proof. Assume that $\mathcal{P}_1 \leq_{DC} \mathcal{P}_2$. Let $\mathcal{C}$ be $\mathit{tester}(\mathcal{P}_1)$ and η be $\mathit{observation}(\mathcal{P}_1)$, that is, the trace distribution of $\mathcal{P}_1 \| \mathcal{C}$ induced by the scheduler $\mathit{observer}(\mathcal{P}_1)$. Define the scheduler σ_2, the probabilistic execution ϵ, and the Θ sets as in the proof of Proposition 5.1.

Define a relation R as follows: $q_1 \; R \; \mu_2$ iff $\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha) > 0$ and for each state $q_2 \in Q_2$,

$$\mu_2(q_2) = \frac{\sum_{\alpha \in \Theta_{q_1,q_2}} \epsilon(C_\alpha)}{\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha)}. \tag{12}$$

That is, the measure μ_2 describes probabilities of the various Θ_{q_1,q_2}'s relative to Θ_{q_1}. Note that the equation above is well defined since, by the tree-structure of $\mathcal{P}_1$, all the cones represented by Θ_{q_1} are disjoint, and thus $\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha) \leq 1$. We claim that R is a probabilistic forward simulation from $\mathcal{P}_1$ to $\mathcal{P}_2$.

Before proving that R is a probabilistic forward simulation we make several observations:

1. Relation R is a function from Q_1 to $\mathit{Disc}(Q_2)$. Indeed, if $\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha) > 0$, then there exists exactly one measure that satisfies (12). Furthermore, given the construction of η and the fact that $\mathcal{P}_1$ is tree-structured (i.e., all states are reachable), every state q_1 of Q_1 occurs with some positive probability in η. Thus, since η is induced by ϵ, $\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha) > 0$ for all states q_1 of Q_1.
2. If $q_1 \; R \; \mu_2$, then, for each state $q_2 \in Q_2$ and each execution $\alpha \in \Theta_{q_1,q_2}$,

$$\epsilon(C_\alpha) > 0 \quad \Rightarrow \quad q_2 \in \mathit{supp}(\mu_2). \tag{13}$$

That is, the execution α occurs with nonzero probability in ϵ only if μ_2 assigns nonzero probability to q_2. This property is a direct consequence of (12).

3. For each transition $q_1 \xrightarrow{a} \mu'_1$ of $\mathcal{P}_1$, the following equation holds:

$$\mu'_1(q'_1) = \frac{\sum_{\alpha\in\Theta_{q'_1}} \epsilon(C_\alpha)}{\sum_{q\in supp(\mu'_1)\,,\,\alpha\in\Theta_q} \epsilon(C_\alpha)}. \tag{14}$$

That is, the relative probabilities of the states of $supp(\mu'_1)$ in ϵ are given by μ'_1. This result follows by instantiating (9) from Proposition 4.7 with $q_1 \xrightarrow{a} \mu'_1$ to derive the probability of a state q'_1 in the support of μ'_1, and by replacing the diamond expressions according to (2) from Proposition 3.3.

4. For each transition $tr = q_1 \xrightarrow{a} \mu'_1$ of $\mathcal{P}_1$, the following equation holds:

$$\sum_{\alpha\in\Theta_{q_1}} \epsilon(C_\alpha) = k \sum_{q\in supp(\mu'_1)\,,\,\alpha\in\Theta_q} \epsilon(C_\alpha), \tag{15}$$

where k is the number of transitions of $\mathcal{P}_1$ enabled from q_1. That is, the probability of reaching q_1 in ϵ is k times the probability of reaching q_1 and scheduling tr. Informally, transition tr is scheduled only if state q_1 is reached and the outcome of the following transition labeled by ch is tr, which happens with probability $1/k$. The reason why $\sum_{q\in supp(\mu'_1)\,,\,\alpha\in\Theta_q} \epsilon(C_\alpha)$ is the probability of reaching q_1 and scheduling tr is that states from $supp(\mu'_1)$ can occur only after q_1 has occurred and tr is reached (see the definition of tester automaton and of observer of a tester automaton), and furthermore states from $supp(\mu'_1)$ occur with probability 1 once tr is reached (see (7) from Proposition 4.6).

This follows by instantiating (8) from Proposition 4.7 with tr, replacing the diamond expressions according to (2) from Proposition 3.3, summing over $supp(\mu'_1)$, observing that $\sum_{q'_1\in supp(\mu'_1)} \mu'_1(q'_1) = 1$, and deriving $\sum_{\alpha\in\Theta_{q_1}} \epsilon(C_\alpha)$ from the resulting equation.

We are now ready to show that R is a probabilistic forward simulation. For the start condition, we must show that $\bar{q}_1 \; R \; \delta(\bar{q}_2)$.

Consider the start state $(\bar{q}_2, \bar{q}_{\mathcal{C}})$ of $\mathcal{P}_2\|\mathcal{C}$. Since there are no internal actions in $\mathcal{P}_2$ or $\mathcal{C}$, and since, by (5) from Proposition 4.6, $\eta(C_{\bar{q}_1}) = 1$, the only action that is scheduled initially by σ_2 is $\bar{q}_1$, leading to state $(\bar{q}_2, \bar{q}_1)$ with probability 1. Thus, the finite execution $\alpha = (\bar{q}_2, \bar{q}_{\mathcal{C}})\bar{q}_1(\bar{q}_2, \bar{q}_1)$ is an element of $\Theta_{\bar{q}_1,\bar{q}_2}$ such that $\epsilon(C_\alpha) = 1$, and, by definition of R, $\bar{q}_1 \; R \; \delta(\bar{q}_2)$, as needed.

For the step condition, assume that $q_1 \; R \; \mu_2$ and let $q_1 \xrightarrow{a}_1 \mu'_1$ be a transition of $\mathcal{P}_1$, which we denote by tr. We must exhibit a probability measure $\xi'_2 \in Disc(Disc(Q_2))$ and a hyper-transition $\mu_2 \xrightarrow{a}_2 \mu''_2$, matching the given transition, where $\mu''_2 = \mathit{flatten}(\xi'_2)$ and $\mu'_1 \; R \; \xi'_2$. We do this by deriving a transition tr_α of $\mathcal{P}_2$ for each execution α of Θ_{q_1} and by combining the tr_α's appropriately into transitions tr_q, for each state $q \in supp(\mu_2)$, that are the basis for the required hyper-transition. The tr_α transitions are derived from η; the construction considers only those α's for which $\epsilon(C_\alpha) > 0$. The other α's can be treated arbitrarily.

Consider an execution α of Θ_{q_1} such that $\epsilon(C_\alpha) > 0$. By property (13), $\alpha \in \Theta_{q_1,q_2}$ for some state q_2 in $supp(\mu_2)$. Since Θ_{q_1,q_2} is a subset of Θ_{q_1}, by definition of Θ_{q_1}, $trace(\alpha) = \beta q_1$ for some finite trace β. Therefore, $\eta(C_{\beta q_1}) > 0$. Since q_1 enables at least one transition in $\mathcal{P}_1$, specifically transition tr, (6) from Proposition 4.6 implies that $\eta(C_{\beta q_1\, ch}) = \eta(C_{\beta q_1})$. Then, since $\mathcal{P}_2$ and $\mathcal{C}$ have no internal actions, σ_2 schedules action *ch* from α with probability 1.

By definition of $\mathcal{C} = \mathit{tester}(\mathcal{P}_1)$, the transition labeled by *ch* leaving from state q_1 of $\mathcal{C}$ leads to state *tr* with probability > 0. Hence, $\epsilon(C_{\alpha\, ch\,(q_2,tr)}) > 0$. By (7) from Proposition 4.6, where only the first term of the right-hand side is nonzero due to the absence of internal actions, $\eta(C_{\beta q_1\, ch}) = \sum_{(a,q')|a\in E, q_1 \xrightarrow{a} q'} \eta(C_{\beta q_1\, ch\, aq'})$. Hence, σ_2 must extend $\alpha\, ch\,(q_2, tr)$ with two steps labeled by an action and a state of $\mathcal{P}_1$, respectively, where the action and the state are compatible with one of the transitions of $\mathcal{P}_1$ that are enabled from q_1. Since state *tr* of $\mathcal{C}$ enables only actions in $\mathit{supp}(\mu_1')$, and since, by the tree-structure of $\mathcal{P}_1$, a is uniquely determined by μ_1', the action that is scheduled is a, and the state that is scheduled is a state in $\mathit{supp}(\mu_1')$. Thus, $\sigma_2(\alpha\, ch\,(q_2, tr))$ returns a probability measure over transitions labeled by a. This measure identifies a combined transition of $\mathcal{P}_2$ labeled by a that leaves from q_2, which we denote by tr_α.

Now, using the tr_α transitions, we define a combined transition from each state in the support of μ_2. Namely, for each state $q \in \mathit{supp}(\mu_2)$, let tr_q be the combined transition of $\mathcal{P}_2$ defined by

$$tr_q \triangleq \sum_{\alpha\in\Theta_{q_1,q}} \frac{\epsilon(C_\alpha)}{\sum_{\alpha'\in\Theta_{q_1,q}} \epsilon(C_{\alpha'})} tr_\alpha. \tag{16}$$

Informally, each element of $\Theta_{q_1,q}$ is an execution that contributes to the emulation of transition $q_1 \xrightarrow{a}_1 \mu_1'$ from q. Equation (16) computes tr_q, the overall contribution to the emulation from q, by averaging over all elements of $\Theta_{q_1,q}$. We could prove that $\Theta_{q_1,q}$ contains only one element α' such that $\epsilon(C_{\alpha'}) > 0$ and simplify (16) accordingly. However, this simplification is not necessary for the proof. Now we define the measure $\mu_2'' \in \mathit{Disc}(Q_2)$:

$$\mu_2'' \triangleq \sum_{q\in \mathit{supp}(\mu_2)} \mu_2(q)\mu_{tr_q}. \tag{17}$$

Then, by construction, $\mu_2 \xrightarrow{a} \mu_2''$ is a hyper-transition of $\mathcal{P}_2$.

It remains to define a probability measure $\xi_2' \in \mathit{Disc}(\mathit{Disc}(Q_2))$ such that $\mu_2'' = \mathit{flatten}(\xi_2')$ and $\mu_1'\ R\ \xi_2'$.

For each $q \in \mathit{supp}(\mu_1')$, let μ_q be the unique measure such that $q\ R\ \mu_q$. We can identify μ_q because R is a function. Define $\xi_2' \in \mathit{Disc}(\mathit{Disc}(Q_2))$ such that, for each $q \in \mathit{supp}(\mu_1')$, $\xi_2'(\mu_q) = \sum_{q'\in \mathit{supp}(\mu_1')|\mu_{q'}=\mu_q} \mu_1'(q')$. Then $\mu_1'\ R\ \xi_2'$ by the definition of ξ_2'.

It remains to show that $\mu_2'' = \mathit{flatten}(\xi_2')$, that is, that $\mu_2'' = \sum_{\rho\in \mathit{supp}(\xi_2')} \xi_2'(\rho)\rho$. From the definition of ξ_2' and of the flatten operator, it suffices to show that for every $q_2 \in Q_2$,

$$\mu_2''(q_2) = \sum_{q\in \mathit{supp}(\mu_1')} \mu_1'(q)\mu_q(q_2). \tag{18}$$

To prove (18) we first claim that the following equation is valid for each pair of states q_1, q_2 of $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively, if k denotes the number of transitions of $\mathcal{P}_1$ that are enabled from q_1:

$$\sum_{\alpha\in\Theta_{q_1}} \epsilon(C_\alpha)\mu_{tr_\alpha}(q_2) = k \sum_{q\in \mathit{supp}(\mu_1'),\, \alpha\in\Theta_{q,q_2}} \epsilon(C_\alpha). \tag{19}$$

Informally, the left-hand side of (19) represents the probability of scheduling q_1 and then reaching q_2 according to the transition tr_α, without considering the outcome of the transition labeled by *ch*. The right-hand side, on the other hand, computes the probability of scheduling q_1, scheduling *ch* and reaching μ_1', and then scheduling tr_α and reaching q_2. State μ_1' is reached by *ch* with probability $1/k$, which justifies the k factor in the right-hand side.

To prove (19), consider an execution $\alpha \in \Theta_{q,q_2}$, where $q \in \mathit{supp}(\mu_1')$. Since q always occurs after q_1, execution α can be split into $\alpha' \frown \alpha''$, where $\alpha' \in \Theta_{q_1}$. Furthermore, $\mathit{trace}(\alpha'') = \mathit{ch}\, aq$, and since there are no internal actions in $\mathcal{P}_2$ and $\mathcal{C}$, α is the unique extension of α' that is in Θ_{q,q_2}. In particular,

$$\alpha'' = (q', q_1)\; \mathit{ch}\; (q', tr)a(q_2, tr)q(q_2, q)$$

for some state q' of $\mathcal{P}_2$, and $\epsilon(C_\alpha) = \epsilon(C_{\alpha'})(1/k)\mu_{tr_{\alpha'}}(q_2)$. Thus, each summand on the right-hand side of (19) has a corresponding summand on the left-hand side that differs by a factor of k, and the correspondence relation is an injection. If the correspondence is not a bijection, then the α terms that are left out on the left-hand side are such that $\mu_{tr_\alpha}(q_2) = 0$ (otherwise an extension in Θ_{q,q_2} for some q exists). This suffices.

We now consider the left-hand side of (18). Consider the definition of μ_2'' given by (17). By expanding $\mu_2(q)$ according to the definition of μ_2 given by (12), and expanding $\mu_{tr}(q_2)$ according to the definition of μ_{tr} given by (16), we obtain

$$\mu_2''(q_2) = \sum_{q \in \mathit{supp}(\mu_2)} \frac{\sum_{\alpha \in \Theta_{q_1,q}} \epsilon(C_\alpha)}{\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha)} \frac{\sum_{\alpha \in \Theta_{q_1,q}} \epsilon(C_\alpha)\mu_{tr_\alpha}(q_2)}{\sum_{\alpha \in \Theta_{q_1,q}} \epsilon(C_\alpha)}.$$

By cross-simplifying the top-leftmost and bottom-rightmost factors, and by factoring the left denominator out of the sum, we obtain

$$\mu_2''(q_2) = \frac{\sum_{q \in \mathit{supp}(\mu_2)} \sum_{\alpha \in \Theta_{q_1,q}} \epsilon(C_\alpha)\mu_{tr_\alpha}(q_2)}{\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha)}.$$

By property (13), we can rewrite the numerator as follows:

$$\mu_2''(q_2) = \frac{\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha)\mu_{tr_\alpha}(q_2)}{\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha)}.$$

By multiplying numerator and denominator by k, applying (19) to the numerator, and applying (15) to the denominator, we obtain

$$\mu_2''(q_2) = \frac{\sum_{q \in \mathit{supp}(\mu_1')\,,\, \alpha \in \Theta_{q,q_2}} \epsilon(C_\alpha)}{\sum_{q \in \mathit{supp}(\mu_1')\,,\, \alpha \in \Theta_q} \epsilon(C_\alpha)}. \tag{20}$$

We now consider the right-hand side of (18). By applying (14) and (12) to the two factors of the right-hand side of (18), and by simplifying common factors algebraically, we obtain

$$\sum_{q \in \mathit{supp}(\mu_1')} \mu_1'(q)\mu_q(q_2) = \frac{\sum_{q \in \mathit{supp}(\mu_1')\,,\, \alpha \in \Theta_{q,q_2}} \epsilon(C_\alpha)}{\sum_{q \in \mathit{supp}(\mu_1')\,,\, \alpha \in \Theta_q} \epsilon(C_\alpha)}. \tag{21}$$

Now (18) follows by direct combination of (20) and (21). ◻

Interestingly, the probabilistic forward simulation that we constructed in the above proof is functional. Functional simulations are usually called *refinement mappings* [21, 26]. Write $\mathcal{P}_1 \leq_{PR} \mathcal{P}_2$ if there exists a functional probabilistic forward simulation from $\mathcal{P}_1$ to $\mathcal{P}_2$. Then we can state the following new proposition, which is a probabilistic version of Proposition 3.12 in [26].

PROPOSITION 6.2. *Let $\mathcal{P}_1$, $\mathcal{P}_2$ be PAs without internal actions such that $\mathcal{P}_1$ is tree-structured. Then $\mathcal{P}_1 \leq_{PF} \mathcal{P}_2$ iff $\mathcal{P}_1 \leq_{PR} \mathcal{P}_2$.*

Proof. It is enough to observe that each state q_1 of $\mathcal{P}_1$ occurs with some positive probability in the trace distribution η of the proof of Proposition 6.1. ◻

As usual, we may eliminate the assumption that $\mathcal{P}$ is tree-structured.

THEOREM 6.3. *Let $\mathcal{P}_1$, $\mathcal{P}_2$ be PAs without internal actions. Then $\mathcal{P}_1 \leq_{DC} \mathcal{P}_2$ iff $\mathcal{P}_1 \leq_{PF} \mathcal{P}_2$.*

Proof. First we prove the soundness of probabilistic forward simulations:

$$\begin{aligned} \mathcal{P}_1 \leq_{PF} \mathcal{P}_2 \quad &\Rightarrow \quad \mathcal{P}_1 \leq_{wPF} \mathcal{P}_2 \quad \text{(Proposition 3.10.1)} \\ &\Rightarrow \quad \mathcal{P}_1 \leq_{DC} \mathcal{P}_2 \quad \text{(Proposition 3.10.3).} \end{aligned}$$

Now we prove completeness:

$$\begin{aligned} \mathcal{P}_1 \leq_{DC} \mathcal{P}_2 \quad &\Rightarrow \quad \mathit{Unfold}(\mathcal{P}_1) \leq_{DC} \mathcal{P}_1 \leq_{DC} \mathcal{P}_2 \quad \text{(Proposition 3.12)} \\ &\Rightarrow \quad \mathit{Unfold}(\mathcal{P}_1) \leq_{DC} \mathcal{P}_2 \quad (\leq_{DC} \text{ is transitive}) \\ &\Rightarrow \quad \mathit{Unfold}(\mathcal{P}_1) \leq_{PF} \mathcal{P}_2 \quad \text{(Proposition 6.1)} \\ &\Rightarrow \quad \mathcal{P}_1 \leq_{PF} \mathit{Unfold}(\mathcal{P}_1) \leq_{PF} \mathcal{P}_2 \quad \text{(Proposition 3.11)} \\ &\Rightarrow \quad \mathcal{P}_1 \leq_{PF} \mathcal{P}_2 \quad (\leq_{PF} \text{ is transitive}). \quad ◻ \end{aligned}$$

6.2. PAs with internal actions. Again, we start with tree-structured PAs.

PROPOSITION 6.4. *Let $\mathcal{P}_1$, $\mathcal{P}_2$ be PAs with $\mathcal{P}_1$ tree-structured. Then $\mathcal{P}_1 \leq_{DC} \mathcal{P}_2$ implies $\mathcal{P}_1 \leq_{wPF} \mathcal{P}_2$.*

Proof. Assume that $\mathcal{P}_1 \leq_{DC} \mathcal{P}_2$. Define the tester PA $\mathcal{C}$ of $\mathcal{P}_1$, the observer σ_1, the trace distribution η, the scheduler σ_2, the probabilistic execution ϵ, and the Θ sets as in the proof of Proposition 5.1. Define relation R according to (12) as in the proof of Proposition 6.1. Observe that formulas (13), (14), and (15) hold for the same reasons as before. Define a new relation R' as follows: $q_1 \; R' \; \mu_2$ iff there exists a measure μ_2' such that $\mu_1 \implies \mu_2'$ and $q_1 \; R \; \mu_2'$. Observe that trivially $R {\subseteq} R'$. We show that R' is a weak probabilistic forward simulation from $\mathcal{P}_1$ to $\mathcal{P}_2$.

For the start condition, we must show that $\bar{q}_1 \; R' \; \delta(\bar{q}_2)$. By Item 1 of Proposition 4.6, $\eta(C_{\bar{q}_1}) = 1$. This means that $\sum_{\alpha \in \Theta_{\bar{q}_1} | \mathit{trace}(\alpha) = \bar{q}_1} \epsilon(C_\alpha) = 1$. Let $\Theta'_{\bar{q}_1}$ be the set of elements of $\Theta_{\bar{q}_1}$ with trace $\bar{q}_1$, and let ϵ' be the truncation of ϵ to $\Theta'_{\bar{q}_1}$. Then ϵ' assigns probability 1 to the set of finite execution fragments with trace $\bar{q}_1$. Furthermore, observing that all elements of $\Theta'_{\bar{q}_1}$ are not prefixes of each other, we derive $\epsilon'(C_\alpha) = \epsilon'(\{\alpha\})$ for each $\alpha \in \Theta'_{\bar{q}_1}$. Finally, observing that each element of $\Theta_{\bar{q}_1} - \Theta'_{\bar{q}_1}$ is not a prefix of any element of $\Theta'_{\bar{q}_1}$, we derive $\epsilon'(C_\alpha) = \epsilon'(\{\alpha\}) = 0$ for each $\alpha \in \Theta_{\bar{q}_1} - \Theta'_{\bar{q}_1}$. Let μ_2' be $\mathit{lstate}(\epsilon')$. By the definition of a weak hypertransition, $\delta(\bar{q}_2) \implies \mu_2'$. We show that $\bar{q}_1 \; R \; \mu_2'$, which suffices. Consider a state q_2 of $\mathcal{P}_2$. By definition of μ_2', definition of $\Theta_{\bar{q}_1, q_2}$, and the fact that $\mathit{supp}(\epsilon') \subseteq \Theta_{\bar{q}_1}$, $\mu_2'(q_2) = \sum_{\alpha \in \Theta_{\bar{q}_1, q_2}} \epsilon'(\{\alpha\})$. Then the result follows immediately by observing that this equation corresponds to (12) since $\epsilon'(\{\alpha\}) = \epsilon'(C_\alpha)$ when $\alpha \in \Theta_{\bar{q}_1}$ and since $\sum_{\alpha \in \Theta_{\bar{q}_1}} \epsilon(C_\alpha) = 1$.

For the step condition, assume that $q_1 \; R' \; \mu_2$, and let $q_1 \xrightarrow{a}_1 \mu_1'$ be a transition of $\mathcal{P}_1$, which we denote by tr. By definition of R', there exists a measure μ_2' such that $\mu_2 \Longrightarrow \mu_2'$ and $q_1\ R\ \mu_2'$. We now show that there exists a measure ξ_2' and a measure $\mu_2'' = \mathit{flatten}(\xi_2')$ such that $\mu_1\ R\ \xi_2'$ and $\mu_2' \overset{a}{\Longrightarrow} \mu_2''$. Then $\mu_1\ R'\ \xi_2'$, and by Proposition 3.6, $\mu_2 \overset{a}{\Longrightarrow} \mu_2''$.

The proof of existence of ξ_2' and μ_2'' proceeds exactly as in the case of Proposition 6.1 except for the definition of the tr_α transitions. Thus, in the rest of the proof we construct the tr_α's and prove that (19) still holds.

We introduce a special *conditional* construction that is needed for the definition of the tr_α's. Let $\mathcal{C}_{tr}$ be the same as $\mathcal{C}$ except that the transition $q_1 \overset{ch}{\rightarrow} \mu$, where μ is uniquely determined by q_1, is replaced by $q_1 \overset{ch}{\rightarrow} \delta(tr)$. Given a scheduler σ for $\mathcal{P}_2\|\mathcal{C}$, define the scheduler $\sigma \mid tr$ for $\mathcal{P}_2\|\mathcal{C}_{tr}$ that is the same as σ except that transition $q_1 \overset{ch}{\rightarrow} \delta(tr)$ of $\mathcal{C}_{tr}$ is chosen whenever σ chooses $q_1 \overset{ch}{\rightarrow} \mu$. Given a probabilistic execution fragment ϵ' of $\mathcal{P}_2\|\mathcal{C}$, generated by some scheduler σ, define $\epsilon' \mid tr$ to be the result of $\sigma \mid tr$ applied to $\mathcal{P}\|\mathcal{C}_{tr}$ from the start state of ϵ'. The intuition behind $\epsilon' \mid tr$ is that we study ϵ' under the condition that tr is the outcoming state of $\mathcal{C}$ whenever $q_1 \overset{ch}{\rightarrow} \mu$ is scheduled. Then, the following two properties are valid:

1. $(\epsilon' \mid tr)\lceil\mathcal{P}_2$ is a probabilistic execution fragment of $\mathcal{P}_2$.
2. For each finite execution fragment α of $\mathcal{P}_2\|\mathcal{C}$ where state tr occurs and such that $\mathit{fstate}(\alpha)$ is not of the form $(\cdot, tr)$, $(\epsilon' \mid tr)(C_\alpha) = k\epsilon(C_\alpha)$, where k is the size of $\mathit{supp}(\mu)$.

The first item follows immediately from Proposition 3.7, given that $\epsilon' \mid tr$ is a probabilistic execution fragment of $\mathcal{P}_2\|\mathcal{C}_{tr}$. The second item follows directly from the definition of probability of a cone, since in ϵ' the probability associated with the edge $q\ ch\ (\cdot, tr)$ is $1/k$ while in $\epsilon' \mid tr$ the probability of the same edge is 1.

We now define the tr_α's. Consider an execution α of Θ_{q_1} such that $\epsilon(C_\alpha) > 0$. Let ϵ^1 be the truncation of ϵ at all the points in $\cup_{q\in \mathit{supp}(\mu_1')}\Theta_q$, which is a probabilistic execution of $\mathcal{P}_2\|\mathcal{C}$ by definition. Let ϵ^1_α be $\epsilon^1 \triangleright \alpha$, which is a probabilistic execution fragment of $\mathcal{P}_2\|\mathcal{C}$ by definition. Finally, let ϵ^2_α be $(\epsilon^1_\alpha \mid tr)\lceil\mathcal{P}_2$, which is a probabilistic execution fragment of $\mathcal{P}_2$ by Property 1.

By definition of Θ_{q_1}, $\mathit{trace}(\alpha) = \beta q_1$ for some finite trace β. Therefore, $\eta(C_{\beta q_1}) > 0$. Since q_1 enables at least one transition in $\mathcal{P}_1$, specifically transition tr, (6) from Proposition 4.6 implies that $\eta(C_{\beta q_1\, ch}) = \eta(C_{\beta q_1})$. Thus, action ch occurs as the first external action with probability 1 in μ^1_α.

By (7) from Proposition 4.6, if the occurrence of action ch leads $\mathcal{C}$ to state tr, then an action in $\mathit{supp}(\mu_1')$ occurs eventually in ϵ with probability 1, leading $\mathcal{C}$ to a state in $\mathit{supp}(\mu_1')$, which is a truncation point according to the definition of ϵ^1. Thus, the probability of termination in $\epsilon^1_\alpha \mid tr$ is 1, as well as the probability of termination in ϵ^2_α; that is, ϵ^2_α assigns probability 1 to the set of finite executions. Furthermore, given that action a is uniquely determined by μ_1' ($\mathcal{P}_1$ is tree-structured), again by (7) from Proposition 4.6 all finite executions α' with $\epsilon^2_\alpha(\alpha') > 0$ have trace $\mathit{trace}(a)$. Thus, ϵ^2_α is a representation of a weak combined transition labeled by a from $\mathit{lstate}(\alpha)\lceil\mathcal{P}_2$. Denote such transition by tr_α.

We are left to show that (19) still holds. That is,

$$\sum_{\alpha\in\Theta_{q_1}} \epsilon(C_\alpha)\mu_{tr_\alpha}(q_2) = k \sum_{q\in \mathit{supp}(\mu_1')\,,\, \alpha\in\Theta_{q,q_2}} \epsilon(C_\alpha).$$

We consider first the term $\mu_{tr_\alpha}(q_2)$. From the definition of tr_α and of weak combined transition we get

$$\mu_{tr_\alpha}(q_2) = \sum_{\alpha' \mid lstate(\alpha') = q_2} \epsilon^2_\alpha(\alpha').$$

By applying the definition of projection, and using the fact that $\epsilon^1_\alpha \mid tr$ assigns probability 1 to the set of finite executions, we get

$$\mu_{tr_\alpha}(q_2) = \sum_{\alpha' \mid lstate(\alpha' \lceil \mathcal{P}_2) = q_2} (\epsilon^1_\alpha \mid tr)(\alpha').$$

Given that the truncation points of ϵ^1 are all at the $\cup_{q \in supp(\mu'_1)} \Theta_q$ points, the only finite executions α' that have nonzero probability are such that $\alpha \frown \alpha'$ is in some set Θ_q. Furthermore, given that no execution in $\cup_{q \in supp(\mu'_1)} \Theta_q$ is a prefix of another (our PAs are tree-structured and all actions in $supp(\mu'_1)$ occur in different branches), the probabilities of the finite executions can be replaced by the probabilities of their cones, thus getting

$$\mu_{tr_\alpha}(q_2) = \sum_{q \in supp(\mu'_1)} \sum_{\alpha' \mid \alpha \frown \alpha' \in \Theta_{q,q_2}} (\epsilon^1_\alpha \mid tr)(C_{\alpha'}).$$

By property 2 we can get rid of the conditional on tr by introducing a k factor, thus getting

$$\mu_{tr_\alpha}(q_2) = \sum_{q \in supp(\mu'_1)} \sum_{\alpha' \mid \alpha \frown \alpha' \in \Theta_{q,q_2}} k \epsilon^1_\alpha(C_{\alpha'}). \tag{22}$$

By replacing $\mu_{tr_\alpha}(q_2)$ according to (22) in the left-hand side of (19) and by rearranging terms algebraically, we obtain

$$\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha) \mu_{tr_\alpha}(q_2) = k \sum_{q \in supp(\mu'_1)} \sum_{\alpha \in \Theta_{q_1}} \sum_{\alpha' \mid \alpha \frown \alpha' \in \Theta_{q,q_2}} \epsilon(C_\alpha) \epsilon^1_\alpha(C_{\alpha'}).$$

By using that $\epsilon^1_\alpha = \epsilon^1 \triangleright \alpha$ (by definition) and Proposition 3.15, the two probabilities in the equation above can be grouped into $\epsilon(C_{\alpha \frown \alpha'})$. By observing that all elements in Θ_{q,q_2}, with $q \in supp(\mu'_1)$, have a prefix in Θ_{q_1}, the intermediate sum can be removed, thus getting

$$\sum_{\alpha \in \Theta_{q_1}} \epsilon(C_\alpha) \mu_{tr_\alpha}(q_2) = k \sum_{q \in supp(\mu'_1)} \sum_{\alpha \in \Theta_{q,q_2}} \epsilon(C_\alpha),$$

which is (19), as needed. □

THEOREM 6.5. *Let $\mathcal{P}_1$, $\mathcal{P}_2$ be PAs. Then $\mathcal{P}_1 \leq_{DC} \mathcal{P}_2$ iff $\mathcal{P}_1 \leq_{wPF} \mathcal{P}_2$.*

Proof. Soundness of weak probabilistic forward simulations follows immediately from Proposition 3.10. Completeness is established as follows:

$$\begin{aligned}
\mathcal{P}_1 \leq_{DC} \mathcal{P}_2 \quad &\Rightarrow \quad \mathit{Unfold}(\mathcal{P}_1) \leq_{DC} \mathcal{P}_1 \leq_{DC} \mathcal{P}_2 \quad \text{(Proposition 3.12)} \\
&\Rightarrow \quad \mathit{Unfold}(\mathcal{P}_1) \leq_{DC} \mathcal{P}_2 \quad (\leq_{DC} \text{ is transitive}) \\
&\Rightarrow \quad \mathit{Unfold}(\mathcal{P}_1) \leq_{wPF} \mathcal{P}_2 \quad \text{(Proposition 6.4)} \\
&\Rightarrow \quad \mathcal{P}_1 \leq_{PF} \mathit{Unfold}(\mathcal{P}_1) \leq_{wPF} \mathcal{P}_2 \quad \text{(Proposition 3.11)} \\
&\Rightarrow \quad \mathcal{P}_1 \leq_{wPF} \mathit{Unfold}(\mathcal{P}_1) \leq_{wPF} \mathcal{P}_2 \quad \text{(Proposition 3.10)} \\
&\Rightarrow \quad \mathcal{P}_1 \leq_{wPF} \mathcal{P}_2 \quad (\leq_{wPF} \text{ is transitive}). \quad \square
\end{aligned}$$

7. Concluding remarks. We have characterized the trace distribution precongruence for nondeterministic and probabilistic automata, with and without internal actions, in terms of four kinds of simulation relations, $\leq_F$, $\leq_{wF}$, $\leq_{PF}$, and $\leq_{wPF}$. In particular, this shows that probabilistic contexts are capable of observing all the distinctions that can be expressed using these simulation relations. Our main technical contribution is the definition of special contexts, called testers, that, under the action of an appropriate scheduler, can reveal the branching structure of a PA via a trace distribution. Some technical improvements are possible. For example, our finite branching restriction can be relaxed to countable branching, simply by replacing uniform distributions in the tester automata by other distributions such as exponential distributions. In that case, however, calculations become more complicated. We have also considered nondeterministic and probabilistic automata with countably many states and actions. Again, this restriction can be relaxed at the cost of complicating the definition of the σ-field of execution fragments: the generators would be arbitrary unions of cones, and the measure of a union of cones would be just the sum of the measures of each single cone. Indeed, discrete transitions and discrete schedulers ensure that there are at most countably many cones with nonzero measure.

Although in this paper we reach a point where we have a full understanding of trace distribution precongruence as a branching relation, a natural question is whether it is possible to define linear probabilistic extensions of language inclusion. A potential approach is to consider ordinary traces paired with their maximal or minimal probabilities under all schedulers, but the induced preorder relations do not appear to be interesting. Another approach, followed in [19], is to extend classical testing preorders by considering the maximal and minimal probabilities of success of a test; however, even in such case the resulting precongruence is characterized in terms of simulation relations that, although weaker than the relations studied in this paper, are still branching relations. Other approaches ensure compositionality of trace distribution inclusion by restricting parallel composition so that the nondeterminism of each component is resolved based only on externally visible behavior of the other components. This approach is investigated in [10] in a synchronous model. In [8, 7], an asynchronous switched probabilistic input/output automaton model (PIOA) is presented, which uses a token structure to eliminate global nondeterministic choices. This token structure ensures that, at any point in time, there is at most one active component in a system and this unique component determines the next active component. Thus, global scheduling is performed jointly by all local schedulers, which have access to local information only. A notion of switched probabilistic systems is defined, which are switched PIOAs paired with sets of *acceptable* I/O schedulers. A trace-style semantics for switched probabilistic systems is given, using the notion of likelihood assignments. This semantics is shown to be compositional with respect to a parallel operator that combines local I/O schedulers into a joint I/O scheduler. Thus, the approach of [8, 7] can be characterized as *schedule-and-compose*, where local nondeterministic choices are resolved before the components are placed in parallel. In [7] also a similar strategy is pursued, but without the token structure. Instead, several axioms are imposed on the reactive and generative transition structures, so that branching occurs only when it is meant to be globally visible (i.e., the branches carry different visible action labels). These axioms capture a local-oblivious assumption on adversaries, which is well known in the area of randomized consensus [9, 3]. The model is proven to be compositional with respect to a schedule-and-compose operator similar to that in [8].

Acknowledgment. We are grateful to Erik de Vink for numerous detailed observations and suggestions for improvements.

REFERENCES

[1] S. Aggarwal, *Time Optimal Self-Stabilizing Spanning Tree Algorithms*, Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1994; also available as Technical Report MIT/LCS/TR-632.

[2] S. Andova and T. Willemse, *Branching bisimulation for probabilistic systems: Characteristics and decidability*, Theoret. Comput. Sci., 356 (2006), pp. 325–355.

[3] Y. Aumann and M. A. Bender, *Efficient low-contention asynchronous consensus with the value-oblivious adversary scheduler*, Distrib. Comput., 17 (2005), pp. 191–207.

[4] F. Bartels, A. Sokolova, and E. de Vink, *A hierarchy of probabilistic system types*, Theoret. Comput. Sci., 327 (2004), pp. 3–22.

[5] J. A. Bergstra, J. W. Klop, and E.-R. Olderog, *Readies and failures in the algebra of communicating processes*, SIAM J. Comput., 17 (1988), pp. 1134–1177.

[6] L. Cheung, *Randomized wait-free consensus using an atomicity assumption*, in Principles of Distributed Systems (Proceedings of OPODIS 2005, Pisa, Italy), J. H. Anderson, G. Prencipe, and R. Wattenhofer, eds., Lecture Notes in Comput. Sci. 3974, Springer, New York, 2006, pp. 47–60.

[7] L. Cheung, *Reconciling Nondeterministic and Probabilistic Choices*, Ph.D. thesis, Faculty of Science, Radboud University, Nijmegen, The Netherlands, 2006.

[8] L. Cheung, N. A. Lynch, R. Segala, and F. W. Vaandrager, *Switched PIOA: Parallel composition via distributed scheduling*, Theoret. Comput. Sci., 365 (2006), pp. 83–108.

[9] B. Chor, A. Israeli, and M. Li, *Wait-free consensus using asynchronous hardware*, SIAM J. Comput., 23 (1994), pp. 701–712.

[10] L. de Alfaro, T. A. Henzinger, and R. Jhala, *Compositional methods for probabilistic systems*, in Proceedings of CONCUR 01, Aalborg, Denmark, 2001, K. G. Larsen and M. Nielsen, eds., Lecture Notes in Comput. Sci. 2154, Springer, New York, 2001, pp. 351–365.

[11] W. Feller, *An Introduction to Probability Theory and Its Applications. Volume* 1, John Wiley & Sons, New York, 1950.

[12] R. J. van Glabbeek, *The linear time-branching time spectrum* I. *The semantics of concrete, sequential processes*, in Handbook of Process Algebra, J. A. Bergstra, A. Ponse, and S. A. Smolka, eds., North–Holland, Amsterdam, 2001, pp. 3–99.

[13] R. J. van Glabbeek, S. A. Smolka, and B. Steffen, *Reactive, generative, and stratified models of probabilistic processes*, Inform. and Control, 121 (1995), pp. 59–80.

[14] J. F. Groote and F. W. Vaandrager, *Structured operational semantics and bisimulation as a congruence*, Inform. and Comput., 100 (1992), pp. 202–260.

[15] H. A. Hansson, *Time and Probability in Formal Design of Distributed Systems*, Real-Time Safety Critical Systems 1, Elsevier, New York, 1994.

[16] A. Hinton, M. Z. Kwiatkowska, G. Norman, and D. Parker, *Prism: A tool for automatic verification of probabilistic systems*, in TACAS, H. Hermanns and J. Palsberg, eds., Lecture Notes in Comput. Sci. 3920, Springer-Verlag, New York, 2006, pp. 441–444.

[17] C. A. R. Hoare, *Communicating Sequential Processes*, Prentice–Hall, Englewood Cliffs, NJ, 1985.

[18] B. Jonsson and K. G. Larsen, *Specification and refinement of probabilistic processes*, in Proceedings of the 6th Annual Symposium on Logic in Computer Science, Amsterdam, IEEE Press, Piscataway, NJ, 1991, pp. 266–277.

[19] B. Jonsson and W. Yi, *Testing preorders for probabilistic processes can be characterized by simulations*, Theoret. Comput. Sci., 282 (2002), pp. 33–51.

[20] M. Z. Kwiatkowska and G. Norman, *Verifying randomized byzantine agreement*, in FORTE, D. Peled and M. Y. Vardi, eds., Lecture Notes in Comput. Sci. 2529, Springer-Verlag, New York, 2002, pp. 194–209.

[21] L. Lamport, *Specifying concurrent program modules*, ACM Trans. Prog. Lang. Syst., 5 (1983), pp. 190–222.

[22] K. G. Larsen and A. Skou, *Bisimulation through probabilistic testing*, Inform. and Comput., 94 (1991), pp. 1–28.

[23] N. A. Lynch, I. Saias, and R. Segala, *Proving time bounds for randomized distributed algorithms*, in Proceedings of the 13th Annual ACM Symposium on the Principles of Distributed Computing, Los Angeles, CA, 1994, ACM, New York, 1994, pp. 314–323.

[24] N. A. Lynch, R. Segala, and F. W. Vaandrager, *Compositionality for probabilistic automata*, in Proceedings of the 14th International Conference on Concurrency Theory (CONCUR 2003), Marseille, France, R. Amadio and D. Lugiez, eds., Lecture Notes in Comput. Sci. 2761, Springer-Verlag, New York, 2003, pp. 208–221.

[25] N. A. Lynch and M. R. Tuttle, *An introduction to input/output automata*, CWI Quarterly, 2 (1989), pp. 219–246.

[26] N. A. Lynch and F. W. Vaandrager, *Forward and backward simulations,* I*: Untimed systems*, Inform. and Comput., 121 (1995), pp. 214–233.

[27] R. Milner, *Communication and Concurrency*, Prentice–Hall, Englewood Cliffs, NJ, 1989.

[28] G. Norman, *Analysing randomized distributed algorithms*, in Validation of Stochastic Systems, C. Baier, B. R. Haverkort, H. Hermanns, J.-P. Katoen, and M. Siegle, eds., Lecture Notes in Comput. Sci. 2925, Springer-Verlag, New York, 2004, pp. 384–418.

[29] A. Philippou, I. Lee, and O. Sokolsky, *Weak bisimulation for probabilistic systems*, in Proceedings of CONCUR 2000, University Park, PA, C. Palamidessi, ed., Lecture Notes in Comput. Sci. 1877, Springer-Verlag, New York, 2000, pp. 334–349.

[30] A. Pogosyants, R. Segala, and N. A. Lynch, *Verification of the randomized consensus algorithm of Aspnes and Herlihy: A case study*, Distributed Computing, 13 (2000), pp. 155–186.

[31] R. Segala, *Compositional trace-based semantics for probabilistic automata*, in Proceedings of CONCUR 95, Philadelphia, PA, I. Lee and S. A. Smolka, eds., Lecture Notes in Comput. Sci. 962, Springer-Verlag, New York, 1995, pp. 234–248.

[32] R. Segala, *Modeling and Verification of Randomized Distributed Real-Time Systems*, Ph.D. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1995; also available as Technical Report MIT/LCS/TR-676.

[33] R. Segala, *Testing probabilistic automata*, in Proceedings of CONCUR 96, Pisa, Italy, 1996, U. Montanari and V. Sassone, eds., Lecture Notes in Comput. Sci. 1119, Springer, New York, 1996, pp. 299–314.

[34] R. Segala and N. A. Lynch, *Probabilistic simulations for probabilistic processes*, Nordic J. Comput., 2 (1995), pp. 250–273.

[35] R. Segala and A. Turrini, *Comparative analysis of bisimulation relations on alternating and nonalternating probabilistic models*, in Proceedings of the Second International Conference on the Quantitative Evaluation of Systems (QEST 2005), Torino, Italy, IEEE Computer Society, Piscataway, NJ, 2005, pp. 44–53.

[36] A. Sokolova and E. P. de Vink, *Probabilistic automata: System types, parallel composition, and comparison*, in Validation of Stochastic Systems—A Guide to Current Research, C. Baier, B. R. Haverkort, H. Hermanns, J.-P. Katoen, and M. Siegle, eds., Lecture Notes in Comput. Sci. 2925, Springer-Verlag, New York, 2004, pp. 1–43.

[37] M. I. A. Stoelinga, *Alea Jacta Est: Verification of Probabilistic, Real-Time, and Parametric Systems*, Ph.D. thesis, Faculty of Science, University of Nijmegen, Nijmegen, The Netherlands, 2002.

[38] M. I. A. Stoelinga, *An introduction to probabilistic automata*, Bull. European Assoc. Theoret. Comput. Sci., 78 (2002), pp. 176–198.

[39] M. I. A. Stoelinga and F. W. Vaandrager, *Root contention in IEEE* 1394, in Proceedings of the 5th International AMAST Workshop on Formal Methods for Real-Time and Probabilistic Systems, Bamberg, Germany, J.-P. Katoen, ed., Lecture Notes in Comput. Sci. 1601, Springer-Verlag, New York, 1999, pp. 53–74.

[40] M. Y. Vardi, *Automatic verification of probabilistic concurrent finite-state programs*, in Proceedings of the 26th IEEE Symposium on Foundations of Computer Science, Portland, OR, 1985, IEEE Press, Piscataway, NJ, pp. 327–338.

SIAM J. Comput.
Vol. 37, No. 4, pp. 1014–1029

GEOMETRIC SEPARATORS AND THEIR APPLICATIONS TO PROTEIN FOLDING IN THE HP-MODEL[*]

BIN FU[†] AND WEI WANG[‡]

Abstract. We develop a new method for deriving a geometric separator for a set of grid points. Our separator has a linear structure, which can effectively partition a grid graph. For example, we prove that for a grid graph G with a set of n points P in a two-dimensional grid, there is a separator with at most $1.129\sqrt{n}$ points in P that partitions G into two disconnected grid graphs each with at most $\frac{2n}{3}$ points. Our separator theorem for grid graphs has a significantly smaller upper bound than that was obtained for the general planar graphs in [H. N. Djidjev and S. M. Venkatesan, *Acta Inform.*, 34 (1997), pp. 231–234]. The protein folding problem in the HP-model is to put a sequence, consisting of two characters H and P, in a d-dimensional grid to have maximal number of HH-contacts, where an HH-contact is a pair of non-consecutive H letters that are put at two grid points of distance 1. Our separator is then applied to develop an exact algorithm for the protein-folding problem in the HP-model, which is NP-hard both in both two and three dimensions [B. Berger and T. Leighton, *J. Comput. Biol.*, 5 (1998), pp. 27–40; P. Crescenzi et al., *J. Comput. Biol.*, 5 (1998), pp. 423–465]. We design a $2^{O(n^{1-\frac{1}{d}}\log n)}$ time algorithm for the d-dimensional protein folding problem in the HP-model. In particular, our algorithm has $O(2^{6.145\sqrt{n}\log n})$ and $O(2^{6.913n^{\frac{2}{3}}\log n})$ computational time in two and three dimensions, respectively.

Key words. separator, protein folding, time complexity, algorithm

AMS subject classification. 68W01

DOI. 10.1137/S0097539704440727

1. Introduction.

Geometric separators are fundamental tools in algorithm design for solving many geometric problems. Lipton and Tarjan [24] showed a well-known geometric separator for planar graphs. Their result has been elaborated on by many authors [9, 15, 2, 10]. The following best known separator theorem for planar graphs was proved by Djidjev and Venkatesan [10].

Theorem 1 (see [10]). *Any planar graph of n vertices has a vertex subset of cardinality* $\leq 1.97\sqrt{n}$ *whose removal separates the graph into two components each having at most* $\frac{2n}{3}$ *vertices.*

Spielman and Teng [38] showed a $\frac{3}{4}$-separator with size $1.82\sqrt{n}$ for planar graphs. Separators for more general graphs were presented in, e.g., [16, 3, 32]. Planar graph separators were applied to derive some $2^{O(\sqrt{n})}$-time algorithms for certain NP-hard problems about planar graphs by Lipton and Tarjan [25] and Ravi and Hunt [35]. Those problems include computing a maximum independent set, a minimum vertex cover, and three-colorings of a planar graph, and the number of satisfying truth assignments to a planar 3CNF formula [23].

[*] Received by the editors February 8, 2004; accepted for publication (in revised form) March 3, 2007; published electronically September 19, 2007. This research was supported by the Louisiana Board of Regents fund under contract LEQSF(2004-07)-RD-A-35. An earlier version of this paper was presented at ICALP 2004 [14].

http://www.siam.org/journals/sicomp/37-4/44072.html

[†] Department of Computer Science, University of Texas-Pan American, Edinburg, TX 78539-2999 (binfu@cs.panam.edu). Part of this research was done while the author was affiliated with the Department of Computer Science, University of New Orleans, New Orleans, LA 70148, and Research Institute for Children, 200 Henry Clay Avenue, New Orleans, LA 70118.

[‡] Department of Chemistry and Biochemistry, University of California at San Diego, San Diego, CA 92093 (wwang@chem.ucsd.edu).

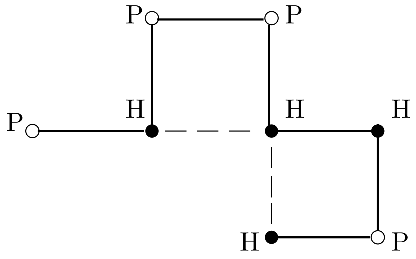


FIG. 1. *The sequence PHPPHHPH is put on the 2D grid. There are two H-H contacts marked by the dotted lines.*

Some other forms of the geometric separators were studied by Miller, Teng, and Vavasis [26], Miller and Thurston [27] and Smith and Wormald [37]. For a set of regular geometric objects such as circles, rectangles, etc., if every point on the plane is covered by at most k objects, the set of the objects is called a k-thick system. Some $O(\sqrt{k \cdot n})$ size separators for k-thick systems and the algorithms for finding them were derived in [27, 26, 37]. Smith and Wormald [37] applied their separators to develop algorithms for some geometric problems such as the planar travelling salesman and the Steiner tree problems (e.g., see [37]). Those problems usually have input points with fixed geometric positions in space.

Finding a minimal size separator, which maintains a similar balance partition condition like one of those mentioned above (e.g., each side of the separator has at most $\frac{2n}{3}$ points), for a grid graph is also an interesting combinatorial problem. Using special geometric properties of grid points, we develop a method for obtaining a separator with a smaller size for grid points via controlling the distances from the grid points to the separator line.

A set of grid points on the plane forms a grid graph by adding edges to every two grid points with distance 1. A grid graph is also a planar graph. In the protein folding of the HP-model, the 20 letter alphabet of amino acids is reduced to a two letter alphabet, namely H and P. H represents hydrophobic amino acids, whereas P represents polar or hydrophilic amino acids. Two monomers form a contact in some specific conformation if they are not consecutive, but occupy neighboring positions in the conformation (i.e., the distance vector between their positions in the conformation is a unit vector). A conformation with minimal energy is just a conformation with the maximal number of contacts between nonconsecutive H-monomers (see Figure 1). The protein folding problem in the HP-model is to find the conformation for any HP-sequence with minimal energy. The protein folding problem in the HP-model is an interesting and challenging problem that deals with a puzzle in the grids. The problem is to put a sequence, consisting of two characters H and P, in a d-dimensional grid to have the the maximal number of HH-contacts. As the input of the protein folding problem is only a sequence of letters, the locations of those letters in space are unknown and will be determined by the algorithm. For this reason, we do not know whether the separator theorems, such as Theorem 1, can be applied to the protein folding problem. We derive a separator theorem for the grid graph with a significantly smaller upper bound on the number of points on the separator than that obtained for planar graphs. Our result is stated as the following theorem.

THEOREM 2. *For a set P of n grid points on the two-dimensional (2D) plane, there is a line on the plane and a subset $Q \subseteq P$ of cardinality $\leq 1.129\sqrt{n}$ such that each half-plane determined by the line contains at most $\frac{2}{3}n$ points of P, and every two* *points $p_1, p_2 \in P$ on the different sides of the line have distance > 1, unless at least one of p_1, p_2 is in Q.*

Furthermore, we also provide $O(n^2)$ possible locations to find such a line based on the folding region within a fixed $n \times n$ square. This makes it possible to use the separator theorem in the algorithm for the protein folding problem, even though the locations of the letters are not known. The approximation method in searching for the separator region highly depends on the linear structure of the separator. We derive a similar separator for three-dimensional (3D) grid points, which is also applied to the 3D protein folding problem.

The lower bounds of $1.555\sqrt{n}$ and $1.581\sqrt{n}$ for the $\frac{2}{3}$-separator like that in Theorem 2 for the planar graph were proved by Djidev and Venkatesan [10] and Smith and Wormald [37], respectively. However, we develop a new approach and obtain an upper bound on the separator for a grid graph with a size smaller than their lower bounds. This shows that our smaller-sized separator for grid graphs cannot be directly obtained from that for planar graphs.

Our development of the separator technology is motivated by finding fast exact algorithms for the protein folding problem. A protein can be folded into a specific 3D structure, which is uniquely determined by the sequence of amino acids. Its 3D structure determines its function. Protein structure prediction with computational technology is one of the most significant problems in bioinformatics.

A simplified representation of proteins is a lattice conformation, which is a self-avoiding sequence in Z^3. An important representative of lattice models is the HP-model, which was introduced in [20, 21]. This problem was proven to be NP-hard both on two and three dimensions [6, 8].

Some algorithms for this problem have been developed based on heuristic, genetic, Monte Carlo, branch, and bound methods (e.g., [39, 40, 41, 36, 30, 33, 18, 19, 31, 22, 34, 5, 4]). Although many experimental results were reported for testing sequences of small lengths, we have not seen any theoretical analysis about upper bounds on the computational time of the algorithms. Another approach is to develop polynomial time approximation algorithms for protein folding in the HP-model, [17, 1, 28]. Hart and Istrail [17] showed a polynomial time $\frac{3}{8}$-approximation algorithm for the 3D protein folding in the HP-model and Newman [28] derived a polynomial time $\frac{1}{3}$-approximation algorithm for the 2D problem, improving the $\frac{1}{4}$-approximation algorithm in [17].

If the first letter of an HP-sequence is fixed at a position of a 2D plane (or 3D space), we have at least 2^{n-1} (3^{n-1}) ways and at most $3^{n-1}(5^{n-1})$ ways to put the rest of the letters on the plane (in the space, resp.). The computational time of our algorithm is bounded by $2^{O(n^{\frac{1}{2}} \log n)}$ $(2^{O(n^{\frac{2}{3}} \log n)}$ $)$ in two dimensions (in three dimensions, resp.). As the average number of amino acids of proteins is between 400 and 600, if an algorithm could solve the protein structure prediction problem with ≤ 1000 amino acids, it would be able to satisfy most of the application demands. Our effort is a theoretical step toward this target. Our algorithm is a divide-and-conquer approach, which is based on our geometric separator for separating the points in a d-dimensional grid.

The paper is organized as follows. In sections 2 and 3, we develop the separator theory. In sections 4 and 5, we apply the separators to the protein folding problem in the HP-model. In section 2, we show a class of easy separators on a set of d-dimensional grid points. This kind of separator is used in section 4 to obtain a $2^{O(n^{1-1/d} \log n)}$-time algorithm for the d-dimensional protein folding problem in the HP-model. In section 3, we develop sharp separators in both two and three dimensions. Those separators are used to obtain faster algorithms for the protein folding problem in section 5. Precisely, those separators help us reduce the constant factor in the exponents of the computational complexity of the protein folding problem.

In this paper, we only apply the separators for grid points to the protein folding problem in the HP-model. When developing algorithms for some geometric problems with the input of a set of points in the Euclidean space, we can select a set of grid points to characterize the distribution of the input points. This brings more applications of separators for grid points. A series of advances [11, 7, 13, 12] has been made along this line of the separator technology, which starts from the earlier version [14] of this paper. The method of this paper was extended and applied to a class of other NP-hard geometric problems by Fu [11], improving their exact algorithms to $2^{O(\sqrt{n})}$-time from $n^{O(\sqrt{n})}$-time. Those problems include the problems of disk covering, maximum independent set, minimum vertex cover, and minimum dominating set on disk graphs. An efficient sublinear time randomized algorithm was developed in [12] for finding separators. The method was also applied to derive an approximation algorithm for a geometric problem [7] that has application in digital image half-toning.

2. An easy separator for grid points.

Given a set of n grid points P, we will show that there is a hyperplane (denoted by $P_{r,a}$ for some r with $1 \le r \le d$ and an integer a in the definition below), which contains $O(n^{1-\frac{1}{d}})$ grid points from P, to partition P into two parts of at most $c(d)n$ grid points on each side, where $0 < c(d) < 1$ and $c(d)$ is a constant for fixed d. The separator in this section has a self-contained proof and is used in deriving an $n^{O(n^{1-\frac{1}{d}})}$-time algorithm for the protein folding problem in the HP-model in section 4. Let the dimensional number d be fixed. We need the following terms.

Definition 3.

- *For a set A, $|A|$ denotes the number of elements in A.*
- *The integer set is represented by $Z = \{\cdots, -2, -1, 0, 1, 2, \cdots\}$. For integers i and j, integer interval $[i, j] = \{i, i+1, \cdots, j\}$. For integers $x_1, \cdots, x_d$, $(x_1, \cdots, x_d)$ is a d-dimensional grid point.*
- *For two points p_1, p_2 with the same dimension,* $\mathrm{dist}(p_1, p_2)$ *is the Euclidean distance between them.*
- *An r-plane is the set $P_{r,a} = \{(x_1, \cdots, x_{r-1}, a, x_{r+1}, \cdots, x_d) | x_1, \cdots, x_{r-1}, x_{r+1}, \cdots, x_d \in Z\}$, which has all of the elements in Z^d with the rth component being a fixed value a.*
- $P_{r,<a} = \{(x_1, \cdots, x_{r-1}, x_r, x_{r+1}, \cdots, x_d) | x_1, \cdots, x_{r-1}, x_r, x_{r+1}, \cdots, x_d \in Z$ and $x_r < a\}$.
- $P_{r,>a} = \{(x_1, \cdots, x_{r-1}, x_r, x_{r+1}, \cdots, x_d) | x_1, \cdots, x_{r-1}, x_r, x_{r+1}, \cdots, x_d \in Z$ and $x_r > a\}$.
- $P_{r,\le a} = Pr_{r,<a} \cup P_{r,a}$*, and* $P_{r,\ge a} = P_{r,>a} \cup P_{r,a}$.
- *For a set of points S in the d-dimensional space and $1 \le r \le d$ and $a \in Z$, define $S(r, < a) = \{(x_1, \cdots, x_d) \in S | x_r < a\}$, $S(r, = a) = \{(x_1, \cdots, x_d) \in S | x_r = a\}$, and $S(r, > a) = \{(x_1, \cdots, x_d) \in S | x_r > a\}$.*
- *For $0 < c < 1$ and a set S in the d-dimensional space, a $P_{r,a}$ is a c-balanced-separator if $|S(r, < a)| \le c \cdot |S|$ and $|S(r, > a)| \le c \cdot |S|$.*

Theorem 4. *For a set S of n grid points in the d-dimensional space, there is a $c(d)$-balanced-separator P^* that contains at most $\le c'(d)n^{1-\frac{1}{d}}$ points from S, where $0 < c(d) < 1$, $0 < c'(d)$ and both $c(d)$ and $c'(d)$ are constants for a fixed dimensional number d.*

Proof. We will construct a series of sets $S = S_0 \supseteq S_1 \supseteq S_2 \supseteq \cdots \supseteq S_t$ such that $t \le d-1$ and $|S_i| \ge \frac{1}{2}|S_{i-1}|$ for $i = 1, 2, \ldots, t$. The construction of P^* starts from Stage 0 and can go up to Stage d.

Stage 0: Let $S_0 = S$ and $r = 1$. Enter Stage 1. **End of Stage 0.**

Stage $\boldsymbol{r}$ $\boldsymbol{(1 \le r \le d-1)}$: Let Q_r contain all of $P_{r,a}$ such that $P_{r,a}$ is a $\frac{3}{4}$-balanced-separator for S_{r-1}. At most $\frac{1}{4}$ of the elements in S_{r-1} with smallest a values (for the rth component) stay to the left of all the $\frac{3}{4}$-balanced separators and at most $\frac{1}{4}$ of the elements in S_{r-1} with largest a values (for the rth component) stay to the right of all the $\frac{3}{4}$-separators. The set $\cup_{P_{r,a} \in Q_r} P_{r,a}$ has at least $\frac{1}{2}$ of the elements from S_{r-1}. Thus, Q_r is not empty. If a $P_{r,a}$ in Q_r contains no more than $n^{1-\frac{1}{d}}$ elements from S, let $P^* = P_{r,a}$ and terminate the construction. We have $|S_{r-1}| \ge \frac{1}{2^{r-1}}|S|$ and

$$|S(r, < a)| \le |S_{r-1}(r, < a)| + |S - S_{r-1}| \le \frac{3}{4}|S_{r-1}| + |S| - |S_{r-1}| \tag{1}$$

$$= |S| - \frac{1}{4}|S_{r-1}| \le \left(1 - \frac{1}{2^{r+1}}\right)|S| \le \left(1 - \frac{1}{2^d}\right)|S|. \tag{2}$$

Similarly, $|S(r, > a)| \le (1 - \frac{1}{2^d})|S|$.

If every $P_{r,a} \in Q_r$ has $> n^{1-\frac{1}{d}}$ elements from S, $|Q_r| \le n^{\frac{1}{d}}$ because $|\cup_{P_{r,a} \in Q_r} (P_{r,a} \cap S)| \le |S| = n$ and all planes in Q_r are disjoint from each other. It is easy to see that there is an integer interval $[c_1, c_2]$ such that $Q_r = \{P_{r,a} | a \in [c_1, c_2]\}$. Let $S_r = \cup_{P_{r,a} \in Q_r} (P_{r,a} \cap S_{r-1})$. We have $S_r \subseteq S_{r-1}$ and $|S_r| \ge |S_{r-1}|/2$ (because $[c_1, c_2]$ is the set of all the integers a such that $P_{r,a}$ is a $\frac{3}{4}$-balanced-separator). Let $r = r + 1$ and go to the next stage. **End of stage $\boldsymbol{r}$.**

Stage $\boldsymbol{d}$: Assume that for each r with $1 \le r \le d-1$, Q_r has no plane $P_{r,a}$ with elements $\le n^{1-\frac{1}{d}}$ from S. Hence, $|Q_r| \le n^{\frac{1}{d}}$ for $1 \le r \le d-1$. If a is fixed, every $p \in P_{r,a}$ has the rth component equal to a. Therefore, $\{x_r | x_r$ is the rth component of some $p \in P_{r,a}$ for some $P_{r,a} \in Q_r\}$ has $\le n^{\frac{1}{d}}$ elements since $|Q_r| \le n^{\frac{1}{d}}$ $(1 \le r \le d-1)$. This implies that for every $P_{d,a}$,

$$|\{p | p \in P_{r,a_r} \text{ for some } P_{r,a_r} \in Q_r \ (r = 1, \ldots, d-1) \text{ and } p \in P_{d,a}\}| \le (n^{\frac{1}{d}})^{d-1} = n^{\frac{d-1}{d}}.$$

As $|S_{d-1}| \ge \frac{|S|}{2^{d-1}} = \frac{1}{2^{d-1}} n$, there are at least $\frac{\frac{1}{2}|S_{d-1}|}{n^{1-\frac{1}{d}}} \ge \frac{1}{2^d} \cdot n^{\frac{1}{d}}$ many $P_{d,a}$'s to be $\frac{3}{4}$-balanced-separators for S_{d-1}. One of them has at most $\frac{|S|}{\frac{1}{2^d} n^{\frac{1}{d}}} = 2^d n^{1-\frac{1}{d}}$ elements from S. Let P^* be such a $P_{d,a}$. As $|S_{d-1}| \ge \frac{1}{2^{d-1}}|S|$, we have

$$|S(d, < a)| \le |S_{d-1}(d, < a)| + |S - S_{d-1}| \le \frac{3}{4}|S_{d-1}| + |S| - |S_{d-1}| \tag{3}$$

$$= |S| - \frac{1}{4}|S_{d-1}| \le \left(1 - \frac{1}{2^{d+1}}\right)|S|. \tag{4}$$

Similarly, we also have $|S(d, > a)| \le (1 - \frac{1}{2^{d+1}})|S|$. **End of stage $\boldsymbol{d}$.** □

For a d-dimensional cube that contains n grid points, its edge length is $n^{\frac{1}{d}}$. Every hyperplane $P_{r,a}$ which intersects the cube shares $n^{\frac{d-1}{d}}$ grid points with the cube. This shows that it is impossible to improve the upper bound on the number of points on the separator to $o(n^{\frac{d-1}{d}})$. In the next section, we shows that we can improve the separator by a constant factor. Theorem 4 indicates that the balanced separator can be found among $O(dn)$ axis-parallel hyperplanes.

3. Sharp separators for grid points.

We will improve the quality of the separator obtained in the previous section. The following lemma is a well-known fact (see [29]) that will be used for deriving our new separator. Our reduced upper bound on the number of points on the separator is from the following fact: For a set P of 2D grid points with the centerpoint o (see Lemma 5), a random line through o has the largest expected number of points of P with distance $\leq a$ to the line when the points in P are tightly arranged in the grid points inside a circle with the least radius. This is also true in the higher dimensional space.

LEMMA 5. *For an n-element set P in the d-dimensional space, there is a point q with the property that any half-space that does not contain q covers at most $\frac{d}{d+1}n$ elements of P. (Such a point q is called a centerpoint of P.)*

DEFINITION 6. *For a grid point (i, j) on the 2D plane, its grid square is a 1×1 square with four corner points $(i-\frac{1}{2}, j-\frac{1}{2}), (i-\frac{1}{2}, j+\frac{1}{2}), (i+\frac{1}{2}, j-\frac{1}{2})$, and $(i+\frac{1}{2}, j+\frac{1}{2})$. For a 3D grid point (i, j, k), its grid cube is a $1 \times 1 \times 1$ cube with eight corner points in $\{(i+\alpha, j+\beta, k+\gamma) | \alpha, \beta, \gamma \in \{-\frac{1}{2}, \frac{1}{2}\}\}$.*

3.1. 2D separators.

LEMMA 7. (1) *A circle of radius r contains at most $\pi(r + \frac{\sqrt{2}}{2})^2$ grid points.*

(2) *A circle of radius r on the 2D plane has at least $\pi r^2 - 4\sqrt{2}\pi r$ grid points inside it.*

(3) *A circle of radius $\frac{1}{\sqrt{\pi}}\sqrt{n} + 4\sqrt{2}$ has at least n grid points in it.*

(4) *For every line segment L of length m, the number of grid points with distance $\leq a$ to at least one point of L is $\leq (2a + \sqrt{2})(m + 2a + \sqrt{2})$.*

(5) *For every line L and fixed $a > 0$, there are at most $(2a+\sqrt{2})(\sqrt{2}n+2a+\sqrt{2})$ grid points inside an $n \times n$ square with $\leq a$ distance to L.*

Proof. (1) If a grid point p is inside a circle C of radius r at center o, the 1×1 grid square with center at p is inside a circle C' of radius $r + \frac{\sqrt{2}}{2}$ at the same center o. The number of those 1×1 grid squares for the grid points inside C is no more than the area size of the circle C'.

(2) Let C_1, C, and C_2 be three circles on the plane with the same center. Their radii are $r - \sqrt{2}, r$, and $r + \sqrt{2}$, respectively. Every 1×1 grid square intersecting C's boundary is outside C_1 and inside C_2. The number of grid squares intersecting C's boundary is no more than $\pi(r + \sqrt{2})^2 - \pi(r - \sqrt{2})^2 = 4\sqrt{2}\pi r$.

(3) Let $r = \frac{1}{\sqrt{\pi}}\sqrt{n} + 4\sqrt{2}$. It is straightforward to verify that $\pi r^2 - 4\sqrt{2}\pi r > n$. Apply (2).

(4) If a point p has $\leq a$ distance to L, every point in the 1×1 grid square with center at p has distance $\leq a + \frac{\sqrt{2}}{2}$ to L. The number of those 1×1 squares with centers at points of distance $\leq a$ to L is no more than $2(a + \frac{\sqrt{2}}{2})(m + 2a + \sqrt{2})$.

(5) The length of a line L inside an $n \times n$ square is $\leq \sqrt{2}n$. Apply (4). □

DEFINITION 8. *Assume that $a > 0$ and that p_0, p are two points on the plane. Define $Pr_2(a, p_0, p)$ to be the probability that a point p has $\leq a$ perpendicular distance to a random line L through the point p_0.*

LEMMA 9. *Let $a > 0$ be a constant and $\delta > 0$ be a small constant. Let P be a set of points in a 2D grid. Assume that all the points of P are inside a circle of radius r with center at point o. For a random line passing through o, the expected number of points in P with distance $\leq a$ to L is bounded by $4ar + \delta r$ for all large r.*

Proof. Assume that $p = (x, y)$ is a point of P and that L is a random line passing through the center $o = (x_0, y_0)$. Let C be the circle of radius r at center o such that C covers all the points in P. Let C' be the circle of radius $r' = r + \frac{\sqrt{2}}{2}$ at the same center o. It is easy to see that every unit square with center at a point in P is inside C'. The probability that a point p has distance $\leq a$ to L is $\frac{2\arcsin\frac{a}{\mathrm{dist}(o,p)}}{\pi}$.

Let $\epsilon > 0$ be a small constant which will be determined later. Select r_0 to be large enough such that for every point p with $\mathrm{dist}(o,p) \geq r_0$, $\arcsin\frac{a}{\mathrm{dist}(o,p)} < (1+\epsilon)\frac{a}{\mathrm{dist}(o,p)}$ and $\frac{1}{\mathrm{dist}(o,p')} < \frac{1+\epsilon}{\mathrm{dist}(o,p)}$ for every point p' with $\mathrm{dist}(p',p) \leq \frac{\sqrt{2}}{2}$. Let P_1 be the set of all the points p in P such that $\mathrm{dist}(o,p) < r_0$. By Lemma 7, the number of grid points in P_1 is no more than $\pi(r_0+\frac{\sqrt{2}}{2})^2$. For each point $p \in P_1$, $Pr_2(a,o,p) \leq 1$. For every point $p \in P - P_1$, $Pr_2(a,o,p) = \frac{2\arcsin\frac{a}{\mathrm{dist}(o,p)}}{\pi} \leq \frac{(1+\epsilon)2a}{\pi\mathrm{dist}(o,p)}$.

The expected number of points in P with distance $\leq a$ to a random line through the point o is

$$\sum_{p\in P} Pr_2(a,o,p) = \sum_{p\in P_1} Pr_2(a,o,p) + \sum_{p\in P-P_1} Pr_2(a,o,p) \tag{5}$$

$$\leq \sum_{p\in P_1} 1 + \sum_{p\in P-P_1} \frac{2\arcsin\frac{a}{\mathrm{dist}(o,p)}}{\pi} \tag{6}$$

$$< \pi\left(r_0+\frac{\sqrt{2}}{2}\right)^2 + \sum_{p\in P-P_1}\frac{(1+\epsilon)2a}{\pi\mathrm{dist}(o,p)} \tag{7}$$

$$\leq \pi\left(r_0+\frac{\sqrt{2}}{2}\right)^2 + \frac{2a(1+\epsilon)^2}{\pi}\int\int_{C'}\frac{1}{\mathrm{dist}(o,p)}d_xd_y \tag{8}$$

$$= \frac{2a(1+\epsilon)^2}{\pi}\int_0^{2\pi}\int_0^{r'}\frac{\rho}{\rho}d_\rho d_\theta + \pi\left(r_0+\frac{\sqrt{2}}{2}\right)^2 \tag{9}$$

$$= 4a(1+\epsilon)^2r' + \pi\left(r_0+\frac{\sqrt{2}}{2}\right)^2 \tag{10}$$

$$< 4ar + \delta r \text{ for all large } r \text{ by selecting } \epsilon \text{ small enough.} \tag{11}$$

We use the transformation $x = \rho\cos\theta + x_0, y = \rho\sin\theta + y_0$ to convert the integral at (8) to that at (9) above. □

THEOREM 10. *Let $a > 0$ be a constant and $\epsilon > 0$ be a small constant. For a set P of n grid points in a 2D grid, there is a line L such that P has at most $(\frac{4a}{\sqrt{\pi}})\cdot\sqrt{n}+\epsilon\sqrt{n}$ points with distance $\leq a$ to L, and each half-plane divided by L has at most $\frac{2}{3}n$ points from P.*

Proof. Assume that the centerpoint of P is at the point o (see Lemma 5). We are going to estimate the upper bound for the expected number of points in P that have $\leq a$ distances to a random line L through o.

Let $r = \frac{1}{\sqrt{\pi}}\sqrt{n} + 4\sqrt{2}$. By Lemma 7, the circle C at center o with radius r contains at least n grid points. Let f be a one-to-one mapping from P to the set of grid points inside C such that $f(p) = p$ for every $p \in P$ with $\mathrm{dist}(o,p) \leq r$. Therefore, f moves those points of P outside the circle C to the inside. It is easy to see that if $\mathrm{dist}(o,p_1) \leq \mathrm{dist}(o,p_2)$, then, $Pr_2(a,o,p_1) \geq Pr_2(a,o,p_2)$. The expected number of points in P with $\leq a$ distance to L is $\sum_{p\in P} Pr_2(a,o,p)$.

By Lemma 9, $\sum_{p\in P} Pr_2(a,o,p) \leq \sum_{p\in P} Pr_2(a,o,f(p)) \leq 4ar+\delta r = \frac{4a}{\pi}\sqrt{n}+\epsilon\sqrt{n}$ by selecting a small δ. □

3.2. 3D separators. The technology used in the previous section can be easily extended to the 3D grid. We give a brief proof for the case in the 3D space.

LEMMA 11. *Let $a = \sqrt{3}$. (1) A sphere of radius r has at least $\frac{4}{3}\pi r^3 - \frac{4}{3}\pi(6ar^2 + 2a^3)$ grid points. (2) A sphere of radius $(\frac{3}{4\pi})^{\frac{1}{3}} n^{\frac{1}{3}} + 7a$ contains at least n grid points.*

Proof. (1) Let $r_1 = r + a$, and let $r_2 = r - a$. The volume difference between the sphere of radius r_1 and the sphere of radius r_2 is $\frac{4}{3}\pi(6ar^2 + 2a^3)$, which is larger or equal to the number of unit grid cubes intersecting the boundary of the sphere of radius r. (2) For $r = (\frac{3}{4\pi})^{\frac{1}{3}} n^{\frac{1}{3}} + 7a$, we have $\frac{4}{3}\pi r^3 - \frac{4}{3}\pi(6ar^2 + 2a^3) \geq n$. □

DEFINITION 12. *Assume that $a > 0$ and that p_0, p are two points in the 3D Euclidean space. Define $Pr_3(a, p_0, p)$ to be the probability that the point p has $\leq a$ perpendicular distance to a random plane L through the point p_0 in the 3D space.*

Assume that both a and p_0 are fixed. We want to compute $Pr_3(a, p_0, p)$, which depends on the parameter a and the distance between p_0 and p. Without loss of generality, we assume that p_0 is the origin point $(0, 0, 0)$ and $p = (x, 0, 0)$, where $x = \text{dist}(p_0, p)$. A random plane through the origin point is uniquely determined by its normal vector (u, v, w) with $u \geq 0$. The distance between p to the plane with normal vector (u, v, w) is equal to xu. If the distance is at most a, then $u \leq \frac{a}{x}$. The set $G_{p,a} = \{(u, v, w) | u^2 + v^2 + w^2 = 1 \text{ and } 0 \leq u \leq \frac{a}{x}\}$ contains all the normal vectors of those planes (through the origin) such that p has distance at most a to each of them. The set $G_{p,a}$ is a subarea of the half-sphere $H_1 = \{(u, v, w) | u^2 + v^2 + w^2 = 1 \text{ and } 0 \leq u\}$ with center at the origin point and radius 1. If a is fixed and x is large, the area size of $G_{p,a}$ can be computed by the formula

$$\int_0^{\frac{a}{x}} 2\pi\sqrt{1 - y^2} d_y = \frac{2\pi a}{x} + O\left(\frac{a^2}{x^2}\right).$$

Since the area size of a half-sphere of radius 1 is 2π, the probability that p has distance at most a to a random plane through the origin is

$$Pr_3(a, p_0, p) = \frac{\text{the area size of } G_{p,a}}{\text{the area size of } H_1} = \frac{\frac{2\pi a}{x} + O(\frac{a^2}{x^2})}{2\pi} = \frac{a}{x} + O\left(\frac{a^2}{x^2}\right).$$

The above formula for computing $Pr_3(a, p_0, p)$ corrects a mistake that we made in the extended abstract of this paper [14]. It also gives a slightly different upper bound for the exact algorithm for the folding problem in the 3D space reported in [14].

LEMMA 13. *Let $a > 0$ be a constant and let $\delta > 0$ be a small constant. Let P be a set of points in a 3D grid. Assume that all the points of P are inside a sphere of radius r with center at point o. For a random plane passing through o, the expected number of points in P with distance at most a to L is bounded by $2\pi ar^2 + \delta r^2$ for all large r.*

Proof. The proof is very similar to that of Lemma 9. Let S be the sphere with radius r at center $o = (x_0, y_0, z_0)$ such that it contains all the points in P. Let S' be the sphere of radius $r' = r + \frac{\sqrt{3}}{2}$ at the same center of S. All unit cubes with center at points in P are inside S'.

The expected number of points in P with distance $\leq a$ to a random plane through o is $\sum_{p=(x,y,z)\in P} Pr_3(a, o, p)$, which has the main part $\int\int\int_{S'} \frac{a}{\text{dist}(a,o,p)} d_x d_y d_z$. By the transformation $x = \rho \sin\theta \cos\alpha + x_0, y = \rho \sin\theta \cos\alpha + y_0, z = \rho \sin\theta + z_0$, we have $\int\int\int_{S'} \frac{a}{\text{dist}(a,o,p)} d_x d_y d_z = \int_0^{r'} \int_0^{\pi} \int_0^{2\pi} \frac{a\rho^2 \sin\theta}{\rho} d_\alpha d_\theta d_\rho = 2\pi a r'^2$. □

THEOREM 14. *Let $a > 0$ be a constant and let $\epsilon > 0$ be a small constant. For a set P of n points in a 3D grid, there is a plane L such that P has at most*

$(2\pi a(\frac{3}{4\pi})^{2/3}) \cdot n^{2/3} + \epsilon n^{2/3}$ points with distance at most a to L, and each half-space divided by L has at most $\frac{3}{4}n$ points from P.

Proof. By Lemma 11, the sphere of radius $(\frac{3}{4\pi})^{\frac{1}{3}}n^{\frac{1}{3}} + 7\sqrt{3}$ contains at least n grid points. Moving points of P into the sphere, which has center at the centerpoint of P (see Lemma 5), from the outside increases the probability to have distance $\leq a$ to a random plane through the sphere center. By Lemma 13, the expected number of points in P with distance $\leq a$ to a random plane is $(2\pi a(\frac{3}{4\pi})^{2/3}) \cdot n^{2/3} + \epsilon n^{2/3}$ for all large n via selecting small δ. □

4. An application of the easy separators to the protein folding problem.

We apply the easy separators to the protein folding problem in the HP-model, and obtain the first subexponential time algorithm for it. We have already shown that there is a small set of letters with size $O(n^{1-\frac{1}{d}})$ on a hyperplane to partition the folding problem of n letters into 2 smaller problems of $\leq c(d)n$ letters, where $0 < c(d) < 1$, $c(d)$ is a constant for fixed d, and n is the size of the input (the number of H and P characters). The 2 smaller problems are recursively solved, and their solutions are merged to derive the solution to the original problem. As the separator has only $O(n^{1-\frac{1}{d}})$ letters, there are at most $n^{O(n^{1-\frac{1}{d}})}$ cases to partition the problem.

DEFINITION 15.

- *For a d-dimensional point $(x_1, \cdots, x_d)$, define $||(x_1, \cdots, x_d)|| = \sum_{i=1}^{d} |x_i|$.*
- *For a set Σ of letters, a Σ-sequence is a sequence of letters from Σ. For example, $PHPPHHPH$ is an $\{H, P\}$-sequence. For a sequence S of length n and $1 \leq i \leq n$, $S[i]$ is the ith letter of S and $S[i,j]$ denotes the subsequence $S[i]S[i+1] \cdots S[j]$. If $[i_1, j_1], [i_2, j_2], \cdots, [i_t, j_t]$ are pairwise disjoint intervals inside $[1,n]$, we call $S[i_1, j_1], S[i_2, j_2], \cdots, S[i_t, j_t]$ disjoint subsequences of S. For a set of integers $A = \{i_1 < i_2 < \cdots < i_k\}$, define $S[A] = S[i_1]S[i_2] \cdots S[i_k]$.*
- *A self-avoiding arrangement f for a sequence S of length n in the d-dimensional grid is a one-to-one mapping from $\{1, 2, \cdots, n\}$ to Z^d such that $||f(i) - f(i+1)|| = 1$ for $i = 1, 2, \ldots, n-1$. For the disjoint subsequences $S[i_1, j_1], \cdots, S[i_k, j_k]$ of S, a partial self-avoiding arrangement of S on $S[i_1, j_1], \cdots, S[i_k, j_k]$ is a partial function f from $\{1, 2, \cdots, n\}$ to Z^d such that f is defined on $\cup_{t=1}^{k}[i_t, j_t]$, and f can be extended to a (full) self-avoiding arrangement of S on Z^d.*
- *For a grid self-avoiding arrangement, its contact map is the graph $G_f = (1, 2, \cdots, n, E)$, where the edge set $E = \{(i,j) : |i-j| > 1 \text{ and } ||f(i) - f(j)|| = 1\}$.*
- *A rectangular region R in a d-dimensional space is the intersection of a finite number of sets $P_1, P_2, \cdots, P_k$, where $P_i = P_{r,<a}$ or $P_i = P_{r,>a}$ with $1 \leq r \leq d$ and $a \in Z$ for $i = 1, \ldots, k$.*
- *A rectangular region R in a d-dimensional space is of size $m_1 \times m_2 \times \cdots \times m_d$ if $m_i = max\{x_i - x_i' | (x_1, \cdots, x_d), (x_1', \cdots, x_d') \in R\} + 1$ for $i = 1, \ldots, d$.*

As we are going to describe our algorithm recursively, we use the following term to characterize the problem. A d-dimensional *multisequence folding problem F* is formulated as follows.

The inputs are

1. a list of disjoint subsequences $S_1, S_2, \cdots, S_k$ of sequence S_0 ($S_t = S_0[i_t, j_t]$ for $t = 1, \ldots, k$),
2. a rectangular region R, where all of the k $\{H, P\}$-sequences are going to be arranged,

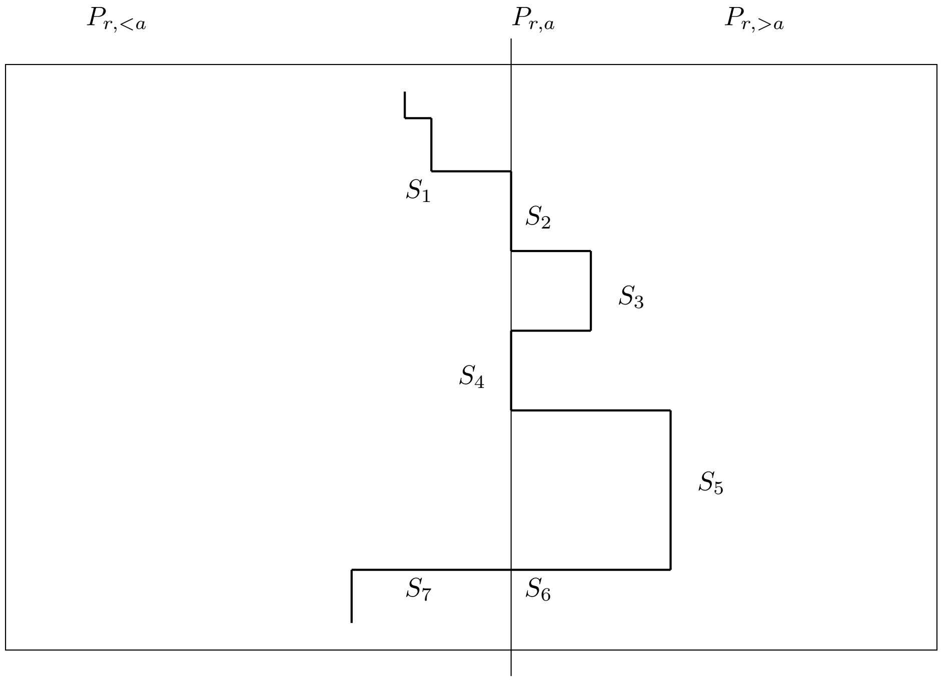


FIG. 2. *The hyperplane $P_{r,a}$ partitions a sequence into 3 groups of disjoint subsequences $\{S_1, S_7\}, \{S_2, S_4, S_6\}$, and $\{S_3, S_5\}$ in $P_{r,<a}, P_{r,a}$, and $P_{r,>a}$, respectively. (Notice that S_6 is a point that is the intersection between the two lines.)*

3. a series of k pairs of points in R: $(p_1, q_1), (p_2, q_2), \cdots, (p_k, q_k)$, in which points $p_t \in R$ and $q_t \in R$ are the positions for putting the first and the last letters of S_t, respectively,
4. a set of available points to put the letters from the k sequences, and
5. a set of $\{H, P\}$ points, which already have letters H and P from $S_0[([1, n] - \cup_{t=1}^{k}[i_t, j_t])]$.

The output is a partial self-avoiding arrangement f of S_0 on $S_1, \cdots, S_k$ in the rectangular region R that satisfies $f(i_t) = p_t, f(j_t) = q_t\,(t = 1, 2, \ldots, k)$ and has the maximal number of H-H contacts, where $f(i)$ is an available point for each $i \in \cup_{t=1}^{k}[i_t, j_t]$. H-H contacts may happen between two available neighbor positions or between an available position and a nonavailable position after the arrangement.

A hyperplane $P_{r,a}$ partitions a multisequence folding problem F into two multisequence folding problems F_1 and F_2 in regions $R \cap P_{r,\leq a}$ and $R \cap P_{r,\geq a}$, respectively, by fixing some letters on $P_{r,a}$ (see Figure 2). Furthermore, the available points of F_1 (F_2, resp.) are the intersection of F's available points with $P_{r,<a}$ ($P_{r,>a}$, resp.).

Algorithm

```
(a) Input a $d$-dimensional multisequence folding problem $F$ (as defined above);
(b)     For each subset $S$ of $\leq c'(d) \cdot n^{\frac{d-1}{d}}$ letters from $S_1, \cdots, S_k$
            For every plane $P_{r,a}$ (with nonempty intersection with $R$) and
              For every arrangement of $S$ at available points on $P_{r,a} \cap R$
(c)           Begin
(d)             For each partition (by $P_{r,a}$) making $F$ into problems $F_1$ and $F_2$ of
                    size $\leq c(d)n$
```

```
(e)                 Begin
(f)                     Solve $F_1$ and $F_2$ recursively using this algorithm (use the brute
                        force method when the problem size is small);
(g)                     Merge the solutions to $F_1$ and $F_2$ to get a potential solution
                        for $F$;
(h)                 End
(i)           End
(j) Output the solution for $F$ with the maximal number of $H$-$H$ contacts among
      all of the potential solutions for $F$;
```

End of the Algorithm

LEMMA 16. *There is an* $(nm)^{O(n^{1-\frac{1}{d}})}$*-time algorithm for the d-dimensional multisequence folding problem with an* $m_1 \times m_2 \times \cdots \times m_d$ *rectangular region in the HP-model, where* $m = \max\{\max\{m_i | i = 1, \ldots, d\}, 2\}$ *and the dimension* d *is assumed to be a constant.*

Proof. By Theorem 4, the folding problem is partitioned into two problems with a separator of size $\leq c'(d) \cdot n^{1-\frac{1}{d}}$ elements. For each $1 \leq r \leq d$, we have at most m planes $P_{r,a}$ that have a nonempty intersection with the $m_1 \times m_2 \cdots \times m_d$ rectangular region. There are at most $d \cdot m$ ways to select the separator plane. If the plane has at most t letters, there are at most $d \cdot m \cdot n^t m^{(d-1)t} = dn^t m^{(d-1)t+1}$ ways to select the plane and the position for the letters and put those letters at the selected position on the plane. Thus, the loop (c)–(i) is repeated $\leq dn^t m^{(d-1)t+1}$ times.

For disjoint subsequences $S_1, \cdots, S_k$ of S_0 inside a rectangular region R, we fix $t \leq c'(d) \cdot n^{1-\frac{1}{d}}$ letters from $S_1, \cdots, S_k$ on the hyper plane $P_{r,a}$ and partition them into three groups of subsequences of S_0 which are in $R \cap P_{r,<a}$, $R \cap P_{r,=a}$, and $R \cap P_{r,>a}$, respectively (see Figure 2). For each subsequence from $R \cap P_{r,<a}$ or $R \cap P_{r,>a}$, we fix the positions for its two endpoints under all possible cases. The subsequences in $R \cap P_{r,<a}$ will not affect those in $R \cap P_{r,>a}$. We have at most 2^{t+1} ways to fix the endpoints of those sequences in $R \cap P_{r,<a}$ and $R \cap P_{r,>a}$. Therefore, the loop (e)–(h) is repeated $\leq 2^{t+1}$ times.

Let $T(m, n)$ be the computational time of our algorithm, where n is the length of S_0 and m is defined as in the lemma. We have the following recursive relationship for the total time of the algorithm:

$$T(m, n) \leq 2 \cdot d \cdot m^{(d-1)c'(d)n^{1-\frac{1}{d}}+1} \cdot n^{c'(d)n^{1-\frac{1}{d}}} \cdot 2^{c'(d)n^{1-\frac{1}{d}}+1} \cdot T(m, c(d)n),$$

where $0 < c(d) < 1$ and $0 < c'(d)$ are constants for fixed d. Expanding the inequality recursively, we have $T(m, n) = (nm)^{O(n^{1-\frac{1}{d}})}$. □

THEOREM 17. *There is a* $2^{O(n^{1-\frac{1}{d}} \log n)}$*-time algorithm for the d-dimensional protein folding problem in the HP-model for fixed d.*

Proof. The folding problem can be put into an $n \times n \times \cdots \times n$ rectangular region in the d-dimensional space by fixing the two middle letters in two central neighbor points in the region. By Lemma 16, we have an $n^{O(n^{1-\frac{1}{d}})} = 2^{O(n^{1-\frac{1}{d}} \log n)}$-time algorithm. □

5. Application of the sharp separators to the protein folding problem.

In the previous section, we show that the d-dimensional folding problem is computable in $O(2^{e(d)n^{1-\frac{1}{d}}})$ time, where $e(d)$ is constant for fixed d. We will reduce the constant $e(d)$ in this section using the sharp separators.

5.1. 2D folding algorithm. It is easy to see that Theorem 10 implies Theorem 2 by setting $a = \frac{1}{2}$. Assume that our input HP-sequence has n_0 letters and the optimal folding is inside an $m \times m$ square. Select a parameter $\epsilon > 0$. Add some points evenly on the four edges of the $m \times m$ square, so that every two neighbor points have distance $\leq \epsilon$. Those points are called *ϵ-regular points.* Every line segment connecting two ϵ-regular points is called an *ϵ-regular line segment.* An ϵ-regular line is a line containing two ϵ-regular points.

LEMMA 18. *Let $\epsilon > 0$ be a constant. Every line segment L_1 inside the $m \times m$ square has an ϵ-regular segment L_2 such that for every point $p_1 \in L_1$, there is a point $p_2 \in L_2$ with $\operatorname{dist}(p_1, p_2) \leq \epsilon$, and for every point $q_2 \in L_2$, there is a point $q_1 \in L_1$ with $\operatorname{dist}(q_1, q_2) \leq \epsilon$.*

Proof. Assume E_1, E_2, E_3, and E_4 are the four edges of the $m \times m$ square. Assume L_1 intersects two of them inside the square at two points p_i and p_j of edges E_i and $E_j (i \neq j)$, respectively. Select the ϵ-regular point q_i closest to p_i from the edge E_i, and q_j closest to p_j from E_j. The ϵ-regular line segment L_2 results from connecting q_i and q_j. Every point p in L_1 has another point $p' \in L_2$ with distance $\leq \max(\operatorname{dist}(p_i, q_i), \operatorname{dist}(p_j, q_j)) \leq \epsilon$, and every point q in L_2 has another point in $q' \in L_1$ with distance $\leq \max(\operatorname{dist}(p_i, q_i), \operatorname{dist}(p_j, q_j)) \leq \epsilon$. □

LEMMA 19. *Let a and ϵ be positive constants. Let P be a set of n points in a $2D$ grid. There is an ϵ-regular line L such that there are $\leq (\frac{2}{3} + \epsilon)n$ points of P on each of the two half-planes, and $\leq 4(a + \epsilon)\frac{\sqrt{n}}{\sqrt{\pi}}$ points of P with distance $\leq a$ to L.*

Proof. Let $\delta > 0$ be a small constant. By Theorem 10, there is a line L such that the number of points of P with distance $a + \delta$ to it is bounded by $4(a + \delta)\frac{\sqrt{n}}{\sqrt{\pi}}$, and each side of L has at most $\frac{2}{3}n$ points in P. By Lemma 18, there is a line L' close to L such that every point in L has another point in L' with distance $\leq \delta$ and every point in L' has another point in L with distance $\leq \delta$. Every point with distance $\leq a$ to the line L' has distance $\leq a + \delta$ to L. Therefore, the number of points in P with distance $\leq a$ to L' is bounded by $4(a + \epsilon)\frac{\sqrt{n}}{\sqrt{\pi}}$, and each half-plane divided by L has at most $(\frac{2}{3} + \epsilon)n$ points in P if δ is small enough. □

LEMMA 20. *For some constants $c_0, \epsilon > 0$, there is an $O(m^{c_0 \log n} n_0^{(6.145 - \epsilon)\sqrt{n}})$-time algorithm for the $2D$ multisequence folding problem F in an $m \times m$ square, where n is the sum of the lengths of the input disjoint subsequences of S_0 and n_0 is the length of S_0.*

Proof. Let $a = 1/2$, $c = 2/3 + \delta$, and $d = \frac{4(a+\delta)}{\sqrt{\pi}}$, where $\delta > 0$ is a small constant which will be fixed later. We assume $m > 1$ and n is large. Let P be an optimal arrangement for the problem F. By Lemma 19, there is a line L such that P has at most $d\sqrt{n}$ points with distance $\leq 1/2$ to L, and each half-plane has at most cn points from P. The letters that stay at those positions with $\leq \frac{1}{2}$ distance to L form a separator for P. For every two letters at different sides of L that have a contact (their distance is 1), at least one of them has distance $\leq \frac{1}{2}$ to L. The algorithm is based on such a separator and is similar to that used in the previous section to find such an optimal solution P.

The number of δ-regular points at every edge of the $m \times m$ square is bounded by $\frac{m}{\delta}$. The total number of δ-regular lines is bounded by $u_1 = \binom{4}{2}(\frac{m}{\delta})^2$. By Stirling's formula, we have $(d\sqrt{n})! > \frac{(d\sqrt{n})^{d\sqrt{n}}}{2^{d\sqrt{n}}}$. There are $u_2 = \binom{n}{0} + \binom{n}{1} + \cdots \binom{n}{d\sqrt{n}} < d\sqrt{n}\frac{n^{d\sqrt{n}}}{(d\sqrt{n})!} < (\frac{2}{d})^{d\sqrt{n}} \cdot d\sqrt{n} \cdot n^{\frac{1}{2}d\sqrt{n}}$ ways to select $\leq d\sqrt{n}$ letters from those n letters among the input disjoint subsequences of S_0.

Assume k $(\leq d\sqrt{n})$ letters $S_0[i_1], S_0[i_2], \cdots, S_0[i_k]$ $(1 \leq i_1 < i_2 < \cdots < i_k \leq n)$ are fixed from the disjoint subsequences of S_0. By Lemma 7, there are at most

$\beta = (2a+\sqrt{2})(\sqrt{2}m+2a+\sqrt{2})$ positions (inside the $m \times m$ square) to put the letter $S_0[i_1]$ such that it has distance $\le a$ to L.

By Lemma 7, after the letter $S_0[i_j]$ is put at a grid point, there are at most $(2a+\sqrt{2})((i_{j+1}-i_j+2a)+2a+\sqrt{2}) \le (2a+\sqrt{2})(1+4a+\sqrt{2})(i_{j+1}-i_j)$ ways to arrange $S_0[i_{j+1}]$ so that its position has at most distance a to a point in L. Let $\alpha = (2a+\sqrt{2})(1+4a+\sqrt{2})$. After the position of the letter $S_0[i_1]$ is fixed, there are at most $\prod_{j=1}^{j=k-1}(\alpha(i_{j+1}-i_j))$ ways to put $S_0[i_2], S_0[i_3], \cdots, S_0[i_k]$ along the separation line with distance $\le a$. Since $k \le d\sqrt{n}$ and $1 \le i_1 < i_2 < \cdots < i_k \le n_0$,

$$\prod_{j=1}^{j=k-1} \left(\alpha\left(i_{j+1}-i_j\right)\right) \le \left(\alpha\left(\frac{n_0}{k-1}\right)\right)^{k-1} \tag{12}$$

$$\le \left(\alpha\left(\frac{n_0}{k}\right)\right)^{k} \tag{13}$$

$$\le \left(\alpha\left(\frac{n_0}{d\sqrt{n}}\right)\right)^{d\sqrt{n}} \tag{14}$$

$$\le \left(\frac{\alpha}{d}\right)^{d\sqrt{n}} n_0^{d\sqrt{n}} n^{-\frac{1}{2}d\sqrt{n}}. \tag{15}$$

The inequality (12) follows from the well-known fact that for positive variables $y_1, \cdots, y_{k-1}$ and fixed h with $y_1+\cdots+y_{k-1} \le h$, the product $\prod_{t=1}^{k-1} y_k$ is maximal when $y_1 = y_2 = \cdots = y_{k-1} = \frac{h}{k-1}$. The number of ways to arrange the k letters along the separation line (with distance $\le a$ to L) is bounded by

$$u_3 = \beta\left(\frac{\alpha}{d}\right)^{d\sqrt{n}} n_0^{d\sqrt{n}} n^{-\frac{1}{2}d\sqrt{n}}.$$

We have $T(n) \le u_1 \cdot u_2 \cdot u_3 \cdot T(cn)$. It implies that

$$T(n) \le \left(\frac{mn}{\delta}\right)^{c_0 \log n} 2^{c_0\sqrt{n}} n_0^{d(\frac{1}{1-\sqrt{c}})\sqrt{n}} = O(m^{c_0 \log n} n_0^{(6.145-\epsilon)\sqrt{n}})$$

by selecting constants ϵ, δ small enough and c_0 large enough. □

THEOREM 21. *There is an $O(n^{6.145\sqrt{n}})$-time algorithm for the 2D protein folding problem in the HP-model.*

Proof. Fix the two middle letters at the two central neighbor positions of an $n \times n$ square. Let the folding be inside the $n \times n$ square, and apply Lemma 20. □

5.2. 3D folding algorithm.

The technology used in the previous section can be easily extended to a 3D grid. We give a brief proof for the case in the 3D space.

Put some regular points on each side of the six faces of an $m \times m \times m$ cube (the folding region) so that every point on each face has $\le \epsilon$ distance to one regular point. Recall that these points are called ϵ-regular points. Every three ϵ-regular points determine a plane, called an ϵ-regular plane.

LEMMA 22. *Let a and ϵ be positive constants. Let P be a set of n points in a 3D grid. There is an ϵ-regular plane such that there are $\le (\frac{3}{4}+\epsilon)n$ points on each side of the plane and $2\pi(a+\epsilon)(\frac{3}{4\pi})^{2/3}n^{2/3}$ points with distance at most a to it.*

Proof. Let L be the plane of Theorem 14. Let H be the area of the intersection between plane L and the six faces of the $m \times m \times m$ cube that contains all the points in P. Let p_1 and p_2 be the two points in H with the maximal distance. Let p_3 be the point in H with the largest perpendicular distance to the line p_1p_2. Let p_1', p_2'

and p'_3 be the δ-regular noncollinear points such that p'_i has distance $\leq \delta$ to p_i for $i = 1, 2, 3$. Use the δ-regular plane determined by p'_1, p'_2, and p'_3 (by selecting δ small enough). $\square$

LEMMA 23. *For some positive constant c_0 and $\epsilon > 0$, there exists an $O(m^{c_0 \log n} n^{-6.9128 n^{2/3}} n_0^{(13.8258-\epsilon) n^{2/3}})$-time algorithm for the 3D multisequence folding problem in an $m \times m \times m$ cube, where n is the sum of the lengths of the input disjoint subsequences of S_0 and n_0 is the length of S_0.*

Proof. Let $a = 1/2$, $c = 3/4 + \delta$, and $d = 2\pi(a + \delta)(\frac{3}{4\pi})^{2/3}$. As in the proof of Lemma 20, let $u_1 = \binom{8}{3}(\frac{m}{\delta})^6$, let $u_2 = (\frac{2}{d})^{dn^{2/3}} \cdot dn^{2/3} \cdot n^{\frac{1}{3} dn^{2/3}}$, and let $u_3 = \beta'(\frac{\alpha'}{d})^{2dn^{\frac{2}{3}}} n^{-\frac{4}{3} dn^{\frac{2}{3}}} n_0^{2dn^{\frac{2}{3}}}$, where α' and β' are similar to the α and β in the proof of Lemma 20. We have $T(n) \leq u_1 \cdot u_2 \cdot u_3 \cdot T(cn)$. This implies that $T(n) = (mn)^{c_0 \log n} 2^{c_0 n^{\frac{2}{3}}} n^{-\frac{d}{(1-c^{2/3})} n^{2/3}} n_0^{\frac{2d}{(1-c^{2/3})} n^{2/3}}$ for some constant $c_0 > 0$. $\square$

THEOREM 24. *There is an $O(n^{6.913 n^{2/3}})$-time algorithm for the 3D protein folding problem in the HP-model.*

Proof. Fix the two middle letters at the two central neighbor positions of an $n \times n \times n$ cube. Let the folding be inside the $n \times n \times n$ cube, and apply Lemma 23. $\square$

6. Conclusions.

We develop an efficient method to obtain an effective separator for a set of grid points. For a set of 2D (3D) grid points, the separator is controlled by a line (plane, resp.) L and a distance a to L. The region of the separator consists of those points with distance at most a to L. The distance parameter a provides us with a flexible way to control the width of the separator region. The separators are used in obtaining a subexponential time algorithm for the protein folding problem in the HP-model. Using the linear structure of the separator, we can find the approximate separator region by checking $O(n^2)$ $(O(n^6))$ possible locations in developing the algorithm for the 2D (3D, resp.) protein folding problem in the HP-model. These algorithms for the protein folding problem have a nontrivial upper bound, but they are not practical enough for implementation. The separators developed in this paper have been found to have more applications in a series of recent papers [11, 7, 13, 12].

Acknowledgments. We are grateful to Mahdi Abdelguerfi, Padmanabhan Mahadevan, and Seth Pincus for helpful discussions during this research. We thank the anonymous referees for helpful comments and for pointing out an error in the earlier version. We are also grateful to Zhixiang Chen and Ming Li for their suggestions for improving the presentation of this paper. The first author would also like to thank Chanda Yadavalli for introducing him to the area of bioinformatics.

REFERENCES

[1] R. Agarwala, S. Batzoglou, V. Dancik, S. Decatur, S. Hannenhalli, M. Farach, M. Muthukrishnan, and S. Skiena, *Local rules for protein folding on a triangular lattice and generalized hydrophobicity in the HP model*, J. Comput. Biol., 21 (1997), pp. 275–296.

[2] N. Alon, P. Seymour, and R. Thomas, *Planar separators*, SIAM J. Discrete Math., 7 (1994), pp. 184–193.

[3] N. Alon, P. Seymour, and R. Thomas, *A separator theorem for graphs with an excluded minor and its applications*, in Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, 1990, pp. 293–299.

[4] R. Backofen, *Constraint techniques for solving the protein structure prediction problem*, in Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming, Lecture Notes in Comput. Sci. 1520, Springer, London, 1998, pp. 72–86.

[5] U. Bastolla, H. Frauenkron, E. Gerstner, P. Grassberger, and W. Nadler, *Testing a new Monte Carlo algorithm for protein folding*, Proteins: Structure, Function, and Genetics, 32 (1998), pp. 52–66.

[6] B. Berger and T. Leighton, *Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete*, J. Comput. Biol., 5 (1998), pp. 27–40.

[7] Z. Chen, B. Fu, Y. Tang, and B. Zhu, *A PTAS for a disc covering problem using width-bounded separators*, J. Comb. Optim., (2006), pp. 203–217.

[8] P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni, and M. Yannakakis, *On the complexity of protein folding*, J. Comput. Biol., 5 (1998), pp. 423–465.

[9] H. N. Djidjev, *On the problem of partitioning planar graphs*, SIAM J. Alg. Disc. Meth., 3 (1982), pp. 229–240.

[10] H. N. Djidjev and S. M. Venkatesan, *Reduced constants for simple cycle graph separation*, Acta Inform., 34 (1997), pp. 231–234.

[11] B. Fu, *Theory and application of width bounded geometric separator*, in Proceedings of the 23rd International Symposium on Theoretical Aspects of Computer Science (STACS'06), Lecture Notes in Comput. Sci. 3884, Springer, Berlin, 2006, pp. 277–288.

[12] B. Fu and Z. Chen, *Sublinear-time algorithms for width-bounded geometric separators and their applications to protein side-chain packing problems*, in Proceedings of the Second International Conference on Algorithmic Aspects in Information and Management, Lecture Notes in Comput. Sci. 4041, Springer, Berlin, pp. 49–160.

[13] B. Fu, S. Oprisan, and L. Xu, *Multi-directional width-bounded geometric separator and protein folding*, in Proceedings of the 16th Annual International Symposium on Algorithms and Computation, Lecture Notes in Comput. Sci. 3827, Springer, Berlin, 2005, pp. 995–1006.

[14] B. Fu and W. Wang, *A* $2^{O(n^{1-1/d}\log n)}$ *time algorithm for d-dimensional protein folding in the HP-model*, in Proceedings of the 31st International Colloquium on Automata, Languages and Programming, Lecture Notes in Comput. Sci. 3142, Springer, Berlin, 2004, pp. 630–644.

[15] H. Gazit, *An Improved Algorithm for Separating a Planar Graph*, manuscript, University of Southern California, Los Angeles, 1986.

[16] J. R. Gilbert, J. P. Hutchinson, and R. E. Tarjan, *A separation theorem for graphs of bounded genus*, J. Algorithms, 5 (1984), pp. 391–407.

[17] W. E. Hart and S. Istrail, *Fast protein folding in the hydrophobic-hydrophilic model within three-eights of optimal*, in Proceedings of the 27th ACM Symposium on the Theory of Computing, 1995, pp. 157–168.

[18] M. Khimasia and P. Coveney, *Protein structure prediction as a hard optimization problem: The genetic algorithm approach*, Molecular Simulation, 19 (1997), pp. 205–226.

[19] N. Krasnogor, D. Pelta, P. E. Martinez Lopez, and E. De La Canal, *Genetic algorithms for the protein folding problem: A critical view*, in Engineering of Intelligent Systems, C. Fyfe and E. Alpaydin, eds., International Computer Science Conventions, ICSC Academic Press, Canada/Switzerland, 1998, pp. 353–360.

[20] K. F. Lau and K. A. Dill, *A lattice statistical mechanics model of the conformational and sequence spaces of proteins*, Macromolecules, 1989, pp. 3986–3997.

[21] K. F. Lau and K. A. Dill, *Theory for protein mutability and biogenesis*, Proc. Natl. Acad. Sci. USA, 87 (1990), pp. 638–642.

[22] F. Liang and W. Wong, *Evolutionary Monte Carlo for protein folding simulations*, J. Chem. Phys., 115 (2001), pp. 3374–3380.

[23] D. Lichtenstein, *Planar formulae and their uses*, SIAM J. Comput., 11 (1982), pp. 329–343.

[24] R. J. Lipton and R. E. Tarjan, *A separator theorem for planar graphs*, SIAM J. Appl. Math., 36 (1979), pp. 177–189.

[25] R. J. Lipton and R. E. Tarjan, *Applications of a planar separator theorem*, SIAM J. Comput., 9 (1980), pp. 615–627.

[26] G. L. Miller, S.-H. Teng, and S. A. Vavasis, *A unified geometric approach to graph separators*, in 32nd Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, CA, 1991, pp. 538–547.

[27] G. L. Miller and W. Thurston, *Separators in two and three dimensions*, in Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, 1990, pp. 300–309.

[28] A. Newman, *A new algorithm for protein folding in the HP model*, in Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, 2002, pp. 876–884.

[29] J. Pach and P. Agarwal, *Combinatorial Geometry*, John Wiley & Sons, New York, 1995.

[30] A. Patton, W.P.III, and E. Goldman, *A standard ga approach to native protein conformation prediction*, in Proceedings of the 6th International Conference on Genetic Algorithms, Morgan Kaufmann, 1995, pp. 574–581.

[31] A. Piccolboni and G. Mauri, *Application of evolutionary algorithms to protein folding prediction*, in Artificial Evolution, Third European Conference 1997, Lecture Notes in Comput. Sci. 1363, Springer, London, pp. 123–136.

[32] S. Plotkin, S. Rao, and W. D. Smith, *Shallow excluded minors and improved graph decompositions*, in Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, Arlington, VA, SIAM, 1994, pp. 462–470.

[33] A. Rabow and H. Scheraga, *Improved genetic algorithm for the protein folding problem by use of a Cartesian combination operator*, Protein Sci., 5 (1996), pp. 1800–1815.

[34] R. Ramakrishnan, B. Ramachandran, and J. Pekney, *A dynamic Monte Carlo algorithm for exploration of dense conformation spaces in heteropolymers*, J. Chem. Phys., 106 (1997), pp. 2418–2425.

[35] S. S. Ravi and H. B. Hunt, III, *Application of the planar separator theorem to computing problems*, Inform. Process. Lett., 25 (1987), pp. 317–322.

[36] A. Sali, E. Shakhnovich, and M. Karplus, *How does a protein fold?*, Nature, 369 (1994), pp. 248–251.

[37] W. D. Smith and N. C. Wormald, *Geometric separator theorems and applications*, in Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science, 1998, pp. 232–243.

[38] D. A. Spielman and S. H. Teng, *Disk packings and planar separators*, in Proceedings of the 12th Annual ACM Symposium on Computational Geometry, 1996, pp. 349–358.

[39] U. Unger and J. Moult, *Genetic algorithm for 3D protein folding simulations*, in Proceedings of the 5th International Conference on Genetic Algorithms, 1993, pp. 581–588.

[40] U. Unger and J. Moult, *Genetic algorithms for protein folding simulations*, J. Mol. Biol., 231 (1993), pp. 75–81.

[41] K. Yue and K. A. Dill, *Sequence-structure relationships in proteins and copolymers*, Phys. Rev. E, 48 (1993), pp. 2267–2278.

SIAM J. COMPUT.
Vol. 37, No. 4, pp. 1030–1045

POPULAR MATCHINGS[*]

DAVID J. ABRAHAM[†], ROBERT W. IRVING[‡], TELIKEPALLI KAVITHA[§], AND KURT MEHLHORN[¶]

Abstract. We consider the problem of matching a set of *applicants* to a set of *posts*, where each applicant has a *preference list*, ranking a nonempty subset of posts in order of preference, possibly involving ties. We say that a matching M is *popular* if there is no matching M' such that the number of applicants preferring M' to M exceeds the number of applicants preferring M to M'. In this paper, we give the first polynomial-time algorithms to determine if an instance admits a popular matching and to find a largest such matching, if one exists. For the special case in which every preference list is strictly ordered (i.e., contains no ties), we give an $O(n+m)$ time algorithm, where n is the total number of applicants and posts and m is the total length of all of the preference lists. For the general case in which preference lists may contain ties, we give an $O(\sqrt{n}m)$ time algorithm.

Key words. matchings, bipartite graphs, one-sided preference lists

AMS subject classification. 68W40

DOI. 10.1137/06067328X

1. Introduction.

An instance of the *popular matching problem* is a bipartite graph $G = (\mathcal{A} \cup \mathcal{P}, E)$ and a partition $E = E_1 \dot{\cup} E_2 \dots \dot{\cup} E_r$ of the edge set. We call the nodes in $\mathcal{A}$ *applicants*, the nodes in $\mathcal{P}$ *posts*, and the edges in E_i the edges of rank i. If $(a,p) \in E_i$ and $(a,p') \in E_j$, with $i < j$, we say that *a prefers p to p'*. If $i = j$, we say that a is *indifferent* between p and p'. This ordering of posts adjacent to a is called a's *preference list*. We say that preference lists are *strictly ordered* if no applicant is indifferent between any two posts on his/her preference list. More generally, if applicants can be indifferent between posts, we say that preference lists contain *ties*.

A *matching* M of G is a set of edges, no two of which share an end point. A node $u \in \mathcal{A} \cup \mathcal{P}$ is either *unmatched* in M or *matched* to some node, denoted by $M(u)$ (i.e., $(u, M(u)) \in M$). We say that an applicant a *prefers* matching M' to M if (i) a is matched in M' and unmatched in M or (ii) a is matched in both M' and M, and a prefers $M'(a)$ to $M(a)$. M' is *more popular than* M, denoted by $M' \succ M$, if the number of applicants that prefer M' to M exceeds the number of applicants that prefer M to M'.

DEFINITION 1.1. *A matching M is* popular *if and only if there is no matching M' that is more popular than M.*

Example 1.1. Figure 1.1 shows the preference lists for an example instance in which $\mathcal{A} = \{a_1, a_2, a_3\}$, $\mathcal{P} = \{p_1, p_2, p_3\}$, and each applicant prefers p_1 to p_2 and p_2

[*] Received by the editors October 25, 2006; accepted for publication (in revised form) June 5, 2007; published electronically October 5, 2007. A preliminary version of this work appeared in *Proceedings of the* 16*th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2005, pp. 424–432.
http://www.siam.org/journals/sicomp/37-4/67328.html

[†] Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213 (dabraham@cs.cmu.edu).

[‡] Department of Computing Science, University of Glasgow, Glasgow G12 8RZ, Scotland, UK (rwi@dcs.gla.ac.uk).

[§] Computer Science and Automation, Indian Institute of Science, Bangalore 560012, India (kavitha@csa.iisc.ernet.in).

[¶] Department of Algorithms and Complexity, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany (mehlhorn@mpi-sb.mpg.de).

a_1	:	p_1	p_2	p_3
a_2	:	p_1	p_2	p_3
a_3	:	p_1	p_2	p_3

FIG. 1.1. *An instance for which there is no popular matching.*

to p_3. Consider the three symmetrical matchings $M_1 = \{(a_1, p_1), (a_2, p_2), (a_3, p_3)\}$, $M_2 = \{(a_1, p_3), (a_2, p_1), (a_3, p_2)\}$, and $M_3 = \{(a_1, p_2), (a_2, p_3), (a_3, p_1)\}$. It is easy to verify that none of these matchings is popular, since $M_1 \prec M_2$, $M_2 \prec M_3$, and $M_3 \prec M_1$. In fact, this instance admits no popular matching, the problem being, of course, that the *more popular than* relation is not acyclic.

The *popular matching problem* is to determine if a given instance admits a popular matching and to find such a matching, if one exists. We remark that popular matchings may have different sizes, and a largest such matching may be smaller than a maximum-cardinality matching. The *maximum-cardinality popular matching problem* then is to determine if a given instance admits a popular matching and to find a *largest* such matching, if one exists.

In this paper, we use a novel characterization of popular matchings to give an $O(\sqrt{n}m)$ time algorithm for the maximum-cardinality popular matching problem, where n is the number of nodes, and m is the number of edges. For instances with strictly ordered preference lists, we give an $O(n + m)$ time algorithm. No polynomial time algorithms were known previously.

Related previous work. The bipartite matching problem with a graded edge set is well-studied in the economics literature; see, for example, [1, 19, 21]. It models some important real-world markets, including the allocation of graduates to training positions [10] and families to government-owned housing [20]. Instances of these markets are restrictions of stable marriage instances [5, 7], in which members of one side of the market (posts) are indifferent between members of the other side of the market (applicants).

The notion of popular matching was originally introduced by Gardenfors [6] in the context of the full stable marriage problem. It is well known that every stable marriage instance admits a weakly stable matching (one for which there is no pair who strictly prefer each other to their partners in the matching). In fact, there can be an exponential number of weakly stable matchings, and so Gardenfors considered the problem of finding one with additional desirable properties, such as popularity. Gardenfors showed that, when preference lists are strictly ordered, every stable matching is popular. He also showed that, when preference lists contain ties, there may be no popular matching.

For the problem setup considered in this paper, various other definitions of optimality have been studied. For example, a matching M is *Pareto optimal* [2, 1, 19] if there is no matching M' such that (i) some applicant prefers M' to M and (ii) no applicant prefers M to M'. In particular, such a matching has the property that no coalition of applicants can collectively improve their allocation (say, by exchanging posts with one another) without requiring some other applicant to be worse off. This is the weakest reasonable definition of optimality—see [2] for an algorithmically oriented exposition. Stronger definitions exist: A matching is *rank-maximal* [11] if it allocates the maximum number of applicants to their first choice and then, subject to this, the maximum number to their second choice, and so on. Rank-maximal matchings always exist and may be found in time $O(\min(n, C\sqrt{n})m)$ [11], where C is the maximum edge rank used in the matching. Finally, we mention *maximum-utility* matchings, which maximize $\sum_{(a,p)\in M} u_{a,p}$, where $u_{a,p}$ is the utility of allocating post p to applicant a. Maximum-utility matchings can be found through an obvious transformation to the maximum-weight matching problem. Neither rank-maximal nor maximum-utility matchings are necessarily popular.

Preliminaries. For exposition purposes, we create a unique *last resort* post $l(a)$ for each applicant a and assign the edge $(a, l(a))$ higher rank than any edge incident on a. In this way, we can assume that every applicant is matched, since any unmatched applicant can be allocated to his/her last resort. From now on then, matchings are *applicant-complete*, and the size of a matching is just the number of applicants not matched to their last resort. We may also assume that instances have no gaps—so if an applicant a is incident to a rank i edge, then a is also incident to edges of all ranks smaller than i.

Organization of the paper. In section 2 we develop an alternative characterization of popular matchings, under the assumption that preference lists are strictly ordered. We then use this characterization as the basis of a linear-time algorithm to solve the maximum-cardinality popular matching problem. In section 3 we consider preference lists with ties and give an $O(\sqrt{n}m)$ time algorithm for the maximum-cardinality popular matching prblem. In section 4 we give some empirical results on the probability that a popular matching exists. Finally, the preliminary version of this paper motivated the study of several other questions related to popular matchings. We end by summarizing this recent work.

2. Strictly ordered preference lists.

In this section, we restrict our attention to strictly ordered preference lists, both to provide some intuition for the more general case and because we can solve the popular matching problem in only linear time. This last claim is not immediately clear, since Definition 1.1 potentially requires an exponential number of comparisons to even check that a given matching is popular. We begin this section then by developing an equivalent (though efficiently checkable) characterization of popular matchings.

2.1. Characterizing popular matchings.

For each applicant a, let $f(a)$ denote the first-ranked post on a's preference list (i.e., $(a, f(a)) \in E_1$). We call any such post p an *f-post* and denote by $f(p)$ the set of applicants a for which $f(a) = p$.

Example 2.1. Figure 2.1 gives the preference lists for an instance with six applicants and six posts that we shall use to illustrate the results in this section. The f-posts for this instance are p_1, p_2, and p_3, and $f(p_1) = \{a_1, a_2\}$, $f(p_2) = \{a_3, a_4, a_5\}$, and $f(p_3) = \{a_6\}$. Note that we use l_i as an abbreviation for $l(a_i)$.

The following lemma gives the first of three conditions necessarily satisfied by a popular matching.

LEMMA 2.1. *Let M be any popular matching. Then for every f-post p,* (i) *p is matched in M, and* (ii) *$M(p) \in f(p)$.*

a_1	:	p_1	p_2	p_3	l_1
a_2	:	p_1	p_5	p_4	l_2
a_3	:	p_2	p_1	p_3	l_3
a_4	:	p_2	p_3	p_6	l_4
a_5	:	p_2	p_6	p_4	l_5
a_6	:	p_3	p_2	p_5	l_6

FIG. 2.1. *An illustrative example.*

a_1	:	$\mathbf{p_1}$	p_2	p_3	$\underline{l_1}$
a_2	:	$\mathbf{p_1}$	$\underline{p_5}$	p_4	l_2
a_3	:	$\mathbf{p_2}$	p_1	p_3	$\underline{l_3}$
a_4	:	$\mathbf{p_2}$	p_3	$\underline{p_6}$	l_4
a_5	:	$\mathbf{p_2}$	$\underline{p_6}$	p_4	l_5
a_6	:	$\mathbf{p_3}$	p_2	$\underline{p_5}$	l_6

FIG. 2.2. *The f-posts and s-posts for the example instance.*

Proof. Every f-post p must be matched in M, for otherwise we can promote any $a \in f(p)$ to p, thereby constructing a matching more popular than M. Suppose for a contradiction then that p is matched to some $M(p) \notin f(p)$. Select any $a_1 \in f(p)$, let $a_2 = M(p)$, and since all f-posts are matched in M, let $a_3 = M(f(a_2))$. We can again construct a matching more popular than M, this time by (i) demoting a_3 to $l(a_3)$, (ii) promoting a_2 to $f(a_2)$, and then (iii) promoting a_1 to p. □

Example 2.2. According to Lemma 2.1, we can be sure that, if a popular matching exists for our example instance, then posts p_1, p_2, and p_3 are matched, and $M(p_1) \in \{a_1, a_2\}$, $M(p_2) \in \{a_3, a_4, a_5\}$, and $M(p_3) = a_6$.

For each applicant a, let $s(a)$ denote the first non-f-post on a's preference list (note that $s(a)$ must exist, due to the introduction of $l(a)$). We call any such post p an *s-post* and remark that f-posts are disjoint from s-posts.

Example 2.3. Figure 2.2 shows the preference lists for our example instance with the f-posts and s-posts highlighted. The bold entry in each preference list is the f-post and the underlined entry is the s-post.

In the next two lemmas, we show that a popular matching can only allocate an applicant a to either $f(a)$ or $s(a)$.

LEMMA 2.2. *Let M be any popular matching. Then for every applicant a, $M(a)$ can never be strictly between $f(a)$ and $s(a)$ on a's preference list.*

Proof. Suppose for a contradiction that $M(a)$ is strictly between $f(a)$ and $s(a)$. Since a prefers $M(a)$ to $s(a)$, we have that $M(a)$ is an f-post. Furthermore, M is a popular matching, so a belongs to $f(M(a))$ (by Lemma 2.1), thereby contradicting the assumption that a prefers $f(a)$ to $M(a)$. □

LEMMA 2.3. *Let M be a popular matching. Then for every applicant a, $M(a)$ is never worse than $s(a)$ on a's preference list.*

Proof. Suppose for a contradiction that a_1 prefers $s(a_1)$ to $M(a_1)$. If $s(a_1)$ is unmatched in M, we can promote a_1 to $s(a_1)$, thereby constructing a matching more popular than M. Otherwise, let $a_2 = M(s(a_1))$, and let $a_3 = M(f(a_2))$ (note that $a_2 \neq a_3$, since f-posts and s-posts are disjoint). We can again construct a matching more popular than M, this time by (i) demoting a_3 to $l(a_3)$, (ii) promoting a_2 to $f(a_2)$, and then (iii) promoting a_1 to $s(a_1)$. □

The three necessary conditions we have just derived form the basis of the following preliminary characterization.

LEMMA 2.4. *A matching M is popular if and only if*

(i) *every f-post is matched in M, and*

(ii) *for each applicant a, $M(a) \in \{f(a), s(a)\}$.*

Proof. Any popular matching necessarily satisfies conditions (i) and (ii) (by Lemmas 2.1–2.3). It remains to show that, together, these conditions are sufficient.

Let M be any matching satisfying (i) and (ii), and suppose for a contradiction that there is some matching M' that is more popular than M. Let a be any applicant that prefers M' to M, and let $p' = M'(a)$ (note that p' is distinct for each such a). Now, since a prefers p' to $M(a)$, it follows from condition (ii) that $M(a) = s(a)$. So,

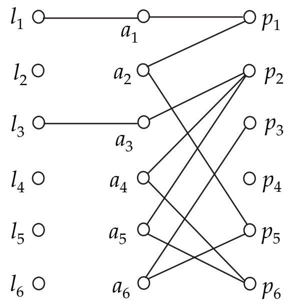


FIG. 2.3. *The reduced graph G' for the example instance.*

p' is an f-post, which by condition (i) must be matched in M, say, to a'. But then $p' = f(a')$ (by condition (ii) and since f-posts and s-posts are disjoint), and so a' prefers M to M'.

Therefore, for every applicant a that prefers M' to M, there is a distinct corresponding applicant a' that prefers M to M'. Hence, M' is not more popular than M, giving the required contradiction. □

Given an instance graph $G = (\mathcal{A} \cup \mathcal{P}, E)$, we define the *reduced graph* $G' = (\mathcal{A} \cup \mathcal{P}, E')$ as the subgraph of G containing two edges for each applicant a: one to $f(a)$ and the other to $s(a)$. We remark that G' need not admit an applicant-complete matching, since $l(a)$ is now isolated whenever $s(a) \neq l(a)$.

Example 2.4. Figure 2.3 shows the reduced graph for our example instance.

Lemma 2.4 gives us that M is a popular matching of G if and only if every f-post is matched in M, and M belongs to the graph G'. Recall that all popular matchings are applicant-complete through the introduction of last resorts. Hence, the following characterization is immediate.

THEOREM 2.5. *M is a popular matching of G if and only if*

(i) *every f-post is matched in M, and*

(ii) *M is an applicant-complete matching of the reduced graph G'.*

Example 2.5. By applying Theorem 2.5 to the reduced graph of Figure 2.3, it may be verified that our example instance admits four popular matchings, two of size 5 and two of size 4, as listed below. (Clearly, in the matchings of size 5, a_3 is matched with his last resort in the reduced graph, and in those of size 4, a_1 is also matched with his last resort.)

$$\begin{aligned}
M_1 &= \{(a_1,p_1),(a_2,p_5),(a_4,p_2),(a_5,p_6),(a_6,p_3)\},\\
M_2 &= \{(a_1,p_1),(a_2,p_5),(a_4,p_6),(a_5,p_2),(a_6,p_3)\},\\
M_3 &= \{(a_2,p_1),(a_4,p_2),(a_5,p_6),(a_6,p_3)\},\\
M_4 &= \{(a_2,p_1),(a_4,p_6),(a_5,p_2),(a_6,p_3)\}.
\end{aligned}$$

2.2. Algorithmic results. Figure 2.4 contains an algorithm for solving the popular matching problem. The correctness of this algorithm follows immediately from the characterization in Theorem 2.5. We remark only that at the termination of the loop, every f-post must be matched, since $f(a)$ is unique for each applicant a, and f-posts are disjoint from s-posts. We now show a linear-time implementation of this algorithm.

It is clear that the reduced graph G' of G can be constructed in $O(n+m)$ time. G' has $O(n)$ edges, since each applicant has degree 2, and so it is also clear that the

```
**Popular matching** $(G = (\mathcal{A} \cup \mathcal{P}, E))$
    $G'$ := reduced graph of $G$;
    **if** $G'$ admits an applicant-complete matching $M$, **then**
        **for each** $f$-post $p$ unmatched in $M$
            let $a$ be any applicant in $f(p)$;
            promote $a$ to $p$ in $M$;
        **return** $M$;
    **else**
        **return** *"no popular matching"*.
```

FIG. 2.4. *Linear-time popular matching algorithm for instances with strictly ordered preference lists.*

```
**Applicant-complete matching** $(G' = (\mathcal{A} \cup \mathcal{P}, E'))$
    $M := \emptyset$;
    **while** some post $p$ has degree 1
        $a$ := unique applicant adjacent to $p$;
        $M := M \cup \{(a, p)\}$;
        $G' := G' - \{a, p\}$;    // *remove a and p from G'*
    **while** some post $p$ has degree 0
        $G' := G' - \{p\}$;
    // *Every post now has degree at least* 2
    // *Every applicant still has degree* 2
    **if** $|\mathcal{P}| < |\mathcal{A}|$ **then**
        **return** *"no applicant-complete matching"*;
    **else**
        // *G' decomposes into a family of disjoint cycles*
        $M'$ := any maximum-cardinality matching of $G'$;
        **return** $M \cup M'$.
```

FIG. 2.5. *Linear-time algorithm for finding an applicant-complete matching in G'.*

loop phase requires only $O(n)$ time. It remains to show how we can efficiently find an applicant-complete matching of G' or determine that no such matching exists.

One approach involves computing a maximum-cardinality matching M of G' and then testing if M is applicant-complete. However, using the Hopcroft–Karp algorithm for maximum-cardinality matching [9], this would take $O(n^{3/2})$ time, which is superlinear, whenever m is $o(n^{3/2})$. Consider then the algorithm in Figure 2.5.

This algorithm begins by repeatedly matching a degree 1 post p with the unique applicant a adjacent to p. No subsequent augmenting path can include p (since it is matched and has degree 1), so we can remove both a and p from consideration. It is not hard to see that this loop can be implemented to run in $O(n)$ time, using, for example, degree counters and lazy deletion. After this, we remove any degree 0 posts, so that all remaining posts have degree at least 2, while all remaining applicants still have degree exactly 2. Now, if $|\mathcal{P}| < |\mathcal{A}|$, G' cannot admit an applicant-complete matching by Hall's marriage theorem [8]. Otherwise, we have that $|\mathcal{P}| \geq |\mathcal{A}|$, and $2|\mathcal{P}| \leq \sum_{p \in \mathcal{P}} \deg(p) = 2|\mathcal{A}|$. Hence, it must be the case that $|\mathcal{A}| = |\mathcal{P}|$, and every post has degree exactly 2. G' therefore decomposes into a family of disjoint cycles, and we need only to walk over these cycles, choosing every second edge.

We summarize the preceding discussion in the following lemma.

LEMMA 2.6. *We can find a popular matching, or determine that no such matching exists, in $O(n + m)$ time.*

We now consider the maximum-cardinality popular matching problem. Let $\mathcal{A}_1$ be the set of all applicants a with $s(a) = l(a)$, and let $\mathcal{A}_2 = \mathcal{A} - \mathcal{A}_1$. Our target matching must satisfy conditions (i) and (ii) of Theorem 2.5 and, among all such matchings, allocate the fewest $\mathcal{A}_1$-applicants to their last resort.

We begin by constructing G' and testing for the existence of an applicant-complete matching M of $\mathcal{A}_2$-applicants to posts (using the applicant-complete matching algorithm in Figure 2.5). If no such M exists, then G admits no popular matching by Theorem 2.5. Otherwise, we remove all edges from G' that are incident on a last resort post and exhaustively augment M, each time matching an additional $\mathcal{A}_1$-applicant with his/her first-ranked post. If any $\mathcal{A}_1$-applicants are unmatched at this point, we simply allocate them to their last resort. Finally, we ensure that every f-post is matched, as in the popular matching algorithm in Figure 2.4. It is clear that the resulting matching is a maximum-cardinality popular matching, and so we comment only on the time complexity of augmenting M.

Note that an alternating path Q from an unmatched applicant a is completely determined (since applicants have degree 2). If we are able to augment along this path, then no subsequent augmenting path can contain a node in Q, since such a path would necessarily terminate at a, which is already matched. Otherwise, if there is no augmenting path from a, then it is not hard to see that again no subsequent augmenting path can contain a node in Q. This means we only need to visit and mark each node at most once, leading to the following result.

Theorem 2.7. *For instances with strictly ordered preference lists, we can find a maximum-cardinality popular matching, or determine that no such matching exists, in $O(n + m)$ time.*

3. Preference lists with ties.

In this section, we relax our assumption that preference lists are strictly ordered and consider problem instances with ties. We begin by developing a generalization of the popular matching characterization, similar to Theorem 2.5. Using this characterization, we then go on to give an $O(\sqrt{n}m)$ time algorithm for solving the maximum-cardinality popular matching problem. Note that we cannot hope for a linear-time algorithm here, since, for the special case where all edges have rank one, the problem of finding a popular matching is equivalent to the problem of finding a maximum-cardinality bipartite matching. Thus the popular matching problem is at least as hard as the maximum-cardinality bipartite matching problem when preference lists contain ties.

3.1. Characterizing popular matchings.

For each applicant a, let $f(a)$ denote the *set* of first-ranked posts on a's preference list. Again, we refer to all such posts p as *f-posts* and denote by $f(p)$ the set of applicants a for which $p \in f(a)$.

It may no longer be possible to match *every* f-post p with an applicant in $f(p)$ (as in Lemma 2.1), since, for example, there may now be more f-posts than applicants. Below then, we work towards generalizing this key lemma.

Let M be a popular matching of some instance graph $G = (\mathcal{A} \cup \mathcal{P}, E)$. We define the *first-choice graph* of G as $G_1 = (\mathcal{A} \cup \mathcal{P}, E_1)$, where E_1 is the set of all rank-one edges.

Example 3.1. Figure 3.1 gives an example instance that we use as an illustration in this section. Ties in the preference lists are indicated by parentheses.

The graph G_1 for this instance is shown in Figure 3.2.

For instances with strictly ordered preference lists, Lemma 2.1 is equivalent to requiring that every f-post is matched in $M \cap E_1$ (note that f-posts are the only posts with nonzero degree in G_1). But since applicants have a unique first choice in

a_1	:	$(p_1$	$p_2)$	p_4	l_1
a_2	:	p_1	$(p_2$	$p_5)$	l_2
a_3	:	p_2	$(p_4$	$p_6)$	l_3
a_4	:	p_2	p_1	p_3	l_4
a_5	:	p_4	p_3	p_2	l_5
a_6	:	$(p_5$	$p_6)$	p_1	l_6

FIG. 3.1. *An example with ties in the preference lists.*

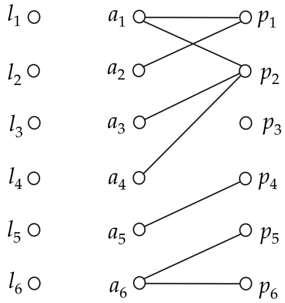


FIG. 3.2. *The graph G_1 for the example instance with ties.*

this context, Lemma 2.1 is also equivalent to the weaker condition that $M \cap E_1$ is a maximum matching of G_1. The next lemma shows that this weaker condition must also be satisfied when ties are permitted.

LEMMA 3.1. *Let M be a popular matching. Then $M \cap E_1$ is a maximum matching of G_1.*

Proof. Suppose for a contradiction that $M_1 = M \cap E_1$ is not a maximum matching of G_1. Then M_1 admits an augmenting path $Q = \langle a_1, p_1, \ldots, p_k \rangle$ with respect to G_1. It follows that $M(a_1) \notin f(a_1)$, and either p_k is unmatched in M, or $M(p_k) \notin f(p_k)$. We now show how to use Q to construct a matching M' that is more popular than M. Begin with $M' = M \setminus \{(a_1, M(a_1))\}$. There are two cases:

(i) *p_k is unmatched in M'.* Since both a_1 and p_k are unmatched in M', we augment M' with Q. In this new matching, a_1 is matched with p_1 (where $p_1 \in f(a_1)$), while all other applicants in Q remain matched to one of their first-ranked posts. Hence M' is more popular than M.

(ii) *p_k is matched in M'.* Let $a_{k+1} = M'(p_k)$, and note that $p_k \notin f(a_{k+1})$. Remove (a_{k+1}, p_k) from M', and then augment M' with Q. Select any $p_{k+1} \in f(a_{k+1})$. If p_{k+1} is unmatched in M', we promote a_{k+1} to p_{k+1}. Otherwise, we demote $a = M'(p_{k+1})$ to either $l(a)$ (if $a \neq a_1$) or back to $M(a_1)$ (if $a = a_1$), after which we can promote a_{k+1} to p_{k+1}. It is clear from this that at least one of a_1 and a_{k+1} prefers M' to M. Also, at most one applicant (that is, a) prefers M to M', though in this case *both* a_1 and a_{k+1} prefer M'. Hence, M' is more popular than M. □

Example 3.2. In our example, we see from Figure 3.2 and Lemma 3.1 that posts p_1, p_2, and p_4 and applicants a_5 and a_6 must be matched in any popular matching M. Furthermore, we deduce that $M(p_1) \in \{a_1, a_2\}$, $M(p_2) \in \{a_1, a_3, a_4\}$, $M(p_4) = a_5$, and $M(a_6) \in \{p_5, p_6\}$.

We now begin working towards a generalized definition of $s(a)$. For instances with strictly ordered preference lists, $s(a)$ is equivalent to the first post in a's preference

a_1	:	$(\mathbf{p_1}$	$\mathbf{p_2})$	p_4	$\underline{l_1}$
a_2	:	$\mathbf{p_1}$	$(p_2$	$\underline{p_5})$	l_2
a_3	:	$\mathbf{p_2}$	$(p_4$	$\underline{p_6})$	l_3
a_4	:	$\mathbf{p_2}$	p_1	$\underline{p_3}$	l_4
a_5	:	$\mathbf{p_4}$	$\underline{p_3}$	p_2	l_5
a_6	:	$(\underline{\mathbf{p_5}}$	$\underline{\mathbf{p_6}})$	p_1	l_6

FIG. 3.3. *An example with ties in the preference lists.*

list that has degree 0 in G_1. However, since Lemma 2.1 no longer holds, $s(a)$ may now contain any number of surplus f-posts. It will help us to know which f-posts *cannot* be included in $s(a)$, and for this we use the following well known ideas from bipartite matching theory.

Let M_1 be a maximum matching of some bipartite graph $G_1 = (\mathcal{A} \cup \mathcal{P}, E_1)$. (Note that we are using notation that matches our use of this theory—so $M_1 = M \cap E_1$, and G_1 is the graph G restricted to rank-one edges.) Using M_1, we can partition $A \cup \mathcal{P}$ into three disjoint sets: A node v is *even* (respectively, *odd*) if there is an even (respectively, odd) length alternating path (with respect to M_1) from an unmatched node to v. Similarly, a node v is *unreachable* if there is no alternating path from an unmatched node to v. Denote by $\mathcal{E}$, $\mathcal{O}$, and $\mathcal{U}$ the sets of even, odd, and unreachable nodes, respectively. The Gallai–Edmonds decomposition lemma, covered in detail in [13], gives some fundamental relationships between maximum matchings and this type of node partition.

LEMMA 3.2 (Gallai–Edmonds decomposition). *Let $\mathcal{E}$, $\mathcal{O}$, and $\mathcal{U}$ be the node sets defined by G_1 and M_1 above. Then*

(a) *$\mathcal{E}$, $\mathcal{O}$, and $\mathcal{U}$ are pairwise disjoint. Every maximum matching in G_1 partitions the node set into the same partition of even, odd, and unreachable nodes.*

(b) *In any maximum-cardinality matching of G_1, every node in $\mathcal{O}$ is matched with some node in $\mathcal{E}$, and every node in $\mathcal{U}$ is matched with another node in $\mathcal{U}$. The size of a maximum-cardinality matching is $|\mathcal{O}| + |\mathcal{U}|/2$.*

(c) *No maximum-cardinality matching of G_1 contains an edge between two nodes in $\mathcal{O}$ or a node in $\mathcal{O}$ and a node in $\mathcal{U}$. Also, there is no edge in G_1 connecting a node in $\mathcal{E}$ with a node in $\mathcal{U}$.*

Example 3.3. In our example, it may be verified from a maximum matching, say, $\{(a_1, p_2), (a_2, p_1), (a_5, p_4), (a_6, p_5)\}$, in Figure 3.2, that $\mathcal{E} = \{a_1, a_2, a_3, a_4, p_3, p_5, p_6, l_1, l_2, l_3, l_4, l_5, l_6\}$, $\mathcal{O} = \{a_6, p_1, p_2\}$, and $\mathcal{U} = \{a_5, p_4\}$.

Now, since M_1 is a maximum-cardinality matching of G_1, Lemma 3.2(b) gives us that every odd or unreachable post p in G_1 must be matched in M to some applicant in $f(p)$. Such posts cannot be members of $s(a)$, and so we define $s(a)$ to be the set of top-ranked posts in a's preference list that are *even* in G_1 (note that $s(a) \neq \emptyset$, since $l(a)$ is always even in G_1). This definition coincides with the one in section 2, since degree 0 posts are even, and whenever every applicant has a unique first choice, posts with nonzero degree (i.e., f-posts) are odd or unreachable.

Example 3.4. Figure 3.3 displays the preference lists for our example instance, annotated as before, with the f-posts in bold and the s-posts underlined. Note that, when ties are present, f-posts and s-posts may coincide, as occurs here for applicant a_6.

Recall that our original definition of $s(a)$ led to Lemmas 2.2 and 2.3, which restrict the set of posts to which an applicant can be matched in a popular matching. We now show that the generalized definition leads to analogous results here.

LEMMA 3.3. *Let M be a popular matching. Then for every applicant a, $M(a)$ can never be strictly between $f(a)$ and $s(a)$ on a's preference list.*

Proof. Suppose for a contradiction that $M(a)$ is strictly between $f(a)$ and $s(a)$. Then since a prefers $M(a)$ to any post in $s(a)$ and because posts in $s(a)$ are the top-ranked even nodes in G_1, it follows that $M(a)$ must be an odd or unreachable node of G_1. By Lemma 3.2(b), odd and unreachable nodes are matched in every maximum matching of G_1. But since $M(a) \notin f(a)$, $M(a)$ is unmatched in $M \cap E_1$. Hence M is not a maximum matching on rank-one edges, and so by Lemma 3.1, M is not a popular matching. □

LEMMA 3.4. *Let M be a popular matching. Then for every applicant a, $M(a)$ is never worse than $s(a)$ on a's preference list.*

Proof. Suppose for a contradiction that $M(a_1)$ is strictly worse than $s(a_1)$. Let p_1 be any post in $s(a_1)$. If p_1 is unmatched in M, we can promote a_1 to p_1, thereby constructing a matching more popular than M. Otherwise, let $a_2 = M(p_1)$. There are two cases:

(a) $p_1 \notin f(a_2)$. Select any post $p_2 \in f(a_2)$, and let $a_3 = M(p_2)$ (note that p_2 must be matched in M, for otherwise Lemma 3.1 is contradicted). We can again construct a matching more popular than M, this time by (i) demoting a_3 to $l(a_3)$, (ii) promoting a_2 to p_2, and then (iii) promoting a_1 to p_1.

(b) $p_1 \in f(a_2)$. Now, since $p_1 \in s(a_1)$ as well, it must be the case that p_1 is an even post in G_1. It follows then that G_1 contains (with respect to $M \cap E_1$) an even length alternating path $Q' = \langle p_1, a_2, \ldots, p_k \rangle$, where p_k is unmatched in $M \cap E_1$ (note that p_k may be matched in M though). Now, let $Q = \langle a_1, p_1, a_2, \ldots, p_k \rangle$ (i.e., a_1 followed by Q'), and let $M' = M \setminus \{(a_1, M(a_1)\}$. The remaining argument follows the proof of Lemma 3.1. If p_k is unmatched in M', $M' \oplus Q$ is more popular than M. Otherwise, p_k is matched in M'. Let $a_{k+1} = M'(p_k)$, and note that $p_k \notin f(a_{k+1})$. Remove (a_{k+1}, p_k) from M', and then augment M' with Q. Select any $p_{k+1} \in f(a_{k+1})$. If p_{k+1} is unmatched in M', we promote a_{k+1} to p_{k+1}. Otherwise, we demote $a = M'(p_{k+1})$ to either $l(a)$ (if $a \neq a_1$) or back to $M(a_1)$ (if $a = a_1$), after which we can promote a_{k+1} to p_{k+1}. It is clear from this that at least one of a_1 and a_{k+1} prefers M' to M. Also, at most one applicant (that is, a) prefers M to M', though in this case *both* a_1 and a_{k+1} prefer M'. Hence, M' is more popular than M. □

The three necessary conditions we have just derived form the basis of the following preliminary characterization.

LEMMA 3.5. *A matching M is popular in G if and only if*

(i) *$M \cap E_1$ is a maximum matching of G_1, and*

(ii) *for each applicant a, $M(a) \in f(a) \cup s(a)$.*

Proof. Any popular matching necessarily satisfies conditions (i) and (ii) (by Lemmas 3.1, 3.3, and 3.4). It remains to show that, together, these conditions are sufficient.

Let M be any matching satisfying conditions (i) and (ii), and suppose for a contradiction that there is some matching M' that is more popular than M. Let a be any applicant that prefers M' to M. We want to show that there is a distinct corresponding applicant a' that prefers M to M'.

The graph $H = (M \oplus M') \cap E_1$ consists of disjoint cycles and paths, each alternating between edges in $M \cap E_1$ and edges in $M' \cap E_1$. We claim that $M'(a)$ must be contained in a *nonempty path* Q of H. First, note that $M'(a)$ is an odd or unreachable node in G_1, since a prefers $M'(a)$ to $M(a)$, and $M(a) \in s(a)$ is a top-ranked even

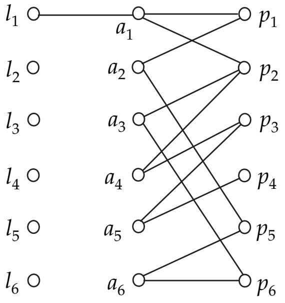


FIG. 3.4. *The reduced graph G' for the example instance with ties.*

node of G_1 in a's preference list. So by condition (i) and Lemma 3.2(b), $M'(a)$ is matched in $M \cap E_1$. However, $M'(a) \neq M(a)$, so $M'(a)$ is not isolated in H. Also, $M'(a)$ cannot be in a cycle, since a is unmatched in $M \cap E_1$. Therefore, $M'(a)$ belongs to some nonempty path Q of H.

Now, one end point of Q must be a (if $M'(a) \in f(a)$) or $M'(a)$ (otherwise). So for each such applicant a, there is a distinct nonempty path Q. Since $M'(a)$ is odd or unreachable, every post p in Q is also odd or unreachable. It follows from Lemma 3.1 that all such posts must be matched in $M \cap E_1$, and so the other end point of Q is an applicant, say, a'. It is easy to see then that a' prefers M to M', since $M(a') \in f(a')$, while $M'(a) \notin f(a')$.

Therefore, for every applicant a that prefers M' to M, there is a distinct corresponding applicant a' that prefers M to M'. Hence, M' is not more popular than M, giving the required contradiction. $\square$

Given an instance graph $G = (\mathcal{A} \cup \mathcal{P}, E)$, we define the *reduced graph* $G' = (\mathcal{A} \cup \mathcal{P}, E')$ as the subgraph of G containing edges from each applicant a to posts in $f(a) \cup s(a)$. We remark that G' need not admit an applicant-complete matching, since $l(a)$ is now isolated whenever $s(a) \neq \{l(a)\}$.

Example 3.5. Figure 3.4 shows the reduced graph for our example instance.

Lemma 3.5 gives us that M is a popular matching of G if and only if M is a maximum matching on rank-one edges, and M belongs to the graph G'. Recall that all popular matchings are applicant-complete through the introduction of last resorts. Hence, the following characterization is immediate.

THEOREM 3.6. *M is a popular matching of G if and only if*

(i) *$M \cap E_1$ is a maximum matching of G_1, and*

(ii) *M is an applicant-complete matching of the reduced graph G'.*

Example 3.6. By applying Theorem 3.6 to the reduced graph of Figure 3.4, it may be verified that our example instance admits five popular matchings, two of size 6 and three of size 5, as listed below. (Clearly, in the three matchings of size 5, a_1 is matched with his last resort l_1 in the reduced graph.)

$$M_1 = \{(a_1, p_1), (a_2, p_5), (a_3, p_2), (a_4, p_3), (a_5, p_4), (a_6, p_6)\},$$
$$M_2 = \{(a_1, p_2), (a_2, p_1), (a_3, p_6), (a_4, p_3), (a_5, p_4), (a_6, p_5)\},$$
$$M_3 = \{(a_2, p_1), (a_3, p_2), (a_4, p_3), (a_5, p_4), (a_6, p_5)\},$$
$$M_4 = \{(a_2, p_1), (a_3, p_2), (a_4, p_3), (a_5, p_4), (a_6, p_6)\},$$
$$M_5 = \{(a_2, p_1), (a_3, p_6), (a_4, p_2), (a_5, p_4), (a_6, p_5)\}.$$

```
Popular matching ($G = (\mathcal{A} \cup \mathcal{P}, E)$)
  1. Construct the graph $G' = (\mathcal{A} \cup \mathcal{P}, E')$, where $E' = \{(a, p) \mid p \in f(a) \cup s(a),\ a \in \mathcal{A}\}$.
  2. Compute a maximum matching $M_1$ on rank-one edges; i.e., $M_1$ is a maximum
     matching in $G_1 = (\mathcal{A} \cup \mathcal{P}, E_1)$.
     ($M_1$ is also a matching in $G'$ because $E' \supseteq E_1$)
  3. Delete all edges in $G'$ connecting two nodes in the set $\mathcal{O}$ or a node in $\mathcal{O}$ with a node
     in $\mathcal{U}$, where $\mathcal{O}$ and $\mathcal{U}$ are the sets of odd and unreachable nodes of $G_1 = (\mathcal{A} \cup \mathcal{P}, E_1)$.
     Determine a maximum matching $M$ in the modified graph $G'$ by augmenting $M_1$.
  4. If $M$ is not applicant-complete, then declare that there is no popular matching in
     $G$.
     Else return $M$.
```

FIG. 3.5. *An $O(\sqrt{n}m)$ popular matching algorithm for preference lists with ties.*

3.2. Algorithmic results. In this section, we present the algorithm popular matching (see Figure 3.5) for solving the popular matching problem. This algorithm is based on the characterization given in Theorem 3.6 and is similar to the algorithm for computing a rank-maximal matching [11].

The following lemma is necessary for the correctness of our algorithm.

LEMMA 3.7. *Algorithm popular matching returns a maximum matching M on rank-one edges.*

Proof. Since M is obtained from M_1 by successive augmentations, every node matched by M_1 is also matched by M. Nodes in $\mathcal{O}$ and $\mathcal{U}$ are matched by M_1 (by Lemma 3.2(b)). Hence, nodes in $\mathcal{O}$ and $\mathcal{U}$ are matched in M.

First, we claim that G' has no edges of rank greater than one incident on nodes in $\mathcal{O}$ and nodes in $\mathcal{U} \cap \mathcal{P}$. Let us consider any odd or unreachable node in $\mathcal{P}$. This is never a candidate for $s(a)$, and hence no edge of the type $(a, p), p \in s(a)$, is incident on such a node. For odd nodes that belong to $\mathcal{A}$, it is the case that they have first-ranked posts that are even, and so $s(a) \subseteq f(a)$. This proves our claim.

So the edges that we removed in step 3 are rank-one edges, and these edges cannot be used by any maximum matching of G_1, by Lemma 3.2(c). (So no popular matching of G can use these edges.) Now the only neighbors of nodes in $\mathcal{O}$ are the even nodes of G_1 (call this set $\mathcal{E}$), and similarly, the only neighbors of nodes in $\mathcal{U} \cap \mathcal{P}$ are nodes in $\mathcal{U} \cap \mathcal{A}$ (by our edge deletions in step 3 and Lemma 3.2(c)). This means that M must match all of the nodes in $\mathcal{O}$ with nodes in $\mathcal{E}$ and all of the nodes in $\mathcal{U} \cap \mathcal{P}$ with nodes in $\mathcal{U} \cap \mathcal{A}$.

So M has at least $|\mathcal{O}| + |\mathcal{U} \cap \mathcal{P}| = |\mathcal{O}| + |\mathcal{U}|/2$ edges of rank one. So M is a maximum matching on rank-one edges (by Lemma 3.2(b)). □

Thus the matching returned by the algorithm popular matching is both an applicant-complete matching in G' and a maximum matching on rank-one edges. The correctness of the algorithm now follows from Theorem 3.6.

It is easy to see that the running time of our algorithm is $O(\sqrt{n}m)$: We use the algorithm of Hopcroft and Karp [9] to compute a maximum matching in G_1 and identify the set of edges E' and construct G' in $O(\sqrt{n}m)$ time. We then repeatedly augment M_1 (by the Hopcroft–Karp algorithm) to obtain M. This gives us the following result.

LEMMA 3.8. *We can find a popular matching, or determine that no such matching exists, in $O(\sqrt{n}m)$ time.*

It is now a simple matter to solve the maximum-cardinality popular matching problem. Let us assume that the instance $G = (\mathcal{A} \cup \mathcal{P}, E)$ admits a popular matching. (Otherwise, we are done.) We now want an applicant-complete matching in G' that is a maximum matching on rank-one edges and which maximizes the number of applicants not matched to their last resort.

Let M' be an arbitrary popular matching in G. We know that M' belongs to the graph G'. Remove all edges of the form $(a, l(a))$ from G' (and M'). Let H denote the resulting subgraph of G'. Note that M' is still a maximum matching on rank-one edges, since no rank-one edge has been deleted from M' or G', but M' need not be a maximum matching in the graph H. Determine a maximum matching N in H by augmenting M'. N is a matching in G' that

(i) is a maximum matching on rank-one edges and

(ii) matches the maximum number of non-last-resort posts.

N need not be a popular matching. Determine a maximum matching M in G' by augmenting N. The matching M will be applicant-complete. Since M is obtained from N by successive augmentations, all posts that are matched by N are still matched by M. Hence, it follows that M is a popular matching that maximizes the number of applicants not matched to their last resort.

The following theorem is therefore immediate.

THEOREM 3.9. *We can find a maximum-cardinality popular matching, or determine that no such matching exists, in* $O(\sqrt{n}m)$ *time.*

4. Concluding remarks. In order to obtain an idea of the probability that a popular matching exists, we performed some simulations. The factors that affect this probability are the number of applicants, the number of posts, the lengths of the preference lists, and the number, size, and position of ties in these lists.

To keep this empirical investigation manageable, we restricted our attention to cases where the numbers of applicants and posts are equal, represented by n, and all preference lists have the same length k. We characterized the ties by a single parameter t, the probability that an entry in a preference list is tied with its predecessor.

Tables 4.1 and 4.2 contain the results of simulations carried out on randomly generated instances with $n = 10$ and $n = 100$, respectively. We set t to a sequence of values in the range 0.0–0.8. For $n = 10$ we allowed k to take all possible values $(1, \ldots, 10)$, and for $n = 100$ we investigated the cases $k = 1, \ldots, 10$ and $k = 20, 30, \ldots, 100$. We generated 1000 random instances in each case. In both cases, the table shows the number of instances admitting a popular matching.

TABLE 4.1
Proportion of instances with a popular matching for $n = 10$.

		t				
		0.0	0.2	0.4	0.6	0.8
k	1	1000	1000	1000	1000	1000
	2	986	988	996	997	1000
	3	898	941	962	983	996
	4	759	846	929	979	999
	5	681	811	915	979	998
	6	636	786	888	976	1000
	7	578	737	893	978	1000
	8	565	738	909	985	1000
	9	553	759	906	980	1000
	10	556	725	890	979	1000

Table 4.2
Proportion of instances with a popular matching for $n = 100$.

		t				
		0.0	0.2	0.4	0.6	0.8
k	1	1000	1000	1000	1000	1000
	2	997	1000	999	1000	1000
	3	884	956	985	990	1000
	4	519	807	925	946	974
	5	204	534	806	863	879
	6	64	346	685	782	798
	7	20	192	534	705	721
	8	8	90	436	628	672
	9	3	39	309	578	670
	10	2	28	243	531	675
	20	0	0	53	346	787
	30	0	0	37	302	776
	40	0	1	37	314	781
	50	0	0	44	291	791
	60	0	1	49	318	775
	70	0	2	36	304	780
	80	0	1	63	280	801
	90	0	0	38	306	776
	100	0	1	51	302	759

These results, and others not reported in detail here, give rise to the following observations:

- When $t = 0.0$, i.e., there are no ties, the likelihood of a popular matching declines rapidly as k increases and, for large n, is negligible except for very small values of k.
- Not surprisingly, increasing the value of t, and therefore the likely number and length of ties, increases the probability of a popular matching.
- For fixed n and t, increasing k initially reduces the likelihood of a popular matching, but beyond a certain range this effect all but disappears.

Thus popular matchings do exist with good probability when the chance of ties in the preference lists is high, which is likely to happen in real-world problems.

In fact, since the preliminary version of this paper [3] appeared, Mahdian [14] has shown that a popular matching exists with high probability, when (i) preference lists are randomly constructed and (ii) the number of posts is a small multiplicative factor larger than the number of applicants.

Of course, for a given instance, it still may be the case that a popular matching does not exist. Recently, McCutchen [16] considered the problem of finding a *least-unpopular* matching, where the unpopularity of a matching M is defined as the maximum ratio over all matchings M' of the number of applicants preferring M' to M to the number of applicants preferring M to M'. This definition of unpopularity makes the problem NP-hard; however, it is not clear if this is the case for other reasonable definitions.

The preliminary version also motivated the study of several other questions related to popular matchings. Manlove and Sng [15] have generalized the algorithms of sections 2.2 and 3.2 to the case where each post has an associated *capacity*, indicating the number of applicants that it can accommodate. (They described this in the equivalent context of the house allocation problem.) They gave an $O(\sqrt{C}n_1 + m)$ time algorithm for the no-ties case and a $O((\sqrt{C} + n_1)m)$ time algorithm when ties are allowed, where n_1 is the number of applicants, m, as usual, is the total length of all preference lists, and C is the total capacity of all of the posts.

In [17] Mestre designed an efficient algorithm for the *weighted* popular matching problem, where each applicant is assigned a priority or weight, and the definition of popularity takes into account the priorities of the applicants. In this case the algorithm for the no-ties version has $O(n + m)$ complexity, and for the version that allows ties, the complexity is $O(\min(k\sqrt{n}, n)m)$, where k is the number of distinct weights assigned to applicants.

In [12], Kavitha and Shah give faster randomized algorithms for the popular matching problem (for problem instances where preference lists contain ties) and a weighted version of the rank-maximal matching problem. Their popular matching algorithm runs in expected time $O(n^\omega)$, where $\omega < 2.376$ is the best exponent for matrix multiplication—this algorithm reduces the popular matching problem to the bipartite perfect matching problem and uses the $O(n^\omega)$ algorithm for the latter problem [18]. The reduction works as follows: In the graph G' (the reduced graph, refer to section 3.2), let us first delete posts which are isolated; now each post in G' is either an odd or unreachable post in G_1 or it is a most preferred even post in G_1. Note that these two sets are disjoint. Let there be k_1 posts of the first type and k_2 posts of the second type. Add $k_1 + k_2 - |\mathcal{A}|$ new nodes to $\mathcal{A}$, and make each of these new nodes adjacent to each of the k_2-posts of the second type (that is, most preferred even posts in G_1). It is easy to see that there is a perfect matching in this resulting graph if and only if there is an applicant-complete matching in G' that is a maximum matching on rank-one edges. Thus it follows that the popular matching problem and the bipartite perfect matching problem have equivalent time complexities.

Finally, in the preliminary version of this paper, we described the following open problem. Suppose we have an instance that admits a popular matching, but we already have a nonpopular matching M_0 in place. Since the *more popular than* relation is not transitive, it may be that no *popular* matching is more popular than M_0. We define a *voting path* then as a sequence of matchings $\langle M_0, M_1, \dots, M_k\rangle$ such that M_i is more popular than M_{i-1} for all $1 \leq i \leq k$, where M_k is popular.

Even though the *more popular than* relation is not acyclic, we were able to show that, for every matching M_0, (i) there is a voting path beginning at M_0 and (ii) the shortest such path has length at most 3. The open problem was to give an efficient algorithm for computing a shortest-length voting path from a given matching. Recently, Abraham and Kavitha [4] have shown that there is always such a voting path of length at most 2 and have given a linear-time algorithm to find one.

Acknowledgments. We thank David Manlove for directing us to previous work in the area and commenting on an early draft. We also thank Julian Mestre for correcting our description of Gardenfors' original results on popular matching.

REFERENCES

[1] A. Abdulkadiroğlu and T. Sönmez, *Random serial dictatorship and the core from random endowments in house allocation problems*, Econometrica, 66 (1998), pp. 689–701.

[2] D. J. Abraham, K. Cechlárová, D. F. Manlove, and K. Mehlhorn, *Pareto-optimality in house allocation problems*, in Proceedings of ISAAC 2004: The 15th Annual International Symposium on Algorithms and Computation, Lecture Notes in Comput. Sci. 3341, Springer-Verlag, Berlin, 2004, pp. 3–15.

[3] D. J. Abraham, R. W. Irving, T. Kavitha, and K. Mehlhorn, *Popular matchings*, in Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms, 2005, pp. 424–432.

[4] D. J. Abraham and T. Kavitha, *Dynamic popular matchings and voting paths*, in Proceedings of SWAT 2006: The 10th Scandinavian Workshop on Algorithm Theory, Lecture Notes in Comput. Sci. 4059, Springer-Verlag, Berlin, 2006, pp. 65–76.

[5] D. Gale and L. S. Shapley, *College admissions and the stability of marriage*, Amer. Math. Monthly, 69 (1962), pp. 9–15.

[6] P. Gardenfors, *Match making: Assignments based on bilateral preferences*, Behavioural Sciences, 20 (1975), pp. 166–173.

[7] D. Gusfield and R. W. Irving, *The Stable Marriage Problem: Structure and Algorithms*, MIT Press, Cambridge, MA, 1989.

[8] P. Hall, *On representatives of subsets*, J. London Math. Soc., 10 (1935), pp. 26–30.

[9] J. E. Hopcroft and R. M. Karp, *An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs*, SIAM J. Comput., 2 (1973), pp. 225–231.

[10] A. Hylland and R. Zeckhauser, *The efficient allocation of individuals to positions*, J. Political Economy, 87 (1979), pp. 293–314.

[11] R. W. Irving, T. Kavitha, K. Mehlhorn, D. Michail, and K. Paluch, *Rank-maximal matchings*, ACM Transactions on Algorithms, 2 (2006), pp. 602–610.

[12] T. Kavitha and C. Shah, *Efficient algorithms for weighted rank-maximal matchings and related problems*, in ISAAC '06: The 17th International Symposium on Algorithms and Computation, 2006, to appear.

[13] L. Lovász and M. D. Plummer, *Matching Theory*, Ann. Discrete Math. 29, North-Holland, Amsterdam, 1986.

[14] M. Mahdian, *Random popular matchings*, in Proceedings of the 7th ACM Conference on Electronic-Commerce, 2006, pp. 238–242.

[15] D. F. Manlove and C. Sng, *Popular matchings in the capacitated house allocation problem*, in Proceedings of ESA 2006, the 14th Annual European Symposium on Algorithms, Lecture Notes in Comput. Sci. 4168, Springer-Verlag, Berlin, 2006, pp. 492–503.

[16] M. McCutchen, *Least-Unpopularity-Factor Matching*, manuscript, 2006.

[17] J. Mestre, *Weighted popular matchings*, in Proceedings of the 33rd International Colloquium on Automata, Languages and Programming, Lecture Notes in Comput. Sci. 4051, Springer-Verlag, Berlin, 2006, pp. 715–726.

[18] M. Mucha and P. Sankowski, *Maximum matchings via Gaussian elimination*, in Proceedings of the 45th Symposium on Foundations of Computer Science, IEEE, Piscataway, NJ, 2004, pp. 248–255.

[19] A. E. Roth and A. Postlewaite, *Weak versus strong domination in a market with indivisible goods*, J. Math. Econom., 4 (1977), pp. 131–137.

[20] Y. Yuan, *Residence exchange wanted: A stable residence exchange problem*, European J. Oper. Res., 90 (1996), pp. 536–546.

[21] L. Zhou, *On a conjecture by Gale about one-sided matching problems*, J. Econom. Theory, 52 (1990), pp. 123–135.

SIAM J. COMPUT.
Vol. 37, No. 4, pp. 1046–1056

A GEOMETRIC APPROACH TO INFORMATION-THEORETIC PRIVATE INFORMATION RETRIEVAL*

DAVID WOODRUFF† AND SERGEY YEKHANIN†

Abstract. A *t-private* private information retrieval (PIR) scheme allows a user to retrieve the ith bit of an n-bit string x replicated among k servers, while any coalition of up to t servers learns no information about i. We present a new geometric approach to PIR and obtain the following: (1) A t-private k-server protocol with communication $O(\frac{k^2}{t} \log k\ n^{1/\lfloor (2k-1)/t \rfloor})$, removing the kt term of previous schemes. This answers an open question of [Y. Ishai and E. Kushilevitz, in *Proceedings of the* 31*st ACM Symposium on Theory of Computing*, 1999, pp. 79–88]. (2) A 2-server protocol with $O(n^{1/3})$ communication, polynomial preprocessing, and online work $O(n/\log^r n)$ for any constant r. This improves the $O(n/\log^2 n)$ work of [A. Beimel, Y. Ishai, and T. Malkin, *J. Cryptology*, 17 (2004), pp. 125–151]. (3) Smaller communication for *instance hiding* [D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway, *J. Cryptology*, 10 (1997), pp. 17–36; Y. Ishai and E. Kushilevitz, in *Proceedings of the* 31*st ACM Symposium on Theory of Computing*, 1999, pp. 79–88], PIR with a polylogarithmic number of servers, and *robust* PIR [A. Beimel and Y. Stahl, in *Proceedings of the* 3*rd Conference on Security in Communications Networks (SCN* 2002*)*, Lecture Notes in Comput. Sci. 2576, Springer, Berlin, 2003, pp. 326–341].

Key words. private information retrieval, preprocessing, partial derivatives

AMS subject classifications. 68P20, 94A60, 94A62

DOI. 10.1137/06065773X

1. Introduction.

Private information retrieval (PIR) was introduced in a seminal paper by Chor et al. [11]. In such a scheme a server holds an n-bit string $x \in \{0,1\}^n$, representing a database, and a user holds an index $i \in [n] = \{1, \ldots, n\}$. At the end of the protocol the user should learn x_i, and the server should learn nothing about i. A trivial solution is for the server to send the user x. While private, the *communication complexity* is linear in n. In contrast, in a nonprivate setting, there is a protocol with only $\log n + 1$ bits of communication. This raises the question of how much communication is really necessary to achieve privacy.

Unfortunately, if information-theoretic privacy is required, then there is no better solution than the trivial one [11]. To get around this, Chor et al. [11] suggested replicating the database among $k > 1$ noncommunicating servers. In this setting, one can do substantially better. Indeed, Chor et al. [11] gave a protocol with complexity $O(n^{1/3})$ for as few as two servers and an $O(k^2 \log k\ n^{1/k})$ solution for the general case. Ambainis [1] then extended the $O(n^{1/3})$ protocol to achieve $O(2^{k^2} n^{1/(2k-1)})$ complexity for every k. Later, Beimel et al. [7] reduced the communication to $2^{\tilde{O}(k)} n^{\frac{2 \log \log k}{k \log k}}$. Finally, in a recent paper Yekhanin [27] presented a 3-server protocol with $O(n^{\epsilon})$ communication for some tiny $\epsilon < 10^{-7}$. Combining the results of [27] and [7], one gets k server protocols with $2^{\tilde{O}(k)} n^{\frac{\delta \log \log k}{k \log k}}$ communication for a small $\delta > 0$ and every k.

*Received by the editors April 20, 2006; accepted for publication (in revised form) April 10, 2007; published electronically October 10, 2007. A preliminary version of this paper appeared in *Proceedings of the* 20*th IEEE Computational Complexity Conference (CCC* '05*)* [26].
http://www.siam.org/journals/sicomp/37-4/65773.html

†Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 32 Vassar Street, Cambridge, MA 02139 (dpwood@mit.edu, yekhanin@mit.edu). The first author's research was supported by an NDSEG fellowship. The second author's research was supported by NTT award MIT 2001-04 and NSF grant CCR 0219218.

Yekhanin [27] has also shown that under a certain plausible number-theoretic conjecture one can achieve $n^{o(1)}$ communication complexity with as few as three servers. The best lower bound is $c \log n$ for a constant $c > 1$ [18, 24, 25]. Stronger lower bounds are known in certain restricted models [5, 16, 14, 24, 20]. For a survey, see [13, 23].

A drawback of all of these solutions is that if any two servers communicate, they can completely recover i. This motivates the notion of a *privacy threshold* t, $1 \le t \le k$, which limits the number of servers that might collude in order to get any information about i. That is, the joint view of any t servers should be independent of i. The case $t > 1$ was addressed in [11, 15, 6]. It is not known whether the techniques of [7] and [27] (that yield the best upper bounds in the 1-private case) can be extended to obtain t-private protocols for arbitrary values of k and $1 \le t \le k$. The best t-private protocols prior to our work were due to Beimel and Ishai [6]. They achieve communication complexity of: $O\big(\binom{k}{t} \frac{k^2}{t} n^{1/\lfloor (2k-1)/t \rfloor}\big)$. Since this bound grows rapidly with t, in [15] the following is asked:

Can one avoid the $\binom{k}{t}$ overhead induced by our use of replication-based secret sharing?

We give a scheme with communication $O\big(\frac{k^2}{t} \log k \ n^{1/\lfloor (2k-1)/t \rfloor}\big)$ for any t and thus answer this question in the affirmative.

Our upper bound is of considerable interest in the *oracle instance-hiding* scenario [3, 4]. In this setting there is a function $F_m : \{0,1\}^m \to \{0,1\}$ held by k oracles. The user has $P \in \{0,1\}^m$ and wants to privately retrieve $F_m(P)$, even if up to t of the oracles collude. The user's computation, let alone the total communication, should be polynomial in m. For $k = \frac{m}{c \log m}$ and constant t, running our PIR scheme on the truth table of F_m gives a scheme with total communication $\tilde{O}(m^{ct/2+2})$. This improves the previous bound of $\tilde{O}(m^{ct/2+2+t})$ (see [15]) by a factor of m^t. When $m = \log n$, this is exactly the problem of PIR with $k = \Omega(\log n / \log\log n)$, for which we obtain the best known bound.

Another application of our techniques is k-out-of-l *robust* PIR [9]. In this scenario a user should be able to recover x_i even if after sending his queries, up to $l - k$ servers do not respond. Previous bounds for this problem include $O(kn^{1/k} l \log l)$ and $2^{\tilde{O}(k)} n^{\frac{\delta \log\log k}{k \log k}} l \log l$ [9] for a small constant $\delta > 0$. The first bound has a better dependence on k, while the second has a better dependence on n. We improve upon the former with a k-out-of-l robust protocol with communication $O(kn^{1/(2k-1)} l \log l)$.

Another concern with the abovementioned solutions is the *time complexity* of the servers per query. Beimel, Ishai, and Malkin [8] show, among other things, that if two servers are given polynomial-time preprocessing, then during the online stage they can respond to queries with $O(n/\log^2 n)$ work, while preserving $O(n^{1/3})$ total communication. By combining a balancing technique similar to that in [10] with a specially designed 2-server protocol in our language, we can reduce the work to $O(n/\log^r n)$ for any constant $r > 0$.

Finally, we believe that an important advantage of our PIR protocols is the simple geometric intuition behind them. The novel technique that we introduce has already found additional applications in the area of secure multiparty computation [2], and we expect further applications to follow.

1.1. Omitted from this version. The original version of this paper [26] contained a section with alternative geometric proofs of some of the upper bounds of Beimel et al. [7] for 1-private k-server PIR protocols. The initial argument of [7] involves a highly nontrivial recursion and is fairly hard to follow. Our proofs shed new light on the nature of these protocols. However, after the original paper [26] was published, these results were superseded by [27] and are therefore omitted from this version.

1.2. Our techniques. The general idea behind our protocols is the idea of polynomial interpolation. As in several previous works, we model the database as a degree-d polynomial $F \in \mathbb{F}_q[z_1, \ldots, z_m]$ with $m = O(dn^{1/d})$. The polynomial F is chosen in such a way that for some encoding $E : [n] \to \mathbb{F}_q^m$ (which is independent of F) we have $F(E(i)) = x_i$ for every $i \in [n]$. The user wants to retrieve the value $F(P)$ for $P = E(i)$ while keeping the identity of P private. To this end the user randomly selects a low-dimensional affine curve (in many cases this is simply an affine line) $\chi \subseteq \mathbb{F}_q^m$ containing the point P and discloses certain points on χ to the servers. Each server computes and returns the value of F and the values of *partial derivatives* of F at the point that it is given. Finally, the user reconstructs the restriction of F to χ. In particular, the user obtains the desired value of $F(P)$. The idea of polynomial interpolation has been used previously in the PIR literature [3, 11, 4]; however, we significantly extend and improve upon earlier techniques through the use of partial derivatives and curves.

A possible high level interpretation of our results is to say that we modify the protocols of [15], replacing the replication-based secret-sharing scheme by a more efficient Shamir's scheme [21]. A discussion of certain merits of replication-based secret sharing (that come at a price of increased communication) can be found in [12]. For another use of derivatives in polynomial-based secret-sharing schemes, see [22].

1.3. Outline. In section 2 we introduce our notation and provide some necessary definitions. In section 3 we describe a 1-private PIR protocol on a line. We also discuss the robustness of our protocol. Section 4 deals with t-private PIR protocols on a curve for arbitrary t and discusses applications to instance hiding. In section 5 we present our construction of PIR protocols with preprocessing.

2. Preliminaries. By default, variables λ_h take values in a finite field $\mathbb{F}_q$, and variables P, V, V^j, Q, and Q^j take values in $\mathbb{F}_q^m$. Let W be an element of $\mathbb{F}_q^m$. We use the subscript W_l to denote the lth component of W.

A k-server PIR protocol involves k servers $\mathcal{S}_1, \ldots, \mathcal{S}_k$, each holding the same n-bit string x (the database), and a user $\mathcal{U}$ who knows n and wants to retrieve some bit x_i, $i \in [n]$, without revealing i. We restrict our attention to *one-round,* information-theoretic PIR protocols.

Definition. [7] A t-private PIR protocol is a triplet of algorithms $\mathcal{P} = (\mathcal{Q}, \mathcal{A}, \mathcal{C})$. At the beginning of the protocol, the user $\mathcal{U}$ invokes $\mathcal{Q}(k, n, i)$ to pick a randomized k-tuple of queries $(q_1, \ldots, q_k)$, along with an auxiliary information string aux. It sends each server $\mathcal{S}_j$ the query q_j and keeps aux for later use. Each server $\mathcal{S}_j$ responds with an answer $a_j = \mathcal{A}(k, j, x, q_j)$. (We can assume without loss of generality that the servers are deterministic; hence, each answer is a function of a query and a database.) Finally, $\mathcal{U}$ computes its output by applying the reconstruction algorithm $\mathcal{C}(k, n, a_1, \ldots, a_k, aux)$. A protocol as above should satisfy the following requirements:

- *Correctness.* For any k, n, $x \in \{0,1\}^n$ and $i \in [n]$, the user outputs the correct value of x_i with probability 1 (where the probability is over the randomness of $\mathcal{Q}$).
- *t-Privacy.* Each collusion of up to t servers learns no information about i. Formally, for any k, n, $i_1, i_2 \in [n]$, and every $T \subseteq [k]$ of size $|T| \leq t$ the

distributions $\mathcal{Q}_T(k, n, i_1)$ and $\mathcal{Q}_T(k, n, i_2)$ are identical, where $\mathcal{Q}_T$ denotes concatenation of jth outputs of $\mathcal{Q}$ for $j \in T$.

The *communication complexity* of a PIR protocol $\mathcal{P}$, denoted $C_{\mathcal{P}}(n, k)$, is a function of k and n measuring the total number of bits communicated between the user and k servers, maximized over all choices of $x \in \{0,1\}^n$, $i \in [n]$, and random inputs.

In all our protocols we represent the database x by a multivariate polynomial $F(z_1, \ldots, z_m)$ over a finite field. The important parameters of the polynomial F are its degree-d and the number of variables m. A very similar representation has been used previously in [7]. The polynomial F represents x in the following sense: with every $i \in [n]$ we associate a point $E(i) \in \mathbb{F}_q^m$; the polynomial F satisfies

$$\forall i \in [n], \quad F(E(i)) = x_i.$$

We use the assignment function $E : [n] \to \mathbb{F}_q^m$ from [7]. Let $E(1), \ldots, E(n)$ denote n distinct points of Hamming weight[1] d with coordinate values from the set $\{0,1\} \subset \mathbb{F}_q$. Such points exist if $\binom{m}{d} \geq n$. Therefore $m = O(dn^{1/d})$ variables are sufficient. Define

$$F(z_1, \ldots, z_m) = \sum_{i=1}^{n} x_i \prod_{E(i)_l = 1} z_l$$

($E(i)_l$ is the lth coordinate of $E(i)$.) Since each $E(i)$ is of weight d, the degree of F is d. Each assignment $E(i)$ to the variables z_i satisfies exactly one monomial in F (whose coefficient is x_i); thus, $F(E(i)) = x_i$.

Our constructions rely heavily on the notion of a derivative of a polynomial over a finite field. Recall that for $f(\lambda) = a_0 + \sum_{i=1}^{d} a_i \lambda^i \in \mathbb{F}_q[\lambda]$ the derivative is defined by $f'(\lambda) = \sum_{i=1}^{d} i a_i \lambda^{i-1}$. Higher order derivatives are defined recursively by $f^{(s)}(\lambda) = \left(f^{(s-1)}(\lambda)\right)'$.

We conclude the section with two technical lemmas.

Lemma 1. *Let $f \in \mathbb{F}_q[\lambda]$ and let s be an integer smaller than the characteristic of $\mathbb{F}_q$. Suppose $f(\lambda_0) = f'(\lambda_0) = \cdots = f^{(s)}(\lambda_0) = 0$; then $(\lambda - \lambda_0)^{s+1} \mid f$.*

Proof. See Lemma 6.51 in [17]. □

Lemma 2. *Suppose $\{\lambda_h\}, \{v_h^0\}, \{v_h^1\}$ are elements of $\mathbb{F}_q$, where $h \in [s]$ and $\{\lambda_h\}$ are distinct; then there exists at most one polynomial $f(\lambda) \in \mathbb{F}_q[\lambda]$ of degree $\leq 2s - 1$ such that $f(\lambda_h) = v_h^0$ and $f'(\lambda_h) = v_h^1$.*

Proof. Assume there exist two such polynomials $f_1(\lambda)$ and $f_2(\lambda)$. Consider their difference $f = f_1 - f_2$. Clearly, $f(\lambda_h) = f'(\lambda_h) = 0$ for all $h \in [s]$. Therefore, by Lemma 1,

$$\prod_{h=1}^{s} (\lambda - \lambda_h)^2 \;\Bigg|\; f(\lambda).$$

This divisibility condition yields $f(\lambda) = 0$ since the degree of f is at most $2s - 1$. □

3. PIR on the line.

We start this section with a PIR protocol of [11]. This protocol has a simple geometric interpretation and has served as the starting point for our work.

Theorem 3 (see [11]). *There exists a 1-private k-server PIR protocol with communication complexity $O(k^2 \log k \; n^{1/(k-1)})$.*

[1] The Hamming weight of a vector is defined to be the number of nonzero coordinates.

Protocol description. Consider a finite field $\mathbb{F}_q$, where $k < q \leq 2k$. Let $\lambda_1, \ldots, \lambda_k \in \mathbb{F}_q$ be distinct and nonzero. Set $d = k - 1$. Let $P = E(i)$. The user wants to retrieve $F(P)$.

$\mathcal{U}$	:	Picks $V \in \mathbb{F}_q^m$ uniformly at random.
$\mathcal{U} \to \mathcal{S}_h$	:	$P + \lambda_h V$.
$\mathcal{U} \leftarrow \mathcal{S}_h$	:	$F(P + \lambda_h V)$.

Privacy. It is immediate to verify that the input $(P + \lambda_h V)$ of each server $\mathcal{S}_i$ is distributed uniformly over $\mathbb{F}_q^m$. Thus the protocol is private.

Correctness. We need to show that values $F(P + \lambda_h V)$ for $h \in [k]$ suffice to reconstruct $F(P)$. Consider the line $L = \{P + \lambda V \mid \lambda \in \mathbb{F}_q\}$ in the space $\mathbb{F}_q^m$. Let $f(\lambda) = F(P + \lambda V)$ be the restriction of F to L. Clearly, $f \in \mathbb{F}_q[\lambda]$ is a univariate polynomial of degree at most $d = k - 1$. Note that $f(\lambda_h) = F(P + \lambda_h V)$. Thus $\mathcal{U}$ knows the values of $f(\lambda)$ at k points and therefore can reconstruct $f(\lambda)$. It remains to note that $F(P) = f(0)$.

Complexity. The user sends each of k servers a length-m vector of values in $\mathbb{F}_q$. Recall that $m = O(dn^{1/d})$ and $k < q \leq 2k$. Thus the total communication from the user to all the servers is $O(k^2 \log k \; n^{1/(k-1)})$. Each $\mathcal{S}_h$ responds with a single value from $\mathbb{F}_q$, which does not affect the asymptotic communication of the protocol.

In the protocol above there is an obvious imbalance between the communication from the user to the servers and vice versa. The next theorem extends the technique of Theorem 3 to fix this imbalance and obtain a better communication complexity.

THEOREM 4. *There exists a* 1*-private k-server PIR protocol with communication complexity* $O(k^2 \log k \; n^{1/(2k-1)})$.

Protocol description. We use the standard mathematical notation $\frac{\partial F}{\partial z_l}\big|_Q$ to denote the value of the partial derivative of F with respect to z_l at point Q. Let $\lambda_1, \ldots, \lambda_k \in \mathbb{F}_q$ be distinct and nonzero. Set $d = 2k - 1$. Let $P = E(i)$. The user wants to retrieve $F(P)$.

$\mathcal{U}$	:	Picks $V \in \mathbb{F}_q^m$ uniformly at random.
$\mathcal{U} \to \mathcal{S}_h$	:	$P + \lambda_h V$.
$\mathcal{U} \leftarrow \mathcal{S}_h$	:	$F(P + \lambda_h V), \frac{\partial F}{\partial z_1}\Big|_{P + \lambda_h V}, \ldots, \frac{\partial F}{\partial z_m}\Big|_{P + \lambda_h V}.$

Privacy. The proof of privacy is identical to the proof from Theorem 3.

Correctness. Again, consider the line $L = \{P + \lambda V \mid \lambda \in \mathbb{F}_q\}$. Let $f(\lambda) = F(P + \lambda V)$ be the restriction of F to L. Clearly, $f(\lambda_h) = F(P + \lambda_h V)$. Thus the user knows the values $\{f(\lambda_h)\}$ for all $h \in [k]$. However, this time the values $\{f(\lambda_h)\}$ do not suffice to reconstruct the polynomial f, since the degree of f may be up to $2k - 1$. The main observation underlying our protocol is that knowing the values of partial derivatives $\frac{\partial F}{\partial z_1}\big|_{P + \lambda_h V}, \ldots, \frac{\partial F}{\partial z_m}\big|_{P + \lambda_h V}$, the user can reconstruct the value of $f'(\lambda_h)$. The proof is a straightforward application of the chain rule:

$$\frac{\partial f}{\partial \lambda}\bigg|_{\lambda_h} = \frac{\partial F(P + \lambda V)}{\partial \lambda}\bigg|_{\lambda_h} = \sum_{l=1}^{m} \frac{\partial F}{\partial z_l}\bigg|_{P + \lambda_h V} V_l.$$

Thus the user can reconstruct $\{f(\lambda_h)\}$ and $\{f'(\lambda_h)\}$ for all $h \in [k]$. Combining this observation with Lemma 2, we conclude that the user can reconstruct f and obtain $F(P) = f(0)$.

Complexity. The user sends each of k servers a length-m vector of values in $\mathbb{F}_q$. Servers respond with length-$(m+1)$ vectors of values in $\mathbb{F}_q$. Recall that $m = O(dn^{1/d})$ and $q \leq 2k$. Thus the total communication is $O(k^2 \log k \; n^{1/(2k-1)})$.

3.1. Application to robust PIR. We review the definition of robust PIR [9].

Definition 5. *A k-out-of-l PIR protocol is a 1-private PIR protocol with the additional property that the user always computes the correct value of x_i from any k out of l of the answers.*

As noted in [9], robust PIR has applications to servers which may hold different versions of a database, as long as some k have the latest version and there is a way to distinguish these k. Another application is to servers with varying response times. Previous bounds for this problem include $O(kn^{1/k} l \log l)$ and $2^{\tilde{O}(k)} n^{\frac{\delta \log \log k}{k \log k}} l \log l$ [9] for a small constant $\delta > 0$. The first bound has a better dependence on k, while the second has a better dependence on n. We improve upon the former with a k-out-of-l robust protocol with communication $O(kn^{1/(2k-1)} l \log l)$.

Indeed, in the protocol above, if for l servers we set the field size $q > l$ and the degree $\deg F = 2k-1$, then from any k servers' answers, we can reconstruct f as before. We conclude that the following theorem holds.

Theorem 6. *For all $k \leq l$ there exists a k-out-of-l robust PIR with communication complexity $O(kn^{1/(2k-1)} l \log l)$.*

4. PIR on the curve. Now we proceed to t-private PIRs for general t.

Theorem 7. *For all $t \leq k$ there exists a t-private k-server PIR protocol with communication complexity $O\big(\frac{k^2}{t} \log k \; n^{1/\lfloor \frac{2k-1}{t} \rfloor}\big)$.*

Protocol description. Again, consider $\mathbb{F}_q$, where $k < q \leq 2k$, and let $\lambda_1, \ldots, \lambda_k \in \mathbb{F}_q$ be distinct and nonzero. Set $d = \lfloor \frac{2k-1}{t} \rfloor$. Let $P = E(i)$. The user wants to retrieve $F(P)$.

$$\boxed{\begin{array}{lcl} \mathcal{U} & : & \text{Randomly picks } V^1, \ldots, V^t \in \mathbb{F}_q^m. \\ \mathcal{U} \rightarrow \mathcal{S}_h & : & Q^h = P + \lambda_h V^1 + \lambda_h^2 V^2 + \ldots + \lambda_h^t V^t. \\ \mathcal{U} \leftarrow \mathcal{S}_h & : & F(Q^h), \left.\frac{\partial F}{\partial z_1}\right|_{Q^h}, \ldots, \left.\frac{\partial F}{\partial z_m}\right|_{Q^h}. \end{array}}$$

Privacy. We need to show that for every $T \subseteq [k]$, where $|T| \leq t$, the collusion of servers $\{\mathcal{S}_h\}_{h \in T}$ learns no information about the point $P = E(i)$. The joint input of servers $\{\mathcal{S}_h\}_{h \in T}$ is $\{P + \lambda_h V^1 + \cdots + \lambda_h^t V^t\}_{h \in T}$. Since the coordinates are shared independently, it suffices to show that for each $l \in [m]$ and $V_l^j \in \mathbb{F}_q$ chosen independently and uniformly at random, the values $\{P_l + \lambda_h V_l^1 + \cdots + \lambda_h^t V_l^t\}_{h \in T}$ disclose no information about P_l. The last statement follows immediately from the fact that for any distinct $\lambda_0, \ldots, \lambda_t \in \mathbb{F}_q$ and arbitrary $u_0, \ldots, u_t \in \mathbb{F}_q$ there exists a unique polynomial $f(\lambda) \in \mathbb{F}_q[\lambda]$ of degree up to t such that $f(\lambda_i) = u_i$ for all i. One other way to argue privacy is to observe that our queries can be obtained by independently secret sharing each of the m coordinates of P according to Shamir's scheme [21].

Correctness. Consider the curve $\chi = \{P + \lambda V^1 + \cdots + \lambda^t V^t \mid \lambda \in \mathbb{F}_q\}$. Let $f(\lambda) = F(P + \lambda V^1 + \cdots + \lambda^t V^t)$ be the restriction of F to χ. Obviously, f is a univariate polynomial of degree at most $2k-1$. By definition, we have $f(\lambda_h) = F(Q^h)$; thus $\mathcal{U}$ knows the values $\{f(\lambda_h)\}$ for all $h \in [k]$. Now we shall see how knowing the values of partial derivatives $\left.\frac{\partial F}{\partial z_1}\right|_{Q^h}, \ldots, \left.\frac{\partial F}{\partial z_m}\right|_{Q^h}$ $\mathcal{U}$ reconstructs the value of $f'(\lambda_h)$.

Again, the reconstruction is a straightforward application of the chain rule:

$$\left.\frac{\partial f}{\partial \lambda}\right|_{\lambda_h} = \left.\frac{\partial F(P+\lambda V^1+\cdots+\lambda^t V^t)}{\partial \lambda}\right|_{\lambda_h} = \sum_{l=1}^{m} \left.\frac{\partial F}{\partial z_l}\right|_{Q^h} \left.\frac{\partial}{\partial \lambda}(P_l+\lambda V_l^1+\cdots+\lambda^t V_l^t)\right|_{\lambda_h}.$$

Thus $\mathcal{U}$ can reconstruct $\{f(\lambda_h)\}$ and $\{f'(\lambda_h)\}$ for all $h \in [k]$. Combining this observation with Lemma 2, we conclude that the user can reconstruct f and obtain $F(P) = f(0)$.

Complexity. As in the protocol of Theorem 4, $\mathcal{U}$ sends each of k servers a length-m vector of values in $\mathbb{F}_q$, and servers respond with length-$(m+1)$ vectors of values in $\mathbb{F}_q$. Here $m = O(dn^{1/d})$ and $q \leq 2k$. Thus the total communication is $O\big(\frac{k^2}{t}\log k\; n^{1/\lfloor\frac{2k-1}{t}\rfloor}\big)$.

4.1. Application to instance hiding. As noted in the introduction, in the instance-hiding scenario [3, 4] there is a function $F_m : \{0,1\}^m \to \{0,1\}$ held by k oracles. The user has a point $P \in \{0,1\}^m$ and should learn $F_m(P)$. Further, the view of up to t oracles should be independent of P. When $k = \frac{m}{c\log m}$, we have the following improvement upon the best known $\tilde{O}(m^{ct/2+2+t})$ communication bound of [15].

THEOREM 8. *Let $c, t > 0$, and suppose that t is an integer. Then for any $m > 0$ there is an oracle instance-hiding scheme with $k = \frac{m}{c\log m}$ oracles and communication and computation $\tilde{O}(m^{ct/2+2})$, where $\tilde{O}(f) = O(f\log^{O(1)} f)$.*

Proof. Using the above protocol on the truth table of F_m, the communication is

$$O\left(\frac{k^2}{t}\log k n^{1/\lfloor(2k-1)/t\rfloor}\right) = \tilde{O}\left(m^2\cdot(2^m)^{(\lfloor(2k-1)/t\rfloor)^{-1}}\right) = \tilde{O}(m^{ct/2+2}).$$

It is also easy to see that $\mathcal{U}$ runs in time which is quasi-linear in the communication. □

5. PIR with preprocessing. To measure the efficiency of an algorithm with preprocessing, we use the definition of *work* in [8] which counts the number of precomputed bits and database bits that need to be read in order to respond to a query. We note that, as mentioned in [8], this definition of work may be more appropriate for proving lower bounds than for upper bounds. However, as in that paper, the actual amount of work in the unit-cost RAM model[2] of our scheme is linear in the number of bits read.

The goal of this section is to prove the following theorem.

THEOREM 9. *There exists a 2-server PIR protocol with $O(n^{1/3})$ communication,* poly(n) *preprocessing, and $O(n/(\log n)^r)$ server work for any constant r.*

The reader may wonder what happens if we consider larger values of r in this theorem. To simplify the presentation of this section, we defer the discussion of this to Remark 11.

We need a lemma about preprocessing polynomials $F \in \mathbb{F}_p[z_1, \ldots, z_m]$. We assume the number of variables m is tending to infinity, while the degree of F is constant. This lemma is similar to Theorem 3.1 of [8]. The main idea is to write the input polynomial F as a sum of poly(m) polynomials G each over disjoint monomials. We do this in such a way that each G involves only a logarithmic number of variables. Thus we can precompute G on all possible assignments. As the different G are over disjoint

[2] Recall that in the unit-cost RAM model each field operation can be performed in constant time.

monomials, to evaluate $F(V)$ we simply read one precomputed answer for each G and sum them up.

LEMMA 10. *Let F be a homogeneous degree-d polynomial in $\mathbb{F}_p[z_1, \ldots, z_m]$. Using* $\mathrm{poly}(m)$ *preprocessing time, for all $V \in \mathbb{F}_p^m$, $F(V)$ can be computed with $O(m^d/\log^d m)$ work.*

Proof. Partition $[m]$ into $\alpha = m/\log m$ disjoint sets $D_1, \ldots, D_\alpha$ of size $\log m$. For every sequence $1 \le t_1, \ldots, t_d \le \alpha$, let $F_{D_{t_1},\ldots,D_{t_d}}$ denote the sum of all monomials of F of the form $cz_{i_1} \cdots z_{i_d}$ for some $c \in \mathbb{F}_p$ and $i_1 \in D_{t_1}, \ldots, i_d \in D_{t_d}$. The following is the preprocessing algorithm.

```
Preprocess(F):
1. For each polynomial $F_{D_{t_1},\ldots,D_{t_d}}$,
     Evaluate $F_{D_{t_1},\ldots,D_{t_d}}$ on all $W \in \mathbb{F}_p^m$
     for which $\mathrm{Supp}(W) \subseteq \cup_i D_{t_i}$.
```

Time Complexity. There are $\alpha^d = (m/\log m)^d$ polynomials $F_{D_{t_1},\ldots,D_{t_d}}$. For each polynomial, there are at most $p^{d \log m} = \mathrm{poly}(m)$ different W whose support is in $\cup_i D_{t_i}$. Thus the total time of the algorithm is $\mathrm{poly}(m)$.

For a set $S \subseteq [m]$, let $V|_S$ denote the point $V' \in \mathbb{F}_p^m$ with $V'_j = V_j$ for $j \in S$ and $V'_j = 0$ otherwise. The following describes how to compute $F(V)$.

```
Evaluate($F, V$):
1. $\sigma \leftarrow 0$.
2. For each polynomial $F_{D_{t_1},\ldots,D_{t_d}}$,
     $\sigma \leftarrow \sigma + F_{D_{t_1},\ldots,D_{t_d}}(V|_{\cup_i D_{t_i}})$.
3. Output $\sigma$.
```

Correctness. This is immediate from

$$F(V) = \sum_{t_1,\ldots,t_d} F_{D_{t_1},\ldots,D_{t_d}}(V|_{\cup_i D_{t_i}}).$$

Work. The sum is over $\alpha^d = (m/\log m)^d$ polynomials $F_{D_{t_1},\ldots,D_{t_d}}$, each with a precomputed answer, and thus the total work is $O(m^d/\log^d m)$. □

5.1. Two server protocol. We start with the intuition underlying our 2-server preprocessing protocol. Suppose the servers were to represent the database as a degree-d polynomial F in $m = \Theta(n^{1/d})$ variables, where $d = 2r + 1$ is an arbitrary odd constant. Proceeding as in the protocol of section 3, the user sends each server a point on a random line L through his point of interest. To reconstruct $F|_L$, the user needs the evaluation of F on his query points, together with all partial derivatives of F up to order r. The observation is that each partial derivative computed by the servers is either a zero polynomial or a polynomial of degree at least $d - r = r + 1$ in at most m variables, and therefore we can apply Lemma 10 to achieve low server work.

However, while the user is sending only $O(m)$ bits to the servers, the servers' answers are of size $O(m^r)$. To fix this, we use a balancing technique similar to that in [10]. Each server partitions the database into t databases F_j, each of size n/t, for some parameter t. Each database will be represented as a degree-d polynomial in $m = O((n/t)^{1/d})$ variables. The user sends t points to each server, one for each database. Suppose the user wants $F_u(P)$. For the $t-1$ databases F_j, $j \neq u$, that do not interest the user, the user sends random V^j and $-V^j$ to servers 1 and 2, respectively. On the other hand, for the database F_u that contains the value of interest, the user proceeds as in the protocol of section 3. The servers compute the lists of partial derivatives for each database, as before, but instead of sending them back, they send the *sum* of each partial derivative over all t databases. We show this information is sufficient for the user to reconstruct $F_u(P)$. The total work will be $O(n/\log^{r+1} n)$, and by carefully choosing t, we can keep the communication at $O(n^{1/3})$.

Consider a prime field[3] $\mathbb{F}_p$ for some $\max(2, r) < p < 2\max(2, r)$. Such a prime p exists by Bertrand's postulate [19]. $\mathcal{S}_1$ and $\mathcal{S}_2$ preprocess as follows.

```
Preprocessing phase($x$):
1. $s \leftarrow \frac{r-1}{3r}$, $t \leftarrow n^s$.
2. Partition $x$ into $t$ databases $\mathrm{DB}_1, \ldots, \mathrm{DB}_t$, each
   containing $n^{1-s}$ elements.
3. Represent $\mathrm{DB}_j$ as a homogeneous polynomial $F_j$
   of degree $d = 2r + 1$ with $m = O\left(n^{(1-s)/d}\right)$ vars.
4. For $a = 0, \ldots, r$, for $j \in [t]$, and for $l_1, \ldots, l_a \in [m]$,
   compute Preprocess$\left(\frac{\partial^a F_j}{\partial z_{l_1} \cdots \partial z_{l_a}}\right)$.
```

Let DB_u be the database containing x_i. Assume the user wants $F_u(P)$. Let $\delta_{\alpha,\beta}$ be 1 if $\alpha = \beta$ and 0 otherwise.

```
$\mathcal{U}$                      : Randomly picks $V^1, \ldots, V^t \in \mathbb{F}_p^m$.
$\mathcal{U} \rightarrow \mathcal{S}_h$ : For $j \in [t]$, $Q^{h,j} = (-1)^{h+1} V^j + \delta_{j,u} P$.
$\mathcal{U} \leftarrow \mathcal{S}_h$  : $\forall a \in \{0, \ldots, r\}$ and $l_1, \ldots, l_a \in [m]$,
                             $\sum_{j=1}^t \frac{\partial^a F_j}{\partial z_{l_1} \cdots \partial z_{l_a}}\Big|_{Q^{h,j}} =$
                             $\sum_{j=1}^t$ Evaluate$\left(\frac{\partial^a F_j}{\partial z_{l_1} \cdots \partial z_{l_a}}, Q^{h,j}\right)$
```

Correctness. Since d is odd, for all V

$$\frac{\partial^a F_j}{\partial z_{l_1} \cdots \partial z_{l_a}}\bigg|_{-V} = (-1)^{a+1} \frac{\partial^a F_j}{\partial z_{l_1} \cdots \partial z_{l_a}}\bigg|_{V}.$$

It follows that for all a and all $j \neq u$,

$$\sum_{l_1,\ldots,l_a} \frac{\partial^a F_j}{\partial z_{l_1} \cdots \partial z_{l_a}}\bigg|_{V^j} + \sum_{l_1,\ldots,l_a} \frac{(-1)^a \partial^a F_j}{\partial z_{l_1} \cdots \partial z_{l_a}}\bigg|_{-V^j} = 0.$$

Put $f(\lambda) = (F_u)|_{P+\lambda V^u}$, and define $g(\lambda) = f(\lambda) + f(-\lambda)$. We then have

$$\begin{aligned}
&\sum_j \sum_{l_1,\ldots,l_a} \frac{\partial^a F_j}{\partial z_{l_1} \cdots \partial z_{l_a}}\bigg|_{Q^{1,j}} V_{l_1}^u \cdots V_{l_a}^u + \frac{(-1)^a \partial^a F_j}{\partial z_{l_1} \cdots \partial z_{l_a}}\bigg|_{Q^{2,j}} V_{l_1}^u \cdots V_{l_a}^u \\
&= \sum_{l_1,\ldots,l_a} \frac{\partial^a F_u}{\partial z_{l_1} \cdots \partial z_{l_a}}\bigg|_{P+V^u} V_{l_1}^u \cdots V_{l_a}^u + \frac{(-1)^a \partial^a F_u}{\partial z_{l_1} \cdots \partial z_{l_a}}\bigg|_{P-V^u} V_{l_1}^u \cdots V_{l_a}^u \\
&= f^{(a)}(1) + (-1)^a f^{(a)}(-1) = g^{(a)}(1).
\end{aligned}$$

[3] In section 5 we base our protocols on prime fields $\mathbb{F}_p$ and do not consider general finite fields $\mathbb{F}_q$. We do this to avoid issues related to subtle properties of derivatives of orders greater than one in finite fields of small characteristic. Another possible solution to this problem is to use Hasse derivatives (referred to as hyperderivatives in [17]) instead of usual derivatives. This allows for protocols over arbitrary finite fields.

Thus $\mathcal{U}$ can compute $g(1), g^{(1)}(1), \dots, g^{(r)}(1)$ from the answers. Since every monomial of g has even degree, for $\gamma = \lambda^2$ we can define $h(\gamma) = g(\lambda)$ for a degree-r polynomial h. Using that

$$\frac{dg}{d\lambda} = \frac{dh}{d\gamma} \cdot \frac{d\gamma}{d\lambda} = 2\lambda \frac{dh}{d\gamma},$$

a simple induction shows that from $g^{(0)}(1), \dots, g^{(r)}(1)$, $\mathcal{U}$ can get $h^{(0)}(1), \dots, h^{(r)}(1)$. The claim is that these values determine h. Indeed, if $h_1 \neq h_2$ agree on these values, then by Lemma 1,

$$(\gamma - 1)^{r+1} \;\Big|\; (h_1 - h_2),$$

which contradicts that $h_1 - h_2$ has degree at most r. Hence the user obtains $h(0) = g(0) = 2f(0) = 2F(P)$, and thus $F(P)$ since the characteristic $p > 2$.

Privacy. Since the V^j are independent and uniformly random, so are the $Q^{1,j}$ and the $Q^{2,j}$. Thus the view of each of $\mathcal{S}_1$, $\mathcal{S}_2$ is independent of P.

Communication. $\mathcal{U}$ sends $O(tm) = O(n^{s+(1-s)/(2r+1)}) = O(n^{(r-1)/(3r)+1/(3r)}) = O(n^{1/3})$ bits. $\mathcal{S}_1, \mathcal{S}_2$ respond with $O(m+m^2+\cdots+m^r) = O(m^r) = O(n^{(1-s)r/(2r+1)}) = O(n^{1/3})$ bits.

Server work. Notice that the work is dominated by the calls to $\mathsf{Evaluate}$. For any $a \in \{0, \dots, r\}$, any $l_1, \dots, l_a \in [m]$, and any $j \in [t]$, the polynomial $\frac{\partial^a F_j}{\partial z_{l_1} \cdots \partial z_{l_a}}$ is either 0 or has degree $2r + 1 - a$ and at most m variables. Thus for any V, $\mathsf{Evaluate}(\frac{\partial^a F_j}{\partial z_{l_1} \cdots \partial z_{l_a}}, V)$ can be computed in $O(m^{2r+1-a}/\log^{2r+1-a} m)$ time. As the number of such $\frac{\partial^a F_j}{\partial z_{l_1} \cdots \partial z_{l_a}}$ is $O(m^a)$, it follows that the time for all calls to $\mathsf{Evaluate}$ per DB is

$$\sum_a O\left(\frac{m^a m^{2r+1-a}}{\log^{2r+1-a} m}\right) = \frac{O(m^d)}{\log^{r+1} m} = \frac{O(n^{1-s})}{\log^{r+1} n}.$$

Thus the total work over all n^s DBs is $O(n/\log^{r+1} n)$.

Remark 11. Now we consider what happens if r is allowed to grow with n. In step 3 of the preprocessing phase we now need the number of variables m to be $\Theta(dn^{1-s})$, where the degree $d = 2r + 1$. Step 4 takes $(r + 1)tm^{O(d)}\text{poly}(\log p)$ time, which is $\text{poly}(n, d^d)$. This step is still efficient provided that $d = O\big(\frac{\log n}{\log \log n}\big)$.

However, in the protocol the communication is now $\Theta((tm+m^r) \log p) = d^{\Theta(d)} n^{1/3}$. This is larger than the desired $O(n^{1/3})$ communication for any superconstant value of d, and thus any superconstant value of r. Similarly, the server work increases by a $d^{\Theta(d)}$ factor. It is an interesting open question to reduce the server work while preserving $O(n^{1/3})$ total communication.

Acknowledgment. Sergey Yekhanin would like to express his deep gratitude to Madhu Sudan for introducing him to the problem and for many helpful discussions during this work.

REFERENCES

[1] A. Ambainis, *Upper bound on the communication complexity of private information retrieval*, in Automata, Languages and Programming, Lecture Notes in Comput. Sci. 1256, Springer, Berlin, 1997, pp. 401–407.

[2] O. Barkol and Y. Ishai, *Secure computation of constant-depth circuits with applications to database search problems*, in Advances in Cryptology—CRYPTO 2005, Lecture Notes in Comput. Sci. 3621, Springer, Berlin, 2005, pp. 395–411.

[3] D. Beaver and J. Feigenbaum, *Hiding instances in multioracle queries*, in Proceedings of the STACS '90, Lecture Notes in Comput. Sci. 415, Springer, Berlin, 1990, pp. 37–48.

[4] D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway, *Locally random reductions: Improvements and applications*, J. Cryptology, 10 (1997), pp. 17–36.

[5] R. Beigel, L. Fortnow, and W. Gasarch, *A tight lower bound for restricted PIR protocols*, Comput. Complexity, 15 (2006), pp. 82–91.

[6] A. Beimel and Y. Ishai, *Information-theoretic private information retrieval: A unified construction*, in Automata, Languages and Programming, Lecture Notes in Comput. Sci. 2076, Springer, Berlin, 2001, pp. 912–926.

[7] A. Beimel, Y. Ishai, E. Kushilevitz, and J. F. Raymond, *Breaking the* $O\left(n^{1/(2k-1)}\right)$ *barrier for information-theoretic private information retrieval*, in Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science (FOCS), 2002, pp. 261–270.

[8] A. Beimel, Y. Ishai, and T. Malkin, *Reducing the servers' computation in private information retrieval: PIR with preprocessing*, J. Cryptology, 17 (2004), pp. 125–151.

[9] A. Beimel and Y. Stahl, *Robust information-theoretic private information retrieval*, in Proceedings of the 3rd Conference on Security in Communications Networks (SCN 2002), Lecture Notes in Comput. Sci. 2576, Springer, Berlin, 2003, pp. 326–341.

[10] B. Chor and N. Gilboa, *Computationally private information retrieval*, in Proceedings of the 29th ACM Symposium on Theory of Computing (STOC), 1997, pp. 304–313.

[11] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, *Private information retrieval*, J. ACM, 45 (1998), pp. 965–982.

[12] R. Cramer, I. Damgard, and Y. Ishai, *Share conversion, pseudorandom secret-sharing, and applications to secure computation*, in Theory of Cryptology, Lecture Notes in Comput. Sci. 3378, Springer, Berlin, 2005, pp. 342–362.

[13] B. Gasarch, *A Webpage on Private Information Retrieval*, http://www.cs.umd.edu/~gasarch/pir/pir.html (2007).

[14] O. Goldreich, H. Karloff, L. Schulman, and L. Trevisan, *Lower bounds for locally decodable codes and private information retrieval*, Comput. Complexity, 15 (2006), pp. 263–296.

[15] Y. Ishai and E. Kushilevitz, *Improved upper bounds on information-theoretic private information retrieval*, in Proceedings of the 31st ACM Symposium on Theory of Computing (STOC), 1999, pp. 79–88.

[16] T. Itoh, *On lower bounds for the communication complexity of private information retrieval*, IEICE Trans. Fund. of Electronics, Commun. and Comp. Sci., E84-A(1) (2001), pp. 157–164.

[17] R. Lidl and H. Niederreiter, *Finite Fields*, Cambridge University Press, Cambridge, UK, 1985.

[18] E. Mann, *Private Access to Distributed Information*, Master's thesis, Technion - Israel Institute of Technology, Haifa, Israel, 1998.

[19] T. Nagell, *Introduction to Number Theory*, Wiley, New York, 1951.

[20] A. Razborov and S. Yekhanin, *An* $\Omega(n^{1/3})$ *lower bound for bilinear group based private information retrieval*, in Proceedings of the 47th IEEE Symposium on Foundations of Computer Science (FOCS), 2006, pp. 739–748.

[21] A. Shamir, *How to share a secret*, Comm. ACM, 22 (1979), pp. 612–613.

[22] T. Tassa, *Hierarchical threshold secret sharing*, in Theory of Cryptology, Lecture Notes in Comput. Sci. 2951, Springer, Berlin, 2004, pp. 473–490.

[23] L. Trevisan, *Some applications of coding theory in computational complexity*, Quad. Mat., 13 (2004), pp. 347–424.

[24] S. Wehner and R. de Wolf, *Improved lower bounds for locally decodable codes and private information retrieval*, in Automata, Languages and Programming, Lecture Notes in Comput. Sci. 3580, Springer, Berlin, 2005, pp. 1424–1436.

[25] D. Woodruff, *New Lower Bounds for General Locally Decodable Codes*, Report TR07-006, Electronic Colloquium on Computational Complexity, http:/eccc.hpi-web.de/eccc-reports/2007/TR07-006 (18 January 2007).

[26] D. Woodruff and S. Yekhanin, *A geometric approach to information theoretic private information retrieval*, in Proceedings of the 20th IEEE Computational Complexity Conference (CCC), IEEE Computer Society, Washington, DC, 2005, pp. 275–284.

[27] S. Yekhanin, *Towards* 3*-query locally decodable codes of subexponential length*, in Proceedings of the 39th ACM Symposium on Theory of Computing (STOC), 2007, pp. 266–274.

SIAM J. Comput.
Vol. 37, No. 4, pp. 1057–1076

STABILITY OF MULTIVALUED CONTINUOUS CONSENSUS*

LIOR DAVIDOVITCH†, SHLOMI DOLEV†, AND SERGIO RAJSBAUM‡

Abstract. Multivalued consensus functions defined from a vector of inputs over the set V of possible input values (and possibly from the previous input and output values) to a single output are investigated. The consensus functions are designed to tolerate t faulty inputs. Two classes of multivalued consensus functions are defined, the *exact value* and the *range value*, which require the output to be one of the nonfaulty inputs or in the range of the nonfaulty inputs, respectively. The *instability* of consensus functions is examined, counting the maximal number of output changes along a geodesic path of input changes, a path in which each input is changed at most once. Lower and upper bounds for the instability of multivalued consensus functions as a function of n, the number of sensors, t, and $|V|$ are presented. A new technique for obtaining such lower bounds, using edgewise simplex subdivision, is presented.

Key words. consensus, stability

AMS subject classifications. 68Q85, 55U10, 93D99

DOI. 10.1137/S0097539704446335

1. Introduction.

The interest in sensor networks and the way they may control the behavior of a system, such as a vehicle, airplane, satellite, or other device, is rapidly growing. The *agreement* functions used to ensure smooth and stable control while reflecting the changes in the environment are of great interest. An abstraction of many agreement functions is the *consensus* problem, where a set of n processors get input values from some set V and must agree on a value. There is always a nontrivial *validity* requirement that specifies restrictions on the decided value as a function of the input values and the failure pattern of the execution. Consensus is a fundamental problem in distributed computing that has been widely studied for more than two decades due to its theoretical and practical interest (e.g., [2, 7, 14]).

Research on consensus has concentrated on the *one-shot* setting, where processors start with their input values and have to solve consensus once. Often, distributed systems need to solve consensus repeatedly, on inputs received one after the other. Thus, researchers have also investigated *continuous* versions of consensus[1] where processors have to adapt their consensus decisions continuously (e.g., [3, 8, 13]).

Although consensus algorithms are generally associated with distributed systems, this need not necessarily be the case. In fact, it is sufficient that the system has multiple sensors. For instance, typical situations where continuous consensus problems arise are in cases of systems that read values from replicated sensors [12]. A fault-tolerant consensus algorithm is needed to decide on a single reading because sensors

*Received by the editors October 19, 2004; accepted for publication (in revised form) March 5, 2007; published electronically November 14, 2007. An extended abstract of this work was presented at *The Sixth Workshop on Geometric and Topological Methods in Concurrency and Distributed Computing* (GETCO 2004), October, 2004.

http://www.siam.org/journals/sicomp/37-4/44633.html

†Department of Computer Science, Ben-Gurion University, Beer-Sheva 84105, Israel (liord@cs.bgu.ac.il, dolev@cs.bgu.ac.il). The work of the second author was partially supported by an IBM faculty award, NSF grant 0098305, the Israeli Ministry of Defense, the Israeli Ministry of Trade and Industry, and the Rita Altura trust chair in computer sciences.

‡Instituto de Matemáticas, UNAM, Ciudad Universitaria, D.F. 04510, México (rajsbaum@math.unam.mx).

[1] Sometimes called *long-lived* consensus.

usually do not give the exact same reading of a physical parameter, or because some sensors can fail. Although in the simplest version of consensus, which is the one most often considered in theory, the validity requirement is that a decided value must have been read by at least one sensor, in many real settings it is desired that the decided value be a value that has been produced by a majority of the sensors. These and other nontrivial validity requirements are possible, but they all imply that as the readings of the sensors change because the physical parameters that are sampled change, the consensus value will have to change. In the extreme case, all sensors can change their readings from one single value to another, forcing the consensus decision to change accordingly.

Although the consensus value may change several times during the repeated executions of a consensus algorithm, we prefer continuous consensus algorithms that are *stable*, i.e., in which the number of times the decision value is changed is as small as possible. Usually averaging functions are used in an independent way from sample to sample, sometimes combined with agreement protocols (e.g., [15]), and hence there is no attempt to maximize stability. There are several reasons for preferring a stable consensus system (more are described in [8]). Some sensors are discrete and are used to control actuators, which may also be discrete. There is the possible operational amplification of decision changes, say turning an engine on and off. The energy or other resources consumed are sometimes proportional to the number of transitions; e.g., turning an engine on and off takes energy and time, and reduces its lifetime; some related work in VLSI (very large scale integration) is [5, 16]. Our results have an independent purely mathematical interest for analyzing discrete functions over vectors of inputs; they may also be useful for studying problems (e.g., [11]) about the number of influencing variables in Boolean functions.

In [8] we initiated a study of the stability of continuous consensus systems for the case of a binary set V of inputs (i.e., $|V| = 2$). We defined an abstract formalization of a continuous consensus system and the stability measures. This formalization, presented in section 2, is not tied to any specific model of computation, in order to understand the basic stability issues. Moreover, it is also valid for nondistributed consensus systems. We considered *memoryless* systems, where consecutive one-shot consensus executions are independent, versus the stability of systems that can keep memory of previous executions. We also studied the stability of *symmetric* systems, where decisions are taken solely on the basis of the distribution of the different input values, but not on what a specific sensor produced as a particular input value. We characterized the stability of systems according to their memory and symmetry properties, proving tight upper and lower bounds for the various cases.

Results. In this paper we extend the results of [8] to the case of a multivalued set V of inputs (i.e., $|V| \geq 2$). It turns out that this generalization makes all the above variants of the problem much harder. Some of them are much more interesting than those of the binary case; in particular, one of the results here uses topological techniques for high-dimensional complexes.

Let n be the number of sensors, and let t be the number of sensors that may give a wrong value. The validity requirement of [8] is that if fewer than $t + 1$ inputs are equal to a value b, then the consensus value is not b; hence, the decision value is a value of a correct sensor. Another natural validity requirement for a multivalued consensus system is that the decision should be between two correct sensor values. Thus we study the following two classes of systems:

Exact value system (EV): the decision is the value of a correct sensor.

Range value system (RV): the decision is in the range of the correct sensor values.

First we identify the combinations of values of $n, v = |V|, t$ for which systems of these classes exist. We prove that an EV system can be defined if and only if $n \geq vt+1$, and that an RV system can be defined if and only if $n \geq 2t + 1$.

We define the *instability* of a consensus system as the maximal number of changes to the consensus value along any *geodesic path*—a sequence of changes to the input values, where any input value changes at most once. Our choice to examine a geodesic path is motivated by the intuition of condition changes that are monitored by the different sensors. Note that for every (memoryless) consensus algorithm there is an input change that yields an output change, and therefore if the above input is allowed to be changed $k > 1$ times, then obviously the number of changes to the output is at least k. We would like to nullify this phenomenon when we evaluate consensus algorithms, and therefore we choose the geodesic path (see [8] for a further discussion on the motivation of the settings). The instability of a consensus system with memory is analyzed proving that it is n in the cases where n is the smallest possible value, namely $n = vt + 1$, and that the instability may go down until it becomes 1 when $n \geq v^2t+1$. The investigation of the rest of the cases for EV systems results in bounds for a range of stability values as a function of n, v, and t.

Lower bounds on the instability of memoryless systems are obtained for symmetric functions. The lower bounds are achieved using a technique to subdivide a simplex from [9] and Sperner's lemma. This can be seen as a generalization of Lemma 2 in [10]: In the one-dimensional case, the input values may be ordered in a line from one extreme to the other, where two consecutive input vectors differ in exactly one input. In the case of several dimensions, the border between the different extreme values is a simplex. We also present an upper bound for a memoryless symmetric system [6], which is about a factor of 2 away from our lower bound. An interesting open question is how to close this gap.

Organization. The rest of the paper is organized as follows. In the next section, we define the system settings and the problem definitions. The case of systems with memory is addressed in section 3. Memoryless systems are considered in section 4.

2. The model. We start with a definition of a multivalued continuous consensus system which extends the definitions for the binary case of [8]. See [8] for a detailed discussion of the motivation of this system, and for applications to specific distributed computing models. Briefly, the system captures a situation where a system (which may be distributed, but not necessarily) gets inputs from sensors, runs some consensus algorithm to agree on one of the sensors' values, gets another vector of sensor values, runs a consensus algorithm on them, and so on. We consider only the state of the system at the beginning of each such round, i.e., when it gets inputs from the sensors. Also, we consider the value decided by the consensus algorithm at the end of each one of these phases. We are not interested in the details of how the consensus algorithm works. Our goal is to find the minimum number of decision value changes that the algorithm will make over the repeated invocations of the consensus algorithm. Thus, we assume that the consensus algorithm decides the same value in every execution starting in the same state and with the same vector of sensor values.

We will use the following notation. Consider a set $V = \{0, 1, \ldots, v-1\}$ of possible sensor values, and an integer $n > 0$ representing the number of sensors. Then $\vec{x}$ is a vector of n values from V, i.e., in V^n. Also, $\vec{x}\,[i]$ is the ith element of $\vec{x}$. We denote the ith vector in a sequence of vectors by $\vec{x}_i$. Thus, $\vec{x}_i\,[j]$ stands for the jth element of the ith vector in a series of vectors. For a vector $\vec{x}$, $\#b(\vec{x})$ is the number of entries of $\vec{x}$ that are equal to b.

2.1. Continuous consensus systems and instability. A multivalued *continuous consensus system* D, called for short a *system*, is defined by a 6-tuple, D $= \langle n, v, t, S, \tau, f\rangle$, where

- $n > 1$ is an integer representing the number of sensors;
- $v > 1$ is an integer specifying the size of the set V of valid sensors' inputs, $V = \{0, 1, \ldots, v-1\}$;
- $t \geq 0$ is a fault tolerance integer parameter;
- S is a set of *states*, which includes a special *initial state*, s_0;
- τ is a transition function, $\tau : V^n \times S \to S$;
- f is a *decision* function $f : V^n \times S \to V$ such that $f(\vec{x}, s)$ is equal to one of the entries of $\vec{x}$, for every $s \in S$.

An *input vector* is any vector $\vec{x} \in V^n$. Any sequence of input vectors, $\vec{x}_0, \vec{x}_1, \vec{x}_2, \ldots$, induces a unique *execution* of the system, $(\vec{x}_0, s_0) \to (\vec{x}_1, s_1) \to (\vec{x}_2, s_2) \to \cdots$, where for every i, $s_{i+1} = \tau(\vec{x}_i, s_i)$. The *ith output* for the execution is $f(\vec{x}_i, s_i)$. Note that the sequence of input vectors uniquely determines a sequence of output values, $f(\vec{x}_0, s_0), f(\vec{x}_1, s_1), \ldots$.

We use the term *path* for a sequence of input vectors $\vec{x}_0, \vec{x}_1, \ldots$. A component k *changes in the path* if the path contains two vectors $\vec{x}_i, \vec{x}_j$ which differ in their kth components, i.e., such that $\vec{x}_i[k] \neq \vec{x}_j[k]$. A *geodesic path* is a path in which each component k changes at most once. For simplicity, we shall assume, w.l.o.g., that in every step of a geodesic path exactly one input value will be changed. Notice that a geodesic path can be of length at most n, since the vectors of the geodesic paths we consider in this paper are of dimension n.

Definition 2.1. *The* instability *of the system* D *is the largest number of decision changes for any* geodesic *input path.*

The fault-tolerant parameter t represents the largest number of input values in a vector $\vec{x}$ that may be wrong. It is desirable that the decision function f choose an input value that is not wrong. If we require that the value b decided by f in $\vec{x}$ appear at least $t+1$ in $\vec{x}$, then b is certainly not wrong. A system satisfying this requirement is denoted EV, formally, as follows.

Definition 2.2 (exact value system). *A system* D *is* exact value *if for any input vector $\vec{x}$ and any system state s, if $f(\vec{x}, s) = b$, then $\#b(\vec{x}) \geq t+1$.*

This is the natural generalization to the requirement of the binary case [8]. In the case of a multivalued consensus system, we may consider also a different requirement: that if the value b is decided by f, then there are at least $t+1$ values greater than or equal to b in $\vec{x}$ and at least $t+1$ values less than or equal to b in $\vec{x}$. Hence, the decided value is between two correct values; i.e., there are at least *two* sensors, p_i and p_j, such that their inputs, $\vec{x}[i]$ and $\vec{x}[j]$, satisfy $\vec{x}[i] \leq b \leq \vec{x}[j]$. Such a system is denoted RV, as follows.

Definition 2.3 (range value system). *A system* D *is* range value *if for any input vector $\vec{x}$ and any system state s, if $f(\vec{x}, s) = b$, then $\sum_{b' \geq b} \#b'(\vec{x}) \geq t+1$ and $\sum_{b' \leq b} \#b'(\vec{x}) \geq t+1$.*

These constraints ensure that if b is the consensus value, then there are at least two nonfaulty sensors, p_i and p_j, such that their inputs, $\vec{x}[i]$ and $\vec{x}[j]$, satisfy $\vec{x}[i] \leq b \leq \vec{x}[j]$. We use RV to denote a range value consensus system.

Note that any EV system is also an RV system. For binary input systems (where $|V| = 2$), an RV system is also an EV system.

2.2. The scope for exact and range value systems. Before we continue describing our results, we need to describe the combination of the parameters n, v, and t for which an exact or a range value system exists. First we show that for any n, v, t for which $n \geq vt+1$, an exact value system can be defined, and for every such defined EV it holds that $n \geq vt+1$. This result generalizes the binary case of [8], in which the analogous condition is $n \geq 2t+1$.

LEMMA 2.4. *An exact value system* EV *can be defined if and only if $n \geq vt+1$.*

Proof. Consider an exact value system EV $= \langle n, v, t, S, \tau, f\rangle$, and assume for contradiction that $n \leq vt$. Then there exists a vector $\vec{x} = 0^{\alpha_0}1^{\alpha_1}\cdots(v-1)^{\alpha_{v-1}}$, with $\alpha_i \leq t$ for every i, since $n \leq vt$; that is, $\#b(\vec{x}) < t+1$ for every $b \in V$. Taking $\vec{x}$ as the first input vector of a path to EV, Definition 2.2 of an exact value system requires that $f(\vec{x}, s_0)$ be equal to some value b with $\#b(\vec{x}) \geq t+1$, a contradiction.

Now, assume that $n \geq vt+1$. Then, every input vector $\vec{x}$ must have at least one value b with $\#b(\vec{x}) \geq t+1$ (otherwise $n \leq vt$). Define an exact value system EV $= \langle n, v, t, S, \tau, f\rangle$ with $S = \{0\}$, $\tau(\cdot, 0) = 0$, and f choosing such a value b, e.g., $f(\vec{x}, 0) = b$ such that b is the smallest value with $\#b(\vec{x}) \geq t+1$. □

For range value systems, RV must satisfy $n \geq 2t+1$, and if $n \geq 2t+1$, then an RV system can be defined, as follows.

LEMMA 2.5. *A range value system* RV *can be defined if and only if $n \geq 2t+1$, with $v > 1$.*

Proof. Consider a range value system RV $= \langle n, v, t, S, \tau, f\rangle$, and assume for contradiction that $n \leq 2t$. Consider the input vector $\vec{x} = 0^t1^{n-t}$. For any $b \in \{0, 1, \ldots, v-1\}$, either $\sum_{b' \geq b} \#b'(\vec{x}) < t+1$ or $\sum_{b' \leq b} \#b'(\vec{x}) < t+1$. Taking $\vec{x}$ as the first input vector of a path to RV, Definition 2.3 of a range value system requires that $f(\vec{x}, s_0)$ be equal to some value b with $\sum_{b' \geq b} \#b'(\vec{x}) \geq t+1$ and $\sum_{b' \leq b} \#b'(\vec{x}) \geq t+1$, a contradiction.

Now, assume that $n \geq 2t+1$. Then, every input vector $\vec{x}$ has a value b such that $\sum_{b' \geq b} \#b'(\vec{x}) \geq t+1$ and $\sum_{b' \leq b} \#b'(\vec{x}) \geq t+1$. To see this consider the distribution of values of $\vec{x}$, $\#i(\vec{x}) = \alpha_i$, for $0 \leq i \leq v-1$, and let $\vec{y}$ be the lexicographically ordered input values of $\vec{x}$:

$$\vec{y} = 0^{\alpha_0}1^{\alpha_1}\cdots(v-1)^{\alpha_{v-1}}.$$

Then, if b is the value in the middle of this sequence, meaning $b = \vec{y}[\lceil \frac{n}{2} \rceil]$, we get the desired property $\sum_{b' \geq b} \#b'(\vec{x}) \geq t+1$ and $\sum_{b' \leq b} \#b'(\vec{x}) \geq t+1$, since $n \geq 2t+1$. And we can define an exact value system EV $= \langle n, v, t, S, \tau, f\rangle$ with $S = \{0\}$, $\tau(\cdot, 0) = 0$, and f, choosing a value b, so that $f(\vec{x}, 0) = b$. □

3. Stability of systems with memory.

In section 3.1 we study the instability of an EV or RV system with the minimal number of sensors according to Lemmas 2.4 and 2.5. We also show that when the number of sensors is big enough, an EV system can be defined whose instability is 1. The case of a general n is considered in section 3.2. Then, in section 4, we study a system that cannot keep any memory from previous input vectors, and from its previous decisions.

It is convenient to visualize the input vector $\vec{x}$ as a bar chart, where every bar relates to a possible input value b, and the height of that bar is $\#b\,(\vec{x})$. By definition, an input value may be the consensus value of an EV system only if the height of the bar is at least $t+1$. This is exemplified in Figure 3.1.

3.1. The extreme cases of n.

We will first consider the extreme case in which n is the minimal possible number of sensors: $vt+1$ for EV systems, as shown in Lemma 2.4, and $2t+1$ for RV systems, as shown in Lemma 2.5. We prove that the instability of the system in this case is n; that is, it is possible to force any system to

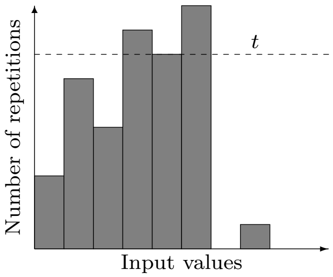


FIG. 3.1. *A visualization of the input vector as a bar chart—every bar represents an input value, and its height is the number of repetitions of that value within the input vector. The dashed line represents height of* t*. Notice that only two input values are a valid consensus value.*

change its decision with each input change. The second extremum is where n is very large. In Lemma 3.4 we will show that when n is large enough, we can build an EV system that changes its decision only once.

LEMMA 3.1. *For every* EV *system* D *with* $n = vt + 1$*, instability*(D) $= vt + 1$.

Proof. Obviously *instability*(D) $\leq n = vt+1$, since any geodesic path is of length at most n, and hence at most n decision value changes can occur.

To prove that *instability*(D) $= n = vt + 1$, we construct a geodesic path of length $vt + 1$ that produces $vt + 1$ changes in system D. We will start with the input vector $\vec{x}_0 = (01 \dots (v-1))^t 0$, and to get $\vec{x}_{i+1}$, switch the input value of the $(i + 1)$th sensor from i mod v to $i + 1$ mod v. The geodesic path can be illustrated as follows:

$$\begin{array}{ll}
(012\dots(v-2)(v-1))^1(012\dots(v-1))^{t-1}0 & \to \\
(112\dots(v-2)(v-1))^1(012\dots(v-1))^{t-1}0 & \to \\
(122\dots(v-2)(v-1))^1(012\dots(v-1))^{t-1}0 & \to \\
(123\dots(v-2)(v-1))^1(012\dots(v-1))^{t-1}0 & \to \\
\dots & \to \\
(123\dots(v-1)(v-1))^1(012\dots(v-1))^{t-1}0 & \to \\
(123\dots(v-1)0)^1(01\dots(v-1))^{t-1}0 & \to \\
\dots & \to \\
(123\dots(v-1)0)^2(01\dots(v-1))^{t-2}0 & \to \\
\dots & \to \\
(123\dots(v-1)0)^t0 & \to \\
(123\dots(v-1)0)^t1. &
\end{array}$$

Notice that $\#b(\vec{x_0}) = t$ for $b \neq 0$ and $\#0(\vec{x_0}) = t + 1$. Since the only input value that appears more than t times is 0, the consensus value must be 0. In general, a simple induction shows that $\#b(\vec{x}_i) = t$ for $b \neq i$ mod v and $\#b(\vec{x}_i) = t + 1$ for $b = i$ mod v. Therefore, the consensus value in $\vec{x}_i$ is i mod v, and there is a consensus value change with each new input vector; i.e., *instability*(D) $= n$. □

The previous result generalizes the result in [8], where it was proved that when $n = 2t + 1$ and $v = 2$ the instability of the system is n.

LEMMA 3.2. *For every* RV *system* D *with* $n = 2t+1$ *and any* $v \geq 2$*, instability*(D) $= 2t + 1$.

Proof. Consider the system D over the restricted set of input values $\{0, 1\} \subseteq V$. When restricted to binary input values, any RV system is also an EV system. Since here $n = 2t + 1$, then by Lemma 3.1 *instability*(D) $= 2t + 1$. □

It is easy to see that, for any EV system D with $v > 1$, there is a geodesic path that forces it to make at least one decision change. This, actually, holds for any continuous consensus system D, as follows.

LEMMA 3.3. *For every system* D *with* $v > 1$*, it holds that instability*(D) ≥ 1.

Proof. Consider a geodesic path starting in $\vec{x}_0 = 0^n$ and ending in $\vec{x}_n = 1^n$ obtained by switching one value at a time from 0 to 1. By the definition of a continuous consensus system, $f(\vec{x}_0, s_0) \in \{\vec{x}[i] \mid 1 \leq i \leq n\} = \{0\}$, and therefore $f(\vec{x}_0, s_0) = 0$. For symmetric reasons $f(\vec{x}_n, s_n) = 1$. Thus, there is at least one change to the consensus value. □

In Lemma 3.1 we considered the instability of an EV system for the smallest possible value of n, namely $n = vt + 1$ (Lemma 2.4 proved that this is the smallest value), and showed that the instability is the largest possible, namely n. There is no upper bound for the number of sensors, since for every $n \geq vt + 1$ there exists an exact value system. However, we prove next that for $n \geq v^2 t + 1$ there exists a system with the smallest possible instability, namely 1, matching the trivial lower bound of Lemma 3.3. To prove it, we describe a simple system, MS. Also, we show that for $n < v^2 t + 1$ no system can achieve an instability of 1.

For its initial state, the system MS decides on the most frequent value in $\vec{x}_0$. The system MS remembers in its state its last consensus value, c, and a vector $\vec{h}$ that records in $\vec{h}[b]$, for each $b \in V$, how many times an input value has switched to b. Then, MS will keep its previous decision, c, unless forced to change; that is, it will change decision only if $\#c(\vec{x}) = t$ in the current input vector $\vec{x}$. In this case, if there is a value b such that $\vec{h}[b] \geq t + 1$, it will decide b, since then no more changes will occur. Otherwise, it will decide on the most frequent value in $\vec{x}$ (if there is more than one most frequent value, it will decide on the smallest one).

To compute $\vec{h}$ the system must remember also the previous input vector; actually it must remember only the multiplicity of each input value in the previous input vector, and not the position in the vector of each value. Thus, for a vector $\vec{x}$, let $bar(\vec{x})$ be the vector of dimension v such that $bar(\vec{x})[b]$ is equal to the number of times $\vec{x}[i] = b$, over $1 \leq i \leq n$. This is illustrated in Figure 3.1. Therefore, the current state is of the form $\tilde{s} = \{bar(\vec{x}_{prev}), c, \vec{h}\}$. For example, if the current state is $\tilde{s} = \{(4, 5, 1), 1, (2, 0, 1)\}$, then the input vector contains four inputs with value 0, five inputs with value 1, and one input with value 2; the consensus value is 1; and two input values were changed to 0 and one input to 2 in the input path $\vec{x}_0 \to \vec{x}_1 \to \vec{x}_2 \to \vec{x}_3$ (since there have been three changes to the input values, the path is of length 3). Formally, the system is defined as follows, where $\tilde{s} = \{bar(\vec{x}_{prev}), c, \vec{h}\}$:

$$f(\vec{x}, \tilde{s}) = \begin{cases} \operatorname{maxarg}_{b \in V} \{\#b(\vec{x})\} & \text{if } \tilde{s} = s_0, \\ \begin{cases} \operatorname{maxarg}_{b \in V} \left\{\vec{h}[b]\right\} & \text{if } \exists b,\ \vec{h}[b] \geq t+1, \\ \operatorname{maxarg}_{b \in V} \{\#b(\vec{x})\} & \text{otherwise,} \end{cases} & \\ & \text{if } \#c(\vec{x}) \leq t, \\ c & \text{if } \#c(\vec{x}) \geq t+1, \end{cases}$$

$$\tau(\vec{x}, \tilde{s}) = \begin{cases} \{bar(\vec{x}), f(\vec{x}, \tilde{s}), (0, \dots, 0)\} & \text{if } \tilde{s} = s_0, \\ \left\{bar(\vec{x}), f(\vec{x}, \tilde{s}), \vec{h} + \sum_{b=0}^{v-1} \vec{\delta}_b \max\{0, bar(\vec{x})[b] - bar(\vec{x}_{prev})[b]\}\right\} & \\ & \text{otherwise,} \end{cases}$$

where

$$\vec{\delta}_m[k] = \begin{cases} 1, & m = k, \\ 0, & m \neq k, \end{cases}$$

$$\operatorname*{maxarg}_{b\in V}\{g(b)\} = \min\{b \mid b \in V,\, g(b) = \max\{g(b) \mid b \in V\}\}\,.$$

This system is a generalization of the system BP, described in [8]. We will prove that for a large enough number of sensors, the instability of this system is 1.

LEMMA 3.4. *When $n \geq v^2t+1$, instability*(MS) $= 1$.

Proof. Lemma 3.3 implies that *instability*(MS) ≥ 1. Therefore, we will prove that *instability*(MS) ≤ 1 when $n \geq v^2t+1$. We will do so by proving that MS changes value at most once in any geodesic path.

Consider any geodesic path $\vec{x}_0 \to \vec{x}_1 \to \cdots$. We already know that for some value b_0, $\#b_0(\vec{x}_0) \geq vt+1$; otherwise for every $b \in V$, $\#b(\vec{x}_0) \leq vt$ and $n \leq v^2t$, a contradiction. Now, assume that b_0 is the one which appears the largest number of times in $\vec{x}_0$. In particular, $\#b_0(\vec{x}_0) \geq vt+1$. Then, $f(\vec{x}_0, s_0) = b_0$. Notice that f will not change its decision before it gets an input vector with t entries equal to b_0. Consider the first time this happens (otherwise no consensus value changes occur, and we are done), say in $\vec{x}_i$. It follows that at least $(v-1)t+1$ entries equal to b_0 have changed to other values, in $\vec{x}_0 \to \vec{x}_1 \to \cdots \to \vec{x}_i$, since $\#b_0(\vec{x}_i) = t$ and $\#b_0(\vec{x}_0) \geq vt+1$. Each time a value changes from b_0 to some other value, b, one unit is added to $\vec{h}[b]$. Therefore, after $(v-1)t+1$ entries equal to b_0 have changed to other values, at least one entry of $\vec{h}$, say c, gets incremented at least $t+1$ times (there are $v-1$ values different from b_0); that is, $\vec{h}[c] \geq t+1$ at the end of the path $\vec{x}_0 \to \vec{x}_1 \to \cdots \to \vec{x}_i$. Then f will decide on such a value c with $\vec{h}[c] \geq t+1$. No more decision changes will occur, because there are $t+1$ entries in the input vector that are equal to c and have already changed value in the past, and the path is geodesic. □

3.2. The general case. We will call a vector such as $\vec{h}$ in the system MS an *accumulator vector*. Notice that for any geodesic input path P we can define an accumulator vector associated with each of the vectors of P. The accumulator records the number of times each value has been changed. That is, initially $\vec{h} = 0^v$, and if one input bit changes from some value b to b', then $\vec{h}[b']$ is incremented by 1. Thus, an accumulator vector is independent of the EV system used and is defined even if the system does not actually use it; it is just a device that we use to reason about the system. The following lemma captures the importance of $\vec{h}$, and it will be useful for proving lower bounds. Here by *optimal* we mean that the system delays a change to the consensus value while possible. More formally, if the consensus value is b, then an optimal system will not change the consensus value until $\#b(\vec{x}) \leq t$. Notice that for any nonoptimal system D there exists an optimal system D$'$ which holds *instability*(D$'$) $\leq$ *instability*(D), i.e., a system which behaves exactly like D except for delaying a change to the consensus value.

LEMMA 3.5. *Let* D *be an optimal* EV *system. Consider an input vector $\vec{x}$ at the end of a geodesic input path P, and the corresponding vector $\vec{h}$. Assume that the consensus value in $\vec{x}$ is b. There is a geodesic path extending P that causes at least one consensus change to* D *if and only if $\vec{h}[b] \leq t$.*

Proof. If $\vec{h}[b] \leq t$, then at most t entries of $\vec{x}$ that are equal to b have already changed in P. The total number of entries equal to b is $\#b(\vec{x}) \geq t+1$, since b is the decision in $\vec{x}$. So we can extend P by switching one by one entries that are equal to b to some other value until we get an input vector $\vec{x}_1$ with $\#b(\vec{x}_1) = t$, and at some point during this path the decision will have to change.

If there is a geodesic path extending P that causes at least one consensus change to D, then clearly $\vec{h}[b] \leq t$, because otherwise at least $t+1$ entries of $\vec{x}$ that are equal to b have already changed in P, and hence a vector $\vec{x}_1$ in any geodesic extension to

P will have $\#b(\vec{x}_1) \geq t+1$, and D does not need to change its decision ever (since it is optimal). ◻

LEMMA 3.6. *For any* EV *system* D, *if* $n \leq v^2 t$, *then instability*(D) ≥ 2.

Proof. We prove that for any exact value system D there is a geodesic path that causes two decision changes. Let $\vec{x}_0 = 0^{\alpha_0} 1^{\alpha_1} \cdots b_i^{\alpha_i} \cdots (v-1)^{\alpha_{v-1}}$, with $\alpha_i \leq vt$ for all $0 \leq i \leq v-1$. Assume $f(\vec{x}_0, s_0) = b_j$ for some j. Then we produce a geodesic path starting in $\vec{x}_0$, by switching one by one $\alpha_j - t \leq (v-1)t$ inputs that are equal to b_j to other values, in a round-robin manner (first to 0, then to 1, ..., then to $v-1$, then to 0, and so on). Eventually, a decision change must occur, at the latest when there are only t entries equal to b_j. Let us call $\vec{x}_{l_1}$ the input vector when a decision change occurs for the first time, and $\vec{h}_{l_1}$ the corresponding accumulator. Then,

$$bar(\vec{x}_{l_1}) = (\alpha_0 + \beta_0, \alpha_1 + \beta_1, \ldots, t + \Delta, \ldots, \alpha_{v-1} + \beta_{v-1}),$$

$$\vec{h}_{l_1} = (\beta_0, \beta_1, \ldots, 0, \ldots, \beta_{v-1}),$$

where $\sum_{i=0,\, i \neq j}^{v-1} \beta_i \leq \alpha_j - t$ and $\Delta \geq 0$. It follows from Lemma 3.5 that whatever the consensus decision is in $\vec{x}_{l_1}$, say b_k ($\neq b_j$), we can continue changing input values, from b_k to other values until we get an input vector with t entries equal to b_k, and force one more decision change. This is since $\vec{h}_{l_1}[k] = \beta_k \leq \lceil \frac{\alpha_j - t}{v-1} \rceil \leq t$. ◻

A concept similar to the accumulator vector is the *consensus accumulator vector*. Such a vector holds for each input value b the number of input values changed *from a consensus value* to b. Notice that if $\vec{h}$ is an accumulator vector and $\vec{\bar{h}}$ is a consensus accumulator vector of the same path, then for any b it holds that $\vec{\bar{h}}[b] \leq \vec{h}[b]$.

We now prove our main upper bound result for EV systems with memory. For any vector $\vec{y}$, define $|\vec{y}|$ to be the $\sum_i \vec{y}[i]$, and $\max \vec{y}$ to be $\max_i \vec{y}[i]$. We will use the fact that $|\vec{h}| \geq vt+1$ implies $\max \vec{h} \geq t+1$, and then apply Lemma 3.5.

LEMMA 3.7. *Let* $x = \lceil \frac{n-vt}{v-1} \rceil$ *and* $y = \lceil \frac{n-vt}{v} \rceil$. *Then instability*(MS) $\leq \lfloor \frac{vt-y}{x} \rfloor + 2$.

Proof. Consider a geodesic path $P = \vec{x}_0 \to \vec{x}_1 \to \cdots$. We will first claim that $\max bar(\vec{x}_0) \geq y+t$. Otherwise,

$$n = |bar(\vec{x}_0)| = \sum_{b \in V} bar(\vec{x}_0)[b] \leq v(y+t-1) = v\left(\left\lceil \frac{n-vt}{v} \right\rceil - 1\right) + vt < (n-vt) + vt = n.$$

Therefore, MS will decide in $\vec{x}_0$ on a value b such that $bar(\vec{x}_0)[b] \geq y+t$. Let $\vec{x}_{i_1}$ be the first vector in P where MS changes its decision, and $\vec{\bar{h}}_{i_1}$ the corresponding consensus accumulator. Since MS changes its decision when only t entries are equal to b, it follows that at least y inputs must have changed to get to $\vec{x}_{i_1}$. That is,

$$|\vec{\bar{h}}_{i_1}| \geq y.$$

We will now claim that $\max bar(\vec{x}_{i_1}) \geq x+t$. Otherwise, since $\#b(\vec{x}_{i_1}) = t$, then

$$\begin{aligned} n = |bar(\vec{x}_{i_1})| &= \sum_{b' \in V} bar(\vec{x}_{i_1})[b'] \leq t + (v-1)(x+t-1) \\ &= t + (v-1)\left(\left\lceil \frac{n-vt}{v-1} \right\rceil + t - 1\right) \\ &< t + (n-vt) + (v-1)t = n. \end{aligned}$$

In general, consider the sequence of input vectors at which the subsequent consensus value changes occur, $\vec{x}_{i_2}, \vec{x}_{i_3}, \ldots$, with the corresponding consensus accumulators,

$\vec{h}_{i_2}, \vec{h}_{i_3}, \ldots$. Notice that in each such vector $\vec{x}_{i_j}$ there is at least one value b such that $\#b(\vec{x}_{i_j}) = t$, namely, b is the consensus value in $\vec{x}_{i_{j-1}}$. Thus, $\max bar(\vec{x}_{i_j}) \geq x + t$, and hence MS will decide on a value d in $\vec{x}_{i_j}$ such that $bar(\vec{x}_{i_j})[d] \geq x + t$. Thus, at least x entries equal to d must change to get to $\vec{x}_{i_{j+1}}$:

$$|\vec{h}_{i_{j+1}}| \geq |\vec{h}_{i_j}| + x \geq y + jx,$$

$$\max bar(\vec{x}_{i_j}) \geq x + t.$$

For any j, let b_j denote the consensus value at $\vec{x}_{i_j}$. Let $\vec{h}_{i_{c+1}}$ be the consensus accumulator vector after the last change to the consensus value, and let $\vec{h}_{i_c}$ be the consensus accumulator after the *previous* consensus change. If there is a value $b \in V$ for which $\vec{h}_{i_c}[b] > t$, then by the definition of MS the consensus value would change to such a value, and by Lemma 3.5 there would not be a change of the consensus value from b_c to b_{c+1}. Therefore, for all $b \in V$, $\vec{h}_{i_c}[b] > t$, and so $|\vec{h}_{i_c}| \leq vt$.

Since

$$y + (c-1)x \leq |\vec{h}_{i_c}| \leq |\vec{h}_{i_c}| \leq vt$$

we have that

$$c \leq \left\lfloor \frac{vt - y}{x} \right\rfloor + 1.$$

Thus the total number of consensus changes is at most $c + 1$ (one change from $\vec{x}_0$ to $\vec{x}_{i_1}$, and c changes after that), and we get the desired result *instability*(MS) $\leq \lfloor \frac{vt-y}{x} \rfloor + 2$. □

Consider the extreme cases for the previous lemma. First, let $n = v^2 t$, so we have $x = vt$ and $y = t(v-1)$. We get *instability*(MS) $\leq \lfloor \frac{vt - t(v-1)}{vt} \rfloor + 2$, and hence *instability*(MS) $\leq \lfloor \frac{1}{v} \rfloor + 2 = 2$, which matches the lower bound of Lemma 3.6. Second, let $n = vt + 1$, so we have $x = 1$, $y = 1$. We get *instability*(MS) $\leq \lfloor \frac{vt-1}{1} \rfloor + 2 = vt + 1$, which matches the lower bound of Lemma 3.1.

Summarizing the stability results for EV. The next theorem summarizes the results obtained for EV systems with memory. Let *instability*(EV) denote the smallest instability over all EV systems.

THEOREM 3.8.

1. *When* $vt + 1 < n < v^2 t$, *instability*(EV) $\leq \lfloor \frac{vt-y}{x} \rfloor + 2$, *where* $x = \lceil \frac{n-vt}{v-1} \rceil$ *and* $y = \lceil \frac{n-vt}{v} \rceil$.
2. *Otherwise,*

$$\textit{instability}\,(\text{EV}) = \begin{cases} vt+1 & \textit{if } n = vt+1, \\ 2 & \textit{if } n = v^2 t, \\ 1 & \textit{if } n \geq v^2 t + 1. \end{cases}$$

4. Stability of symmetric memoryless systems. In this section we turn to investigating the case in which no memory is kept by the system from previous input vectors and from previous decisions; that is, we assume a *memoryless* system D $= \langle n, v, t, S, \tau, f \rangle$, where $S = \{s_0\}$. Hence instead of $f(\vec{x}, s_0)$ we simply write $f(\vec{x})$ for any input vector $\vec{x}$. Furthermore, we assume that the system is *symmetric*, meaning that the decision function is oblivious to the order of the input values; i.e., for any two input vectors $\vec{x}, \vec{y}$, $bar(\vec{x}) = bar(\vec{y})$ implies $f(\vec{x}) = f(\vec{y})$. Thus, f is a function of just $\#i(\vec{x})$ for every $i \in V$. We study in this section the corresponding function $\tilde{f}$ defined on the set A,

$$A = \{(\alpha_0, \alpha_1, \ldots, \alpha_{v-1}) \mid \alpha_i \in \mathbb{N} \cup \{0\},\ \alpha_0 + \alpha_1 + \cdots + \alpha_{v-1} = n\}\,.$$

We also define $\tilde{f} : A \to V$ in terms of f as follows:

$$\tilde{f}(\alpha_0, \alpha_1, \ldots, \alpha_{v-1}) = f(0^{\alpha_0} 1^{\alpha_1} \cdots (v-1)^{\alpha_{v-1}}).$$

4.1. Overview of the lower bound. The lower bound proof is a generalization of the one in [10], from dimension 1 to higher dimensions. The proof uses Sperner's lemma, a well-known result that generalizes the graph connectivity notion to higher dimensions. See, e.g., [4] for a proof, and see, e.g., [1] for a recent application to distributed computing.

In dimension 1, Sperner's lemma says that if the vertices of a path are colored with $0, 1$, with 0 coloring one end-vertex and 1 coloring the other, then at least one edge will have its two vertices colored with different colors. More generally, if the vertices of a connected graph are colored with 0, 1 such that at least one vertex is colored with 0 and at least one vertex is colored with 1, then at least one edge of the graph will have its two vertices colored with different colors.

In general, a *k-simplex* is a set of $k+1$ vertices; the *dimension* k of the simplex is equal to its number of vertices minus one. For $j \leq k$, a j-face of a k-simplex is a subset of $j+1$ of its vertices. A 0-face is just a vertex. A *complex* is a set of simplexes closed under containment. It is often convenient to assume that these vertices are points in space and that the simplex is the convex hull of these points. A *subdivision* of a simplex is a partition of the simplex into simplices, such that the union of the dividing simplices equals the original simplex, and any two dividing simplices intersect at most in a common face. A *Sperner coloring* of a subdivided k-simplex $\mathcal{S}$ is an assignment of colors $\{0, \ldots, k\}$ to its vertices such that the $k+1$ corners are assigned the $k+1$ distinct colors, and the vertices in the subdivision of a face of $\mathcal{S}$ are assigned colors from the corners of that face.

In our proof we put the input vectors of A in a space of dimension v. The input vectors in which one component is n and all the rest are zero form a simplex, in fact a $(v-1)$-simplex. The points corresponding to all other input vectors are convex combinations of the above vertices and therefore reside within the simplex. We use Sperner's lemma to conclude that there is a dividing simplex such that its v vertices have v distinct decision values.

There are several ways to define such a subdivision, and fortunately we found in [9] a subdivision that serves us well, in that for every dividing subsimplex of dimension $v-1$ there is a geodesic path of length vt whose input vectors correspond to the vertices of the subsimplex. In particular, we have such a path in the dividing simplex that has different function values for each of its vertices.

We illustrate the instability lower bound for the case of $v = 3$. This should give the reader a more intuitive view of the general proof.

When $v = 3$, the set A can be used as vertices of a subdivision of a 2-simplex—or, in other words, a solid flat triangle. This triangle can be divided into subsimplices (subtriangles) such that the vertices of a subtriangle represent input vectors that are adjacent, in the sense that they can result from each other by a change of a single input value. The vertices will be colored by the consensus value of a given input vector, and using Sperner's lemma, we will find a subtriangle for which the colors of

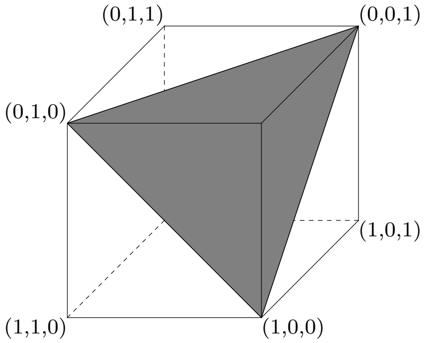


FIG. 4.1. *A* 2*-simplex defined over* $(1,0,0)$*,* $(0,1,0)$*, and* $(0,0,1)$*.*

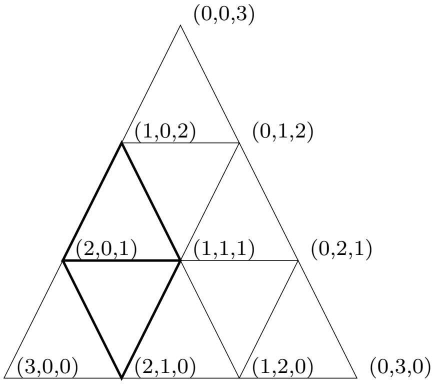


FIG. 4.2. *Subdivision of a* 2*-simplex—the emphasized subsimplices represent the two types of subsimplices.*

its vertices are pairwise different. This subtriangle will imply a geodesic path with $3t+1$ changes to the consensus value.

LEMMA 4.1. *Let* D *be a symmetric memoryless* EV *system with* $v = 3$*. Then* *instability*(D) $\geq 3t+1$.

Proof. Every point of A is in the 2-simplex $\mathcal{S}$, defined by the vectors $(n,0,0)$, $(0,n,0)$, and $(0,0,n)$. A k-simplex is defined by $k+1$ linearly independent vectors, $\vec{v}_0,\ldots,\vec{v}_k$, and is defined as $\left\{\vec{x} = \sum_{i=0}^{k} \lambda_i \vec{v}_i \;\middle|\; \lambda_i \geq 0,\ \sum_{i=0}^{k} \lambda_i = 1\right\}$. Figure 4.1 describes a 2-simplex.

We will use the points of A to subdivide $\mathcal{S}$. Every subsimplex in the subdivision will be defined by $\vec{a}_0, \vec{a}_1, \vec{a}_2 \in A$, where either $\vec{a}_0-(1,0,0) = \vec{a}_1-(0,1,0) = \vec{a}_2-(0,0,1)$ or $\vec{a}_0+(1,0,0) = \vec{a}_1+(0,1,0) = \vec{a}_2+(0,0,1)$. For example, the subsimplex defined by $(2,0,1)$, $(1,1,1)$, and $(1,0,2)$ is of the first type, whereas the simplex defined by $(1,1,1)$, $(2,0,1)$, and $(2,1,0)$ is of the second type (see Figure 4.2). The subdivision that corresponds to the case where $n=3$ is demonstrated in Figure 4.2. We will color every vertex $\vec{a}$ with the color $\tilde{f}(\vec{a})$. Since $\#i(\vec{a}) = 0$ implies that $f(\vec{a}) \neq i$, the coloring is a Sperner coloring, and hence there must exist a subsimplex in the subdivision so that every vertex of the subsimplex is colored by a different color.

Let $\vec{a}_0$, $\vec{a}_1$, and $\vec{a}_2$ be the vertices of that subsimplex. Let j_i be defined by $f(\vec{a}_i) = j_i$, $i = 0,1,2$. Then $\#j_i(\vec{a}_i) > t$. Since the subsimplex is of one of the two types above, for any pair $\vec{a}_k, \vec{a}_l$ we have $\#b(\vec{a}_l) - 1 \leq \#b(\vec{a}_k) \leq \#b(\vec{a}_l) + 1$ for all b.

Hence for every $k, b \in \{0,1,2\}$, $\#b(\vec{a}_k) \geq t$.

We will examine the case where the given subsimplex is of the first type. We say that an input vector $\vec{x}$ *corresponds* to vector $\vec{a} \in A$ if and only if $bar(\vec{x}) = \vec{a}$, namely $\#i(\vec{x}) = \vec{a}[i]$ for any $0 \leq i \leq v-1$. Assume, w.l.o.g., $\vec{a}_0 - (1,0,0) = \vec{a}_1 - (0,1,0) = \vec{a}_2 - (0,0,1)$. Thus, $\vec{a}_0 + (-1,1,0) = \vec{a}_1$, which means that a vector of n input values of the sensors that corresponds to $\vec{a}_1$ can be reached from a vector of n input values of the sensors that corresponds to $\vec{a}_0$ by switching the input value of one sensor from 0 to 1. Generally, if the input vector $\vec{x}$ corresponds to $\vec{a}_i$, then changing the input value of one sensor from i to $(i+1) \bmod 3$ will change the value of the decision function and result in a new input vector that corresponds to $\vec{a}_{(i+1) \bmod 3}$.

Since for any $k, b \in \{0,1,2\}$ it holds that $\#b(\vec{a}_k) \geq t$, then a vector corresponding to $\vec{a}_0$ will be some permutation of $(012)^t\vec{z}$. We can examine a geodesic path which starts with $(012)^t\vec{z}$, which corresponds to $\vec{a}_0$. The path continues for the next $3t$ steps, at every step changing the value of the ith sensor from $i \bmod 3$ into $(i+1) \bmod 3$, thus achieving $3t$ value changes. The path ends with the input vector $(120)^t\vec{z}$. We can continue the geodesic path to $(120)^t k^{n-3t}$, where $f((120)^t\vec{z}) \neq k$, thus yielding $3t+1$ value changes.

The case where the simplex is of the second kind is analogous. □

4.2. Lower bound for memoryless systems. We will now prove that for any EV system D with a symmetric function f, *instability*(D) $\geq vt+1$. To do so, we will use the *edgewise subdivision of a simplex*, defined in [9] (see the appendix for details).

Let $\mathcal{S}$ be a d-simplex, spanned by $\vec{V}_0, \vec{V}_1, \ldots, \vec{V}_d$. An edgewise subdivision is a function that, given an integer k, transforms every point $\vec{X} \in \mathcal{S}$ into a color scheme M, which is defined by a matrix as follows:

$$M = \begin{pmatrix} \chi_{1,0} & \chi_{1,1} & \cdots & \chi_{1,j} \\ \chi_{2,0} & \chi_{2,1} & \cdots & \chi_{2,j} \\ \vdots & \vdots & \ddots & \vdots \\ \chi_{k,0} & \chi_{k,1} & \cdots & \chi_{k,j} \end{pmatrix},$$

where $j \leq d$. Each entry of the matrix is a nonnegative integer from 0 through d, the columns are pairwise different, and the entries appear in nondecreasing order when read like English text:

$$\chi_{1,0} \leq \chi_{1,1} \leq \cdots \leq \chi_{1,j} \leq \chi_{2,0} \leq \cdots \leq \chi_{k,j}.$$

The color scheme defines $j+1$ independent vectors $\vec{V}_0^*, \vec{V}_1^*, \ldots, \vec{V}_j^*$, where $\vec{V}_l^* = \frac{1}{k}\sum_{i=1}^{k} \vec{V}_{\chi_{i,l}}$, which span a j-simplex. By applying the function to every point $\vec{X} \in \mathcal{S}$, we obtain a subdivision of $\mathcal{S}$ into subsimplices, some of which are d-simplices. [9] proves that the above simplices indeed form a subdivision of $\mathcal{S}$, and that every point $\vec{X}$ resides within the simplex it defines.

Recall that

$$\vec{\delta}_m[k] = \begin{cases} 1, & m = k, \\ 0, & m \neq k. \end{cases}$$

LEMMA 4.2. *Let $\mathcal{S}$ be the $(v-1)$-simplex, spanned by $\vec{V}_0, \vec{V}_1, \ldots, \vec{V}_{v-1}$, where $\vec{V}_i = n \cdot \vec{\delta}_i$. Let $\vec{a}$ be a vertex of any $(v-1)$-simplex in the edgewise subdivision of $\mathcal{S}$ using $k = n$. Then for every $0 \leq i \leq v-1$, $\vec{a}[i]$ is an integer. Moreover, $\sum_{i \in V} \vec{a}[i] = n$.*

Proof. Let $\vec{a}$ correspond to a column j in some color scheme M. Then,

$$\vec{a}\left[i\right] = \frac{1}{n}\sum_{l=1}^{n} \vec{V}_{\chi_{l,j}}\left[i\right] = \frac{1}{n}\sum_{l=1}^{n} n \cdot \vec{\delta}_{\chi_{l,j}}\left[i\right] = \sum_{l=1}^{n} \vec{\delta}_{\chi_{l,j}}\left[i\right].$$

Since $\vec{\delta_l}\left[m\right]$ is always an integer, then $\vec{a}\left[i\right]$ is also an integer. Finally,

$$\sum_{i \in V} \vec{a}\left[i\right] = \sum_{i \in V}\sum_{l=1}^{n} \vec{\delta}_{\chi_{l,j}}\left[i\right] = n. \qquad \square$$

Lemma 4.2 implies that for every vertex $\vec{a}$ of a $(v-1)$-simplex of the edgewise n-subdivision of $\mathcal{S}$, $\vec{a} \in A$. We will color every vertex $\vec{a}$ of the subdivision with $\tilde{f}(\vec{a})$. Since it holds that for every $\vec{a} \in A$ such that $\tilde{f}(\vec{a}) = b$, $\#b(\vec{a}) \geq t+1 > 0$, then the coloring is a Sperner coloring, and according to Sperner's lemma there must exist in the subdivision a subsimplex $\mathcal{S}^*$ such that all its vertices' colors are different.

We will now introduce the concept of a geodesic path over a subsimplex. A path over a subsimplex is a sequence of vertices of the subsimplex. The elements of a vertex are its coordinates at a point in A. A geodesic path over a subsimplex is a path in which any element is *increased* at most once. We refer only to increments, since we would like to allow a geodesic loop—a path which begins and ends at the same vertex. A geodesic path with minimal changes is a geodesic path for which two consecutive vertices differ only in two entries: one increased by 1, the other decreased by 1.

Notice that the above definition does not ensure that a geodesic path with minimal changes over a subsimplex will necessarily correspond to a geodesic path of input values: For example, the geodesic path $(5,0,0) \to (4,1,0) \to (4,0,1)$ does not correspond to any geodesic path of input values. However, when all the entries are positive there is always a corresponding geodesic path of input values.

Lemma 4.3. *Let $\mathcal{S}$ be a d-simplex spanned by $\vec{V_0}, \vec{V_1}, \ldots, \vec{V_d}$ such that for every $0 \leq i \leq d$, $\vec{V_i} = n \cdot \vec{\delta_i}$. Let $\mathcal{S}^*$ be a d-simplex of the edgewise subdivision of $\mathcal{S}$ using the integer n, where $\mathcal{S}^*$ is spanned by $\vec{V}_0^*, \vec{V}_1^*, \ldots, \vec{V}_d^*$. Then $\vec{V}_0^* \to \vec{V}_1^* \to \cdots \to \vec{V}_d^* \to \vec{V}_0^*$ is a geodesic path with minimal changes.*

Proof. Let M be the color scheme corresponding to $\mathcal{S}^*$ such that for every $0 \leq j \leq d$, $\vec{V}_j^* = \frac{1}{n}\sum_{i=1}^{n} \vec{V}_{\chi_{i,j}}$. Consider a scan S of M in the following way: Each row is scanned from left to right, and then the next row is scanned in the same order, until reaching the rightmost element in the last row. Namely, the scan order is $S = \chi_{1,0} \to \chi_{1,1} \to \cdots \to \chi_{1,d} \to \chi_{2,0} \to \cdots \to \chi_{n,d}$. By the definition of M, there are exactly d color changes along S. Namely, every color $0 \leq b \leq d-1$ changes to $b+1$ exactly once. Since, by the definition of M, no two successive columns are identical, then for every column $0 \leq j \leq d-1$ there is exactly one row i such that $\chi_{i,j} \neq \chi_{i,j+1}$. Moreover, $\chi_{i,j+1} = \chi_{i,j} + 1$. Therefore, exactly one color $c_j = \chi_{i,j}$ is changed when moving from column j to column $j+1$. Note further that we have already accounted for d color changes along S, and since all those color changes were of the form $\chi_{i,j} \neq \chi_{i,j+1}$, we can conclude that for every $1 \leq i \leq n-1$, $\chi_{i,d} = \chi_{i+1,0}$.

For every $0 \leq j \leq d-1$,

$$\begin{aligned}\vec{V}_{j+1}^* - \vec{V}_j^* &= \frac{1}{n}\sum_{i=1}^{n} \vec{V}_{\chi_{i,j+1}} - \frac{1}{n}\sum_{i=1}^{n} \vec{V}_{\chi_{i,j}} \\ &= \frac{1}{n}\sum_{i=1}^{n} (\vec{V}_{\chi_{i,j+1}} - \vec{V}_{\chi_{i,j}}) = \frac{1}{n}(\vec{V}_{c_j+1} - \vec{V}_{c_j})\end{aligned}$$

$$= \frac{1}{n}(n \cdot \vec{\delta}_{c_j+1} - n \cdot \vec{\delta}_{c_j}) = \vec{\delta}_{c_j+1} - \vec{\delta}_{c_j}.$$

Now examine the end of the path:

$$\begin{aligned}\vec{V}_0^* - \vec{V}_d^* &= \frac{1}{n}\sum_{i=1}^{n} \vec{V}_{\chi_{i,0}} - \frac{1}{n}\sum_{i=1}^{n} \vec{V}_{\chi_{i,d}} \\ &= \frac{1}{n}\vec{V}_{\chi_{1,0}} + \frac{1}{n}\sum_{i=1}^{n-1}(\vec{V}_{\chi_{i+1,0}} - \vec{V}_{\chi_{i,d}}) - \frac{1}{n}\vec{V}_{\chi_{n,d}} \\ &= \frac{1}{n}(\vec{V}_{\chi_{1,0}} - \vec{V}_{\chi_{k,d}}) = \frac{1}{n}(\vec{V}_0 - \vec{V}_d) \\ &= \frac{1}{n}(n \cdot \vec{\delta}_0 - n \cdot \vec{\delta}_d) = \vec{\delta}_0 - \vec{\delta}_d.\end{aligned}$$

Thus $\vec{V}_0^* \rightarrow \vec{V}_1^* \rightarrow \cdots \rightarrow \vec{V}_d^* \rightarrow \vec{V}_0^*$ is a path with minimal changes.

When passing the path from $\vec{V}_j^*$ to $\vec{V}_{j+1}^*$, there are only two changes in the elements of $\vec{V}_j^*$: $\vec{V}_j^*[c_j]$ is decreased by 1, and $\vec{V}_j^*[c_j+1]$ is increased by 1. Since the colors $c_0, c_1, \ldots, c_{d-1}$ are a permutation of the colors $0, 1, \ldots, d-1$, it follows that along the path $\vec{V}_0^* \rightarrow \vec{V}_1^* \rightarrow \cdots \rightarrow \vec{V}_d^*$ the 0th element is decreased once, the dth element is increased once, and for each $1 \leq i \leq d-1$ the ith element is increased once and decreased once. When passing from $\vec{V}_d^*$ to $\vec{V}_0^*$ the dth element is decreased once and the 0th element is increased once. Therefore, $\vec{V}_0^* \rightarrow \vec{V}_1^* \rightarrow \cdots \rightarrow \vec{V}_d^* \rightarrow \vec{V}_0^*$ is a geodesic path. ◻

Now we can prove the main theorem, given next.

THEOREM 4.4. *For every* EV *system* D *with a symmetric function f, instability*(D) $\geq vt + 1$.

Proof. $\tilde{f}$ is defined over A, which subdivides $\mathcal{S}$ using the edgewise subdivision (Lemma 4.2). We will use $\tilde{f}$ to color the vertices in A. According to Sperner's lemma, there is a $(v-1)$-simplex in the subdivision so that the colors of its vertices are pairwise different. Let $\vec{a}_0, \vec{a}_1, \ldots, \vec{a}_{v-1} \in A$ be the vertices spanning $\mathcal{S}$.

According to Lemma 4.3, $\vec{a}_0 \rightarrow \vec{a}_1 \rightarrow \cdots \rightarrow \vec{a}_{v-1} \rightarrow \vec{a}_0$ is a geodesic path with minimal changes. This means that for any b, the bth element is increased only once. Given that this path is a loop, every element is decreased only once (since every element should be increased and decreased an equal number of times), which means that for any i, j it holds that $\vec{a}_j[b] - 1 \leq \vec{a}_i[b] \leq \vec{a}_j[b] + 1$. Since for every b there is a vertex $\vec{a}_{i_b}$ such that $f(\vec{a}_{i_b}) = b$, then $\vec{a}_{i_b}[b] \geq t+1$, and hence for any k, b it holds that $\vec{a}_k[b] \geq t$.

Let c_j be the color that changes between $\vec{a}_j$ and $\vec{a}_{(j+1) \bmod v}$ for every $0 \leq j \leq v-1$. Now, for every step i we start with an input vector corresponding to $\vec{a}_{i \bmod v}$ and will switch an input value $c_{i \bmod v}$ to $(c_{i \bmod v} + 1) \bmod v$, thus arriving at an input vector corresponding to $\vec{a}_{(i+1) \bmod v}$. We can repeat these steps vt times, in total changing t input values from c_j to $c_{(j+1) \bmod v}$ for any $0 \leq j \leq v-1$. Hence, after vt steps only t input values were changed to $f(\vec{a}_0)$, so changing all the input values yet unchanged to some $b \neq f(\vec{a}_0)$ will result in another change to the consensus value, thus resulting in a total of $vt + 1$ changes to the consensus value. ◻

4.3. Instability upper bound, MLS system. We now describe a memoryless symmetric EV system MLS, and prove that its instability is a factor of two away from the lower bound presented in the previous section. An interesting open question is how to close this gap. Since MLS is memoryless and symmetric, it is defined solely by its decision function f_{MLS}. We will define f_{MLS} as follows:

$$f_{MLS}(\vec{x}) = \min \{ b \mid \#b(\vec{x}) \geq t+1 \}.$$

Namely, f_{MLS} decides upon the smallest value possible.

LEMMA 4.5. *Instability*(MLS) $\geq \min\{n, 2(v-1)(t+1)\}$.

Proof. We will prove the lemma by providing a geodesic path which yields $\min\{n, 2(v-1)(t+1)\}$ changes to the consensus value.

We will first assume that $n \geq 2(v-1)(t+1)$, and start with an input vector $\vec{x}_0 = 0^{t+1}1^{t+1} \dots (v-2)^{t+1}(v-1)^{n-(v-1)(t+1)}$. Notice that $\#\,(v-1)\,(\vec{x}_0) = n - (v-1)(t+1) \geq (v-1)(t+1)$. Since $\#0(\vec{x}_0) = t+1$, $f_{MLS}(\vec{x}_0) = 0$. Next, we switch one input value from 0 to $v-1$. Now $\#0(\vec{x}_0) = t$, and therefore the consensus value will change to 1. When we switch an input value from $v-1$ to 0, the consensus value will change back to 0. We will repeat these two steps t times, and on the $(t+1)$th time we will not change an input value from $v-1$ to 0. Notice that up until here we have changed the input value of $t+1$ sensors from 0, and of t sensors from $v-1$, and ended up with an input vector $\vec{x}_{2t+1}$, where $\#0(\vec{x}_{2t+1}) = t$. Next, we switch input values from 1 to $v-1$ and back again in the same fashion, and so forth. The geodesic path achieved this way is as follows:

$$\begin{array}{lccc}
0^{t+1}1^{t+1}2^{t+1} \dots (v-2)^{t+1}(v-1)^{n-(v-1)(t+1)} & \rightarrow & \cdots & \rightarrow \\
(v-1)^{t+1}1^{t+1}2^{t+1} \dots (v-2)^{t+1}0^{t}(v-1)^{n-(v-1)(t+1)-t} & \rightarrow & \cdots & \rightarrow \\
(v-1)^{2t+2}2^{t+1} \dots (v-2)^{t+1}0^{t}1^{t}(v-1)^{n-(v-1)(t+1)-2t} & \rightarrow & \cdots & \rightarrow \\
\vdots & & & \\
(v-1)^{(v-1)(t+1)}0^{t}1^{t} \dots (v-2)^{t}(v-1)^{n-(v-1)(2t+1)}. & & &
\end{array}$$

Every step yields $2t+1$ changes to the consensus value, resulting in a total sum of $(v-1)(2t+1)$ changes. Notice that at the end of the current path $\#i(\vec{x}_{(v-1)(2t+1)}) = t$ for every $0 \leq i \leq v-2$, thus $f_{MLS}(\vec{x}_{(v-1)(2t+1)}) = v-1$. Next, we will switch one input value from $v-1$ to $v-2$. As a result, $\#(v-2)(\vec{x}_{(v-1)(2t+1)+1}) = t+1$, and the consensus value changes to $v-2$. We can repeat this step by changing an input value from $v-1$ to $v-3$, then from $v-1$ to $v-4$, and so forth, resulting in $v-1$ more changes to the consensus value, and a total sum of $2(v-1)(t+1)$ changes. The end of the geodesic path is as follows:

$$\begin{array}{lc}
\mathcal{Z}(v-1)^{n-(v-1)(2t+1)} & \longrightarrow \\
\mathcal{Z}(v-2)(v-1)^{n-(v-1)(2t+1)-1} & \longrightarrow \\
\mathcal{Z}(v-2)(v-3)(v-1)^{n-(v-1)(2t+1)-2} & \longrightarrow \\
\mathcal{Z}(v-2)(v-3) \dots 0(v-1)^{n-2(v-1)(t+1)}, &
\end{array}$$

where

$$\mathcal{Z} = (v-1)^{(v-1)(t+1)}0^{t}1^{t} \dots (v-2)^{t}.$$

Now, let us consider the case where $n < 2(v-1)(t+1)$.

If $(v-1)(2t+1) \leq n < 2(v-1)(t+1)$, then $\#(v-1)(\vec{x}_0) \geq (v-1)t$. In this case, we can take the first $(v-1)(2t+1)$ steps described above, ending with $\vec{x}_{(v-1)(2t+1)} = \mathcal{Z}(v-1)^{n-(v-1)(2t+1)}$. There are exactly $n-(v-1)(2t+1) < v-1$ input values which equal $v-1$ and have not been changed yet. Therefore, we can follow the above $n-(v-1)(2t+1)$ steps (of the second group of steps), and yield a total of n changes to the consensus value.

If $n<(v-1)(2t+1)$, then let $\alpha=\lfloor\frac{n}{2t+1}\rfloor$, $\beta=n-\alpha(2t+1)<2t+1$. We will start the geodesic path with the input vector $\vec{x}_0=0^{t+1}1^{t+1}\dots(\alpha-1)^{t+1}\alpha^{\lceil\frac{\beta}{2}\rceil}(v-1)^{\alpha t+\lfloor\frac{\beta}{2}\rfloor}$. Notice that $\alpha<v-1$, and since $n\geq vt+1$ and $v\geq 2$, then $\alpha\geq 1$. Now, we change an input value from 0 to α, thus changing the consensus value from 0 to 1, and then change an input value from $v-1$ to 0, thus changing the consensus value from 1 to 0. We repeat these two steps $t-\lceil\frac{\beta}{2}\rceil+1$ times. Next, we change an input value from 0 to $v-1$, thus changing the consensus from 0 to 1, and then change an input value from $v-1$ to 0, changing the consensus value from 1 to 0. We repeat these two steps $\lceil\frac{\beta}{2}\rceil-1$ times, and then change an input value once from 0 to $v-1$. Now, given $\#0(\vec{x}_{2t+1})=t$, for every $1\leq b\leq\alpha$ it holds that $\#b(\vec{x}_{2t+1})=t+1$, and there are $(\alpha-1)t+\lfloor\frac{\beta}{2}\rfloor$ input values which can be changed from $v-1$ to another value. For the next $(\alpha-1)(2t+1)$ steps we change first an input value from 1 to $v-1$ and back from $v-1$ to 1 (t times) plus one change of an input value from 1 to $v-1$, we change an input value from 2 to $v-1$ and back from $v-1$ to 2 (t times) plus one change of an input value from 2 to $v-1$, etc. Every change leads to a change in the consensus value. After the $(\alpha-1)(2t+1)$th step, there are β input values left which can be changed: $\lfloor\frac{\beta}{2}\rfloor$ input values which can be changed from $v-1$ to another value, and $\lceil\frac{\beta}{2}\rceil$ input values which can be changed from α to another value. Finally, we change an input value from α to $v-1$, and then from $v-1$ to α, repeating these two steps $\lfloor\frac{\beta}{2}\rfloor$ times. If necessary (if β is odd), we change an input value from α to $v-1$. The total geodesic path is as follows:

$$\begin{array}{ll}
0^{t+1}1^{t+1}2^{t+1}\dots(\alpha-1)^{t+1}\alpha^{\lceil\frac{\beta}{2}\rceil}(v-1)^{\alpha t+\lfloor\frac{\beta}{2}\rfloor} & \longrightarrow\ \cdots\ \longrightarrow \\
\alpha^{t-\lceil\frac{\beta}{2}\rceil+1}0^{\lceil\frac{\beta}{2}\rceil}1^{t+1}2^{t+1}\dots(\alpha-1)^{t+1}\alpha^{\lceil\frac{\beta}{2}\rceil}0^{t-\lceil\frac{\beta}{2}\rceil+1}(v-1)^{(\alpha-1)t+\beta-1} & \longrightarrow\ \cdots\ \longrightarrow \\
\alpha^{t-\lceil\frac{\beta}{2}\rceil+1}(v-1)^{\lceil\frac{\beta}{2}\rceil}1^{t+1}2^{t+1}\dots(\alpha-1)^{t+1}\alpha^{\lceil\frac{\beta}{2}\rceil}0^{t}(v-1)^{(\alpha-1)t+\lfloor\frac{\beta}{2}\rfloor} & \longrightarrow\ \cdots\ \longrightarrow \\
\alpha^{t-\lceil\frac{\beta}{2}\rceil+1}(v-1)^{\lceil\frac{\beta}{2}\rceil+t+1}2^{t+1}\dots(\alpha-1)^{t+1}\alpha^{\lceil\frac{\beta}{2}\rceil}0^{t}1^{t}(v-1)^{(\alpha-2)t+\lfloor\frac{\beta}{2}\rfloor} & \longrightarrow\ \cdots\ \longrightarrow \\
\vdots & \\
\alpha^{t-\lceil\frac{\beta}{2}\rceil+1}(v-1)^{\lceil\frac{\beta}{2}\rceil+(\alpha-1)(t+1)}\alpha^{\lceil\frac{\beta}{2}\rceil}0^{t}1^{t}2^{t}\dots(\alpha-1)^{t}(v-1)^{\lfloor\frac{\beta}{2}\rfloor} & \longrightarrow\ \cdots\ \longrightarrow \\
\alpha^{t-\lceil\frac{\beta}{2}\rceil+1}(v-1)^{2\lceil\frac{\beta}{2}\rceil+(\alpha-1)(t+1)}0^{t}1^{t}2^{t}\dots(\alpha-1)^{t}\alpha^{\lfloor\frac{\beta}{2}\rfloor}. &
\end{array}$$

Since every input value changed along the geodesic path, and since every change to an input value resulted in a change to the consensus value, the instability is n. □

LEMMA 4.6. *Instability*(MLS) $\leq 2(v-1)(t+1)$.

Proof. Let $P=\vec{x}_0\rightarrow\vec{x}_1\rightarrow\cdots\rightarrow\vec{x}_n$ be a geodesic path. The number of decision value changes in P is equal to the sum of the times the decision changes to a higher value plus the times it changes to a lower value, i.e., $\#\uparrow+\#\downarrow$. We next show that each one of these quantities is at most $(v-1)(t+1)$, which proves the lemma.

To compute $\#\uparrow$, notice that if the decision d changes to a higher value, then it must be the case that an input bit equal to d changes: In the current input $\vec{x}_i$ we must have $\#d(\vec{x}_i)=t+1$ while $\#d(\vec{x}_{i+1})=t$. Now

$$\#\uparrow=\sum_{d=0}^{v-2}\#\uparrow_d,$$

where $\#\uparrow_d$ denotes the number of changes of a decision from d to a higher value. Consider the first time, $\vec{x}_j$, where the decision is d and the change in $\vec{x}_{j+1}$ is to a higher value. We have $\#d(\vec{x}_j) = t + 1$, and therefore the total number of changes from d to a higher value is at most $t + 1$ (since once this happens, $t + 1$ inputs equal to d have changed and thus are fixed in a geodesic path).

To compute $\# \downarrow$, observe that if the decision d changes in $\vec{x}_i$ to a smaller value d', then it must be the case that a value changes to d': $\#d'(\vec{x}_i) = t$ and $\#d'(\vec{x}_{i+1}) = t+1$. This can happen at most $t + 1$ times for each such d' in a geodesic path (once this happens there are $t + 1$ entries with value d' fixed). There are $(v - 1)$ such values d', so the total number of such changes is at most $(v - 1)(t + 1)$. □

COROLLARY 4.7. *Instability*(MLS) $= \min\{n, 2(v - 1)(t + 1)\}$.

Appendix. Edgewise subdivision. In [9] an edgewise subdivision of a d-simplex is suggested, which uses an abacus model of the simplex. In this appendix, we summarize the method described in [9], and its main results.

Let $\vec{V}_0, \vec{V}_1, \ldots, \vec{V}_d$ be independent vectors which define a d-simplex $\mathcal{S}$ in $\mathbb{R}^d$. Every point $\vec{X} \in \mathcal{S}$ can be described by $\lambda_0, \lambda_1, \ldots, \lambda_d$ nonnegative real numbers which sum to 1, so that

$$\vec{X} = \sum_{i=0}^{d} \lambda_i \cdot \vec{V}_i.$$

These are called *the barycentric coordinates* of $\vec{X}$. We may portray them graphically by drawing the unit interval as a rectangle with regions colored from 0 through d, making sure to arrange the colors from left to right, so that λ_i is the fraction of points with color i. Figure A.1 illustrates this for $d = 7$ and the point with barycentric coordinates $(0.26, 0.11, 0.07, 0.11, 0.19, 0.08, 0.04, 0.14)$. The coordinates of the dividing lines are displayed above the rectangle. They are provided by the partial sums $0 = B_0, B_1, \ldots, B_d, B_{d+1} = 1$ with $B_i = \lambda_0 + \lambda_1 + \cdots + \lambda_{i-1}$. B_1 through B_d can be any nondecreasing sequence in the unit interval.

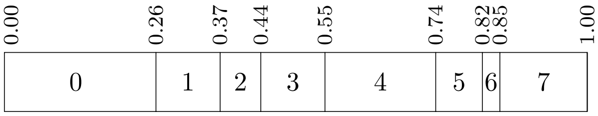


FIG. A.1. *The rectangle represents the unit interval with points colored from* 0 *to* 7.

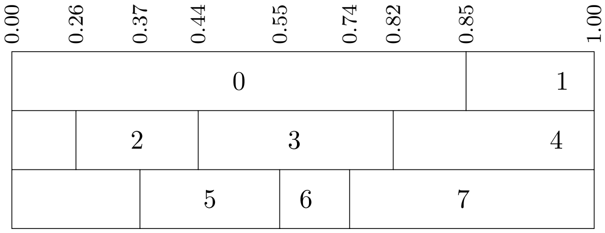
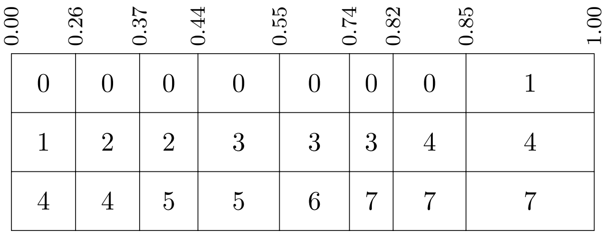


FIG. A.2. *The rectangle in Figure* A.1 *is chopped into three pieces, which are stacked and expanded.*

Suppose we chop the rectangle in Figure A.1 into k pieces of equal width, stack them on top of each other, and expand the scale by factor of k in the horizontal direction (see Figure A.2). The coordinates of the dividing lines are obtained by



0	0	0	0	0	0	0	1
1	2	2	3	3	3	4	4
4	4	5	5	6	7	7	7

FIG. A.3. *Each row is cut vertically by the coordinates, and each region keeps the original color of its points.*

multiplying the coordinates of the earlier dividing lines by k and discarding the integer part, keeping only the fractional part. We discard any duplications, letting $j+2$ be the number of distinct values remaining. Call these numbers $0 = C_0, C_1, \ldots, C_j, C_{j+1} = 1$, making sure to sort them in increasing order. We extend the dividing lines vertically until they span the entire stack, and label each region by its color (see Figure A.3).

The number of regions of each stack is $j + 1$. Forgetting the positions of the vertical dividing lines, we get a matrix,

$$M = \begin{pmatrix} \chi_{1,0} & \chi_{1,1} & \cdots & \chi_{1,j} \\ \chi_{2,0} & \chi_{2,1} & \cdots & \chi_{2,j} \\ \vdots & \vdots & \ddots & \vdots \\ \chi_{k,0} & \chi_{k,1} & \cdots & \chi_{k,j} \end{pmatrix},$$

of $k(j+1)$ color entries. We call $M = M(\vec{X}, k)$ a *color scheme* because it determines the combinatorics but not the geometry of the coloring. The matrices that may occur are those whose entries are drawn from the set $\{0, 1, \ldots, d\}$, whose entries are in nondecreasing order when read like English text,

$$\chi_{1,0} \leq \chi_{1,1} \leq \cdots \leq \chi_{1,j} \leq \chi_{2,0} \leq \cdots \leq \chi_{k,j},$$

and whose columns are pairwise different.

For every k, not necessarily distinct, colors χ_1 through χ_k we introduce the notation

$$\vec{V}_{\chi_1,\chi_2,\ldots,\chi_k} = \frac{1}{k}(\vec{V}_{\chi_1} + \vec{V}_{\chi_2} + \cdots + \vec{V}_{\chi_k}).$$

Now we can define for every $0 \leq l \leq j$, $\vec{V}_l^* = \vec{V}_{\chi_{1,l},\chi_{2,l},\ldots,\chi_{k,l}}$, where each vector corresponds to a column in the color scheme $M = M(\vec{X}, k)$. In [9] it is proved that these vectors are independent, thus spanning a j-simplex. Let $\mathcal{S}_M$ denote the simplex spanned by the vectors corresponding to M. Then the *k-edgewise subdivision* of $\mathcal{S}$ consists of all the simplices defined this way by points of $\mathcal{S}$:

$$\text{Esd}_k(\mathcal{S}) = \left\{\mathcal{S}_M \mid M = M(\vec{X}, k),\ \vec{X} \in \mathcal{S}\right\}.$$

[9] proves that not only is $\text{Esd}_k(\mathcal{S})$ a subdivision of $\mathcal{S}$, but also for every $\vec{X} \in \mathcal{S}$ it holds that $\vec{X} \in \mathcal{S}_{M(\vec{X},k)}$.

Acknowledgment. It is a pleasure to thank Ronen Peretz, Nir Shavit, and Yuri Vasilevski for their comments.

REFERENCES

[1] H. Attiya and S. Rajsbaum, *The combinatorial structure of wait-free solvable tasks*, SIAM J. Comput., 31 (2002), pp. 1286–1313.

[2] H. Attiya and J. Welch, *Distributed Computing: Fundamentals, Simulations and Advanced Topics*, McGraw–Hill, New York, 1998.

[3] A. Bar-Noy, X. Deng, J. Garay, and T. Kameda, *Optimal amortized distributed consensus*, Inform. and Comput., 120 (1995), pp. 93–100.

[4] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, American Elsevier, New York, 1976.

[5] A. P. Chandrakasan and R. W. Brodersen, *Low Power Digital CMOS Design*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1995.

[6] L. Davidovitch, S. Dolev, and S. Rajsbaum, *Consensus Continue? Stability of Multi-Valued Continuous Consensus!*, Technical Report #2004-03, Department of Computer Science, Ben-Gurion University, Beer-Sheva, Israel, 2004.

[7] S. Dolev, *Self-Stabilization*, MIT Press, Cambridge, MA, 2000.

[8] S. Dolev and S. Rajsbaum, *Stability of long-lived consensus*, J. Comput. System Sci., 67 (2003), pp. 26–45.

[9] H. Edelsbrunner and D. R. Grayson, *Edgewise subdivision of a simplex*, Discrete Comput. Geom., 24 (2000), pp. 707–719.

[10] M. J. Fischer, N. A. Lynch, and M. S. Paterson, *Impossibility of distributed consensus with one faulty process*, J. ACM, 32 (1985), pp. 374–382.

[11] J. Kahn, G. Kalai, and N. Linial, *The influence of variables on Boolean functions*, in Proceedings of the 29th Annual Symposium on Foundations of Computer Science, White Plains, NY, 1988, IEEE Computer Society Press, Piscataway, NJ, 1988, pp. 68–80.

[12] H. Kopetz and P. Veríssimo, *Real time and dependability concepts*, in Distributed Systems, S. Mullender, ed., ACM Press, New York, 1993, pp. 411–416.

[13] L. Lamport, *The part-time parliament*, ACM Trans. Comput. Systems, 16 (1998), pp. 133–169.

[14] N. A. Lynch, *Distributed Algorithms*, Morgan Kaufmann, San Francisco, 1996.

[15] K. Marzullo, *Tolerating failures of continuous-valued sensors*, ACM Trans. Comput. Systems, 8 (1990), pp. 284–304.

[16] E. Musoll, T. Lang, and J. Cortadella, *Exploiting the locality of memory references to reduce the address bus energy*, in Proceedings of the International Symposium on Low Power Electronics and Design, Monterey, CA, 1997, ACM Press, New York, pp. 202–207.

SIAM J. COMPUT.
Vol. 37, No. 4, pp. 1077–1106

PARAMETRIC DUALITY AND KERNELIZATION: LOWER BOUNDS AND UPPER BOUNDS ON KERNEL SIZE[*]

JIANER CHEN[†], HENNING FERNAU[‡], IYAD A. KANJ[§], AND GE XIA[¶]

Abstract. Determining whether a parameterized problem is kernelizable and has a small kernel size has recently become one of the most interesting topics of research in the area of parameterized complexity and algorithms. Theoretically, it has been proved that a parameterized problem is kernelizable if and only if it is fixed-parameter tractable. Practically, applying a data reduction algorithm to reduce an instance of a parameterized problem to an equivalent smaller instance (i.e., a kernel) has led to very efficient algorithms and now goes hand-in-hand with the design of practical algorithms for solving $\mathcal{NP}$-hard problems. Well-known examples of such parameterized problems include the VERTEX COVER problem, which is kernelizable to a kernel of size bounded by $2k$, and the PLANAR DOMINATING SET problem, which is kernelizable to a kernel of size bounded by $335k$. In this paper we develop new techniques to derive upper and lower bounds on the kernel size for certain parameterized problems. In terms of our lower bound results, we show, for example, that unless $\mathcal{P} = \mathcal{NP}$, PLANAR VERTEX COVER does not have a problem kernel of size smaller than $4k/3$, and PLANAR INDEPENDENT SET and PLANAR DOMINATING SET do not have kernels of size smaller than $2k$. In terms of our upper bound results, we further reduce the upper bound on the kernel size for the PLANAR DOMINATING SET problem to $67k$, improving significantly the $335k$ previous upper bound given by Alber, Fellows, and Niedermeier [*J. ACM*, 51 (2004), pp. 363–384]. This latter result is obtained by introducing a new set of reduction and coloring rules, which allows the derivation of nice combinatorial properties in the kernelized graph leading to a tighter bound on the size of the kernel. The paper also shows how this improved upper bound yields a simple and competitive algorithm for the PLANAR DOMINATING SET problem.

Key words. parameterized algorithm, planar graph, dominating set, vertex cover, independent set, kernel

AMS subject classifications. 05C85, 68Q17

DOI. 10.1137/050646354

1. Introduction.

Many practical algorithms for $\mathcal{NP}$-hard problems start by applying data reduction subroutines to the input instances of the problem. The hope is that after the data reduction phase the instance of the problem has shrunk to a moderate size. This makes the applicability of a second phase, such as a branch-and-bound phase, to the resulting instance more feasible. Weihe showed in [41] how a practical preprocessing algorithm for a variation of the DOMINATING SET problem, called the RED/BLUE DOMINATING SET problem, resulted in breaking up input instances of the problem into much smaller instances. Abu-Khzam, Langston, and Shanbhag [2], in their implementation of algorithms for the VERTEX COVER problem,

[*] Received by the editors November 30, 2005; accepted for publication (in revised form) April 11, 2007; published electronically November 14, 2007. An extended abstract of this paper appeared in STACS 2005.

http://www.siam.org/journals/sicomp/37-4/64635.html

[†] Department of Computer Science, Texas A&M University, College Station, TX 77843 (chen@cs.tamu.edu). The research of this author was supported in part by NSF grants CCF-0430683 and CCR-0311590.

[‡] FB 4–Abteilung Informatik, Universität Trier, D-54286 Trier, Germany (fernau@informatik.uni-trier.de).

[§] Corresponding author. School of CTI, DePaul University, 243 S. Wabash Avenue, Chicago, IL 60604 (ikanj@cs.depaul.edu). The work of this author was supported in part by a DePaul University competitive research grant.

[¶] Department of Computer Science, Acopian Engineering Center, Lafayette College, Easton, PA 18042 (gexia@cs.lafayette.edu).

observed the following: "In many cases, reduction was so effective that it eliminated the core completely, and with it the need for decomposition and search." Similar success was reported with DOMINATING SET as well [3].

On the other hand, many applications seek solutions of very small sizes to fairly large input instances of $\mathcal{NP}$-hard problems. This has been the main concern for the area of parameterized computation. A *parameterized problem* is a set of instances of the form (x, k), where x is the input instance and k is a nonnegative integer called the *parameter*. A parameterized problem is said to be *fixed-parameter tractable* [17] if there is an algorithm that solves the problem in time $f(k)|x|^c$, where c is a fixed constant and $f(k)$ is a recursive function. The development of efficient parameterized algorithms has provided a new approach for practically solving problems that are theoretically intractable. For example, parameterized algorithms for the $\mathcal{NP}$-hard problem VERTEX COVER [9, 13] have found applications in biochemistry [10], and variants thereof are applicable to problems arising in chip manufacturing [11, 21, 24], while parameterized algorithms in computational logic [35] have provided an effective method for solving practical instances of the ML TYPE-CHECKING problem, which is complete for the class EXPTIME [30].

The notion of a parameterized problem being parameterized tractable, and of the problem having a good data reduction algorithm, turns out to be very closely related. Informally speaking, a *kernelization*—precisely defined below—is a data reduction procedure that reduces an instance of the problem to another (smaller) instance called the *kernel*.

It has been proved that a parameterized problem is fixed-parameter tractable if and only if the problem is kernelizable [18].

Designing efficient parameterized algorithms and constructing kernels of reasonable sizes have recently been two of the main topics of research in the area of parameterized computation. More specifically, constructing a problem kernel has become one of the main components in the design of an efficient parameterized algorithm for a problem [9, 11, 12, 13], and designing efficient parameterized algorithms for a parameterized problem now goes hand-in-hand with the construction of a problem kernel of a moderate size for the problem. Two of the most celebrated problems that have been receiving a lot of attention recently from both perspectives are the VERTEX COVER and PLANAR DOMINATING SET problems. After a long sequence of algorithms, the VERTEX COVER problem can be solved in time $O(1.274^k + kn)$ [14]. Moreover, the VERTEX COVER problem enjoys a kernel of size bounded by $2k$, and reducing this bound further seems to be a very challenging task, since it would probably lead to an approximation algorithm for the problem of ratio smaller than 2—a result believed by many people to be unlikely. The PLANAR DOMINATING SET problem as well has undergone some extensive study which culminated in a recent algorithm solving the problem in time $O(2^{15.13\sqrt{k}} + n^3)$ [25]. Recently, and after many strenuous efforts, it was shown that the PLANAR DOMINATING SET problem has a problem kernel of size $335k$ that is computable in $O(n^3)$ time [5]. The question of whether such a bound on the problem kernel could be significantly improved remains open.

In this paper we develop new techniques to derive upper and lower bounds on the kernel size for certain parameterized problems. We define the notion of *duality* of a parameterized problem. Many parameterized tractable problems are the dual of parameterized intractable problems (see [19, 34, 38]). As an example, consider the VERTEX COVER and INDEPENDENT SET problems. If n denotes the number of vertices in the whole graph G, then it is well known that (G, k) is a YES-instance of VERTEX

COVER if and only if (G, k_d), where $k_d = n-k$, is a YES-instance of INDEPENDENT SET. In this sense, INDEPENDENT SET is the parametric dual problem of VERTEX COVER. While VERTEX COVER is fixed-parameter tractable on general graphs, INDEPENDENT SET is not [17]. Similarly, while DOMINATING SET is fixed-parameter intractable on general graphs, its parametric dual, called NONBLOCKER, is fixed-parameter tractable; see [15]. The landscape changes when we turn our attention towards special graph classes, e.g., problems on *planar graphs* [6]. Here, for example, both INDEPENDENT SET and DOMINATING SET are fixed-parameter tractable. In fact, and in contrast to what was stated above, there are very many problems for which both the problem itself and its dual are fixed-parameter tractable. This is also true for problems on graphs of bounded genus, as well as on graphs of bounded degree.

The beauty of problems which, together with their dual problems, are fixed-parameter tractable is that this constellation allows for, from an algorithmic standpoint, a two-sided attack on the original problem. This two-sided attack enabled us to derive lower bounds on the kernel size for such problems (under classical complexity assumptions). For example, we show that unless $\mathcal{P} = \mathcal{NP}$, PLANAR VERTEX COVER does not have a kernel of size smaller than $4k/3$, and PLANAR INDEPENDENT SET and PLANAR DOMINATING SET do not have kernels of size smaller than $2k$. To the authors' knowledge, this is the first group of results establishing lower bounds on the kernel size of parameterized problems. We also show that some lower bound results obtained using the techniques devised in this paper are sharp by exhibiting a family of graph classes on which the lower bound on the kernel size of the restricted $\mathcal{NP}$-hard VERTEX COVER problem approaches the upper bound $2k$ with an arbitrary precision.

Whereas the lower bounds on the kernel size for PLANAR VERTEX COVER and PLANAR INDEPENDENT SET come close to the known upper bounds of $2k$ and $4k$ on the kernel size for the two problems, respectively, the lower bound derived for PLANAR DOMINATING SET is still very far from the $335k$ upper bound on the problem kernel, which was given by Alber, Fellows, and Niedermeier [5]. To bridge this gap, we investigate the problem of finding a kernel of smaller size for the PLANAR DOMINATING SET problem and derive better upper bounds on the problem kernel for the problem. We improve the reduction rules proposed in [5] and introduce new rules that *color* the vertices of the graph, enabling us to observe many new combinatorial properties of its vertices. These properties allow us to prove a much stronger bound on the number of vertices in the reduced graph. We show that the PLANAR DOMINATING SET problem has a kernel of size $67k$ that is computable in $O(n^3)$ time. This is a significant improvement over the results in [5]. Finally, we show how the resulting bound on the kernel size yields a very simple algorithm for the PLANAR DOMINATING SET problem that beats some previous algorithms for the problem, and whose running time even comes close to some of the recently proposed algorithms.

2. Preliminaries.

A graph G is said to be *planar* if G can be embedded on the plane such that no two edges in G cross. It is well known that deciding whether a graph is planar and constructing a planar embedding of the graph in such case can be done in linear time [31]. The number of edges in a planar graph with n vertices for $n \geq 3$ is bounded by $3n - 6$ [16].

A dominating set in a graph G is a set of vertices D such that every vertex in G is either in D or adjacent to at least one vertex in D. The size of a dominating set D is the number of vertices in D. A *minimum dominating set* of G is a dominating set with the minimum size. We will denote by $\gamma(G)$ the size of a minimum dominating set in G. The PLANAR DOMINATING SET problem, abbreviated PLANAR-DS henceforth, is the following: given a planar graph G and a positive integer k, either construct a dominating set for G of size at most k or report that no such dominating set exists. It is well known that the PLANAR-DS problem is $\mathcal{NP}$-complete [27].

A *parameterized problem* P is a subset of $\Sigma^* \times N$, where Σ is a fixed alphabet and N is the set of all nonnegative integers. Therefore, each instance of the parameterized problem P is a pair (I, k), where the second component k is called the *parameter*. The language $L(P)$ is the set of all YES-instances of P. We say that the parameterized problem P is *fixed-parameter tractable* [17] if there is an algorithm that decides whether an input (I, k) is a member of $L(P)$ in time $f(k)|I|^c$, where c is a fixed constant and $f(k)$ is a recursive function independent of the input length $|I|$. The class of all fixed-parameter tractable problems is denoted by FPT.

A mapping $s : \Sigma^* \times N \to N$ is called a *size function* for a parameterized problem P if

- $0 \leq k \leq s(I, k)$,
- $s(I, k) \leq |I|$, and
- $s(I, k) = s(I, k')$ for all appropriate k, k' *(independence)*. Hence, we can also write $s(I)$ for $s(I, k)$.

A problem P together with its size function s is denoted (P, s). The *dual problem* P_d of P is the problem whose corresponding language (i.e., the set of YES-instances) $L(P_d) = \{(I, s(I) - k) \mid (I, k) \in L(P)\}$. The dual of the dual of a problem (with a given size function) is again the original problem. We give some examples below.

d-HITTING SET

Given: A hypergraph $G = (V, E)$ with *edge degree* bounded by d, i.e., for all $e \in E$, $|e| \leq d$.

Parameter: k.

Question: Is there a *hitting set* of size at most k, i.e.,

$$\exists C \subseteq V, |C| \leq k, \quad \forall e \in E, C \cap e \neq \emptyset?$$

The special case in which $d = 2$ corresponds to the VERTEX COVER problem in undirected graphs. Let $L(d\text{-HS})$ denote the language of d-HITTING SET. Taking as size function $s(G) = |V|$, it is clear that the dual problem obeys $(G, k_d) \in L(d\text{-HS}_d)$ if and only if G has an *independent set* of cardinality k_d.

DOMINATING SET

Given: A (simple) graph $G = (V, E)$.

Parameter: k.

Question: Is there a *dominating set* of size at most k, i.e.,

$$\exists D \subseteq V, |D| \leq k, \quad \forall v \in V \setminus D \; \exists w \in D, (w, v) \in E?$$

Taking as size function $s(G) = |V|$, it is clear that the dual problem obeys $(G, k_d) \in L(\text{DS}_d)$ if and only if G has a *nonblocker set* (i.e., the complement of a dominating set) of cardinality k_d.

Generally speaking, it is easy to "correctly" define the dual of a problem for the so-called selection problems as formalized in [7]. The concept of duality is less clear, say, for weighted graph problems (with the slight exception of ROMAN DOMINATION; see [22]). Also, different graph parameterizations like treewidth seem to possess no natural dualization.

A *kernelization (reduction)* for a parameterized problem P with size function s is a polynomial-time computable reduction which maps an instance (I, k) onto (I', k') such that (1) $s(I') \leq g(k)$ (g is a recursive function), (2) $k' \leq k$, and (3) $(I, k) \in L(P)$

if and only if $(I', k') \in L(P)$. I' is called the *problem kernel* of I. It is known (see [18]) that a parameterized problem is fixed-parameter tractable if and only if it has a kernelization. Of special interest to us in this paper are problems with *linear kernels* in which $g(k) = \alpha k$ for some constant $\alpha > 0$. Such small kernels are known for many important graph problems restricted to planar graphs.

3. Lower bounds on kernel size.

Practice in the study of parameterized algorithms has suggested that improved kernelization can lead to improved parameterized algorithms. Many efforts have been made towards obtaining smaller kernels for well-known $\mathcal{NP}$-hard parameterized problems (see, for example, [5, 13, 18]). A natural question to ask along this line of research is about the limit of polynomial-time kernelization. In this section we develop techniques for deriving lower bounds on the kernel size for certain well-known $\mathcal{NP}$-hard parameterized problems.

3.1. General lower bound results.

THEOREM 3.1. *Let (P, s) be an $\mathcal{NP}$-hard parameterized problem (with size function s). Suppose that P admits an αk kernelization and its dual P_d admits an $\alpha_d k_d$ kernelization, where $\alpha, \alpha_d \geq 1$. If $(\alpha - 1)(\alpha_d - 1) < 1$, then $\mathcal{P} = \mathcal{NP}$.*

Proof. Suppose that the assumption of the theorem is true, and let $r(\cdot)$ denote the assumed linear kernelization reduction for P. Similarly, let $r_d(\cdot)$ be the linear kernelization reduction for P_d. Consider the following reduction R, which on input (I, k) of P performs the following:

```
    **if** $k \leq \frac{\alpha_d}{\alpha+\alpha_d} s(I)$ **then** compute $r(I, k)$;
    **else** compute $r_d(I, s(I) - k)$.
```

Let (I', k') be the instance computed by the reduction R. If $k \leq \frac{\alpha_d}{\alpha+\alpha_d} s(I)$, then $s(I') \leq \alpha k \leq \frac{\alpha\alpha_d}{\alpha+\alpha_d} s(I)$. Otherwise,

$$\begin{aligned} s(I') &\leq \alpha_d k_d \\ &= \alpha_d(s(I) - k) \\ &< \alpha_d \left(s(I) - \frac{\alpha_d}{\alpha + \alpha_d} s(I) \right) \\ &= \frac{\alpha\alpha_d}{\alpha + \alpha_d} s(I). \end{aligned}$$

Since $(\alpha - 1)(\alpha_d - 1) < 1$, or equivalently $\frac{\alpha\alpha_d}{\alpha+\alpha_d} < 1$, by repeatedly applying R (at most polynomially many times), the problem P can be solved in polynomial time. This completes the proof. □

The condition "$\alpha, \alpha_d \geq 1$" in the previous theorem is not crucial in the light of the following lemma.

LEMMA 3.2. *Let (P, s) be a parameterized problem such that P admits a kernelization $r(\cdot)$ with $s(r(I, k)) \leq \alpha k$ for some $\alpha < 1$. Then P is in $\mathcal{P}$.*

Proof. According to our definition of the size function, we have $s(I') \geq k'$ for each instance (I', k'). This is particularly true for the parameter k' of the problem kernel instance $I' = r(I, k)$. Therefore, $k' \leq \alpha k$ for some $\alpha < 1$. By repeated kernelization, we arrive at a problem with an arbitrarily small parameter and, hence, of arbitrarily small size. In fact, we need $\mathcal{O}(\log k)$ many such kernelizations, each of them requiring polynomial time. It follows that the given problem can be decided in polynomial time. □

Remark. The assumption that $s(I) \geq k$ is crucial here. As a concrete "counter-example," consider the *decision tree problem*, specified by n objects $X = \{x_1, \ldots, x_n\}$,

t boolean tests $T = \{T_1, \dots, T_t\}$, and a parameter k. In this setting, a decision tree is a binary tree B whose leaves are (uniquely) labeled with objects and whose inner nodes are labeled with tests such that on the path from the root to the leaf labeled x_i, tests are performed that uniquely distinguish x_i from all other objects. The overall length of all paths from the root to each leaf is usually considered as the cost function. The question is whether there exists a decision tree with cost bounded by k. This problem is known to be $\mathcal{NP}$-complete (see [32]).

If $n = 2^\ell$, the decision tree with optimal cost is surely the complete binary tree (possibly not attainable with the given set of tests), since it is optimally balanced. Hence, we have $k > n \log_2 n$ (otherwise, an algorithm can simply answer NO); this can be seen as a trivial kernelization algorithm. Therefore, $n \in o(k)$. This can be interpreted as giving the (to our knowledge) first natural parameterized problem having a sublinear kernel. On the other hand, this relation also implies that $s(I, k) < k$ is true here, so that the previous lemma does not lead to a contradiction with the known $\mathcal{NP}$-hardness result.

The problem here is the seemingly innocuous choice of the size function as being $n = |X|$. Observe that any "reasonable" encoding of an instance would rather use $n \log n$ bits, since each element of X would need to get a name. This way, the problem would disappear.

3.2. Concrete lower bound results. From Theorem 3.1 and assuming $\mathcal{P} \neq \mathcal{NP}$, we immediately obtain Corollaries 3.3–3.8.

COROLLARY 3.3. *For any $\epsilon > 0$, there is no $(4/3-\epsilon)k$ kernel for* PLANAR VERTEX COVER.

Proof. The four-color theorem implies a $4k$-kernelization for PLANAR INDEPENDENT SET, which is the dual problem of PLANAR VERTEX COVER. □

COROLLARY 3.4. *For any $\epsilon > 0$, there is no $(2-\epsilon)k$ kernel for* PLANAR INDEPENDENT SET. *This result remains true if we restrict the problem to graphs of maximum degree bounded by three, or even to planar graphs of maximum degree bounded by three (both problems are $\mathcal{NP}$-hard).*

Proof. The general VERTEX COVER problem, which is the dual of the INDEPENDENT SET problem, has a $2k$-kernelization [13] (based on a result of Nemhauser and Trotter). This kernelization is both planarity and bounded-degree preserving. □

COROLLARY 3.5. *For any $\epsilon > 0$, there is no $(3/2-\epsilon)k$-kernelization for* VERTEX COVER *restricted to triangle-free planar graphs (this problem is still $\mathcal{NP}$-hard* [40, *Chapter* 7]*).*

Proof. Based on a theorem by Grötzsch (which can be turned into a polynomial-time coloring algorithm; see [29]), it is known that planar triangle-free graphs are 3-colorable. This implies a $3k$ kernel for INDEPENDENT SET restricted to this graph class, which gives the result. Observe that the mentioned $2k$-kernelization for VERTEX COVER on general graphs preserves planarity and triangle-freeness, which implies that this restriction of the problem has a $2k$-kernelization. □

COROLLARY 3.6. *For any $\epsilon > 0$, there is no $(335/334-\epsilon)k_d$ kernel for* PLANAR NONBLOCKER.

Proof. A $335k$ kernel for PLANAR-DS was derived in [5]. □

COROLLARY 3.7. *For any $\epsilon > 0$, there is no $(2-\epsilon)k$ kernel for* PLANAR-DS. *This remains true when further restricting the graph class to planar graphs of maximum degree three (the problem is still $\mathcal{NP}$-hard).*

Proof. In [20], a $2k_d$-kernelization for NONBLOCKER on general graphs which preserves planarity and degree bounds was derived (see also [37, Thm. 13.1.3]). □

Corollary 3.8 (see [15]). *For any $\epsilon > 0$, there is no $(2-\epsilon)k$ kernel for* dominating set on cubic graphs. This is interesting, since that case is the best match between upper and lower bounds for domination problems.

The above results open a new line of research and prompt us to ask whether we can find examples of problems such that the derived kernel sizes are optimal (unless $\mathcal{P} = \mathcal{NP}$), and whether we can close the gaps between the upper bounds and lower bounds on the kernel size. According to our previous discussion, planar vertex cover on triangle-free graphs is our "best match": we know how to derive a kernel of size $2k$ and (assuming $\mathcal{P} \neq \mathcal{NP}$) we know that no kernel smaller than $3k/2$ exists. On the other hand, the $335k$ upper bound on the kernel size for planar-DS [5] is very far from the $2k$ lower bound proved above. In the next sections, we improve this upper bound to $67k$ in an effort to bridge the huge gap between the upper bound and lower bound on the kernel size for this problem. This allows us to state the following corollary.

Corollary 3.9. *Assuming $\mathcal{P} \neq \mathcal{NP}$, there is no $(67/66-\epsilon)k_d$ kernel for* planar nonblocker *for any choice of $\epsilon > 0$.*

Remark. Since "Euler-type" theorems exist for graphs of bounded genus g, it can be shown that there is a constant c_g such that each graph of genus g is c_g-colorable. Therefore, lower bounds on the kernel sizes of vertex cover on graphs of genus g can be derived. For triangle-free graphs of genus g, Thomassen has shown that the corresponding constant c'_g is in $\mathcal{O}(g^{1/3}(\log g)^{-2/3})$ (see [28, 39]).

Remark. Recently, Fomin and Thilikos [26] were able to extend the linear kernel result for dominating set to graphs on surfaces of bounded genus. Therefore, our lower bound results extend to these more general graph classes as well.

3.3. Can we improve on the lower bounds?

In the following, we reproduce a construction that is essentially due to Paul Seymour.[1] This construction shows that the lower bound results obtained using the techniques devised in this section can be sharp for certain problems.

Consider the following family $\mathcal{G}_n$ of graph classes. A graph G is in $\mathcal{G}_n$ if and only if it satisfies the following two conditions.

1. $G = (V, E)$ can be partitioned into $2n+1$ mutually disjoint independent sets, i.e., $V = I_1 \cup \cdots \cup I_{2n+1}$, $I_i \cap I_j = \emptyset$ for all $1 \leq i < j \leq 2n+1$, and the induced graphs $G[I_i]$ contain no edges.
2. The edge set E can be partitioned into $2n+1$ groups $E_1, \ldots, E_{2n+1}$ such that
$$E_i = E(G[I_i \cup I_{i \bmod (2n+1)+1}]).^2$$

Figure 1 provides an example of a graph in $\mathcal{G}_n$.

Since each of these classes is closed under taking induced subgraphs, we can deduce by the Nemhauser–Trotter kernelization [13] the following lemma.

Lemma 3.10. vertex cover *restricted to $\mathcal{G}_n$ admits a kernel of size $2k$ (within $\mathcal{G}_n$).*

Since the graphs in $\mathcal{G}_n$ are "nearly bipartite," we have the following result.

Lemma 3.11. independent set *restricted to $\mathcal{G}_n$ admits a kernel of size $(2 + 1/n)k_d$ (within $\mathcal{G}_n$).*

Proof. Let $G = (V, E) \in \mathcal{G}_n$ with an independent set decomposition $V = I_1 \cup \cdots \cup I_{2n+1}$ that certifies this membership. To simplify the notation, we assume that

[1] Personal communication (2005).

[2] We assume that the graph G is given with a certificate of membership in $\mathcal{G}_n$, which is a partitioning of its vertex set into the $2n+1$ subsets.

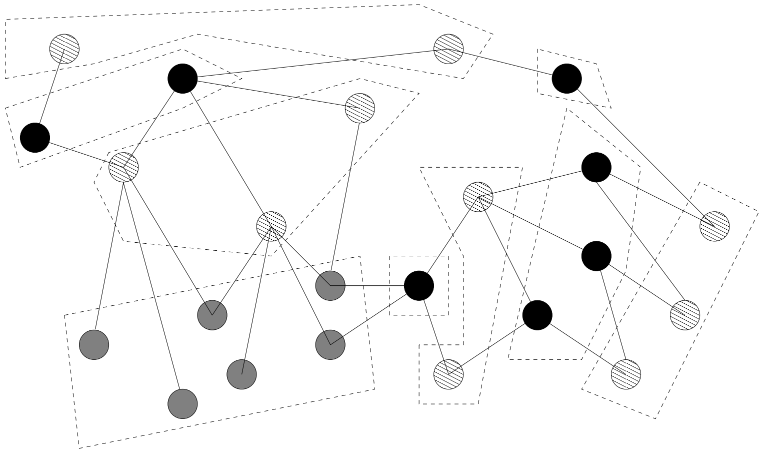


FIG. 1. *An example of a graph in $\mathcal{G}_n$.*

additions and subtractions of indices are all performed modulo $2n+1$. Consider the sets

$$J_i = I_i \cup I_{i+2} \cup \cdots \cup I_{i-3}, \quad 1 \le i \le 2n+1.$$

J_i greedily collects "every second" set starting at I_i so that each set J_i forms an independent set of G. It can be easily verified that

$$\sum_{i=1}^{2n+1} |J_i| = n|V|.$$

This shows that there exists an index i for which the set J_i contains at least a fraction $n/(2n+1)$ of all the vertices. Moreover, such a J_i can be found in polynomial time. Therefore, we can answer YES straightaway whenever we are given a graph $G = (V, E) \in \mathcal{G}_n$ with a parameter $k_d \le (n/(2n+1))|V|$, as an instance of INDEPENDENT SET. This means that we have $|V| < (2+1/n)k_d$ for all the remaining instances. □

THEOREM 3.12. *For each n,* VERTEX COVER *restricted to $\mathcal{G}_n$ is $\mathcal{NP}$-complete.*

Proof. Membership in $\mathcal{NP}$ is inherited from the general VERTEX COVER problem. We will show that 3-SAT is polynomial-time reducible to VERTEX COVER restricted to $\mathcal{G}_n$. We highlight the main elements in the reduction here and leave the verification of some of the details to the interested reader.

Let $\mathcal{C} = \{C_1, \ldots, C_m\}$ be a collection of clauses. Without loss of generality, we can assume that $|C_i| = 3$ for all $1 \le i \le m$. Let ℓ_i^j refer to the jth literal in clause C_i, i.e., $\ell_i^j = y_i^j$ or $\ell_i^j = \bar{y}_i^j$ for some variable $y_i^j \in X = \{x_1, \ldots, x_r\}$.

We construct a graph $G = (V, E) \in \mathcal{G}_n$ as follows. For each variable x_i, we introduce a cycle $(v_i^1, \ldots, v_i^{4n+2})$ of even length. Clearly, $2n+1$ of these vertices will be in any feasible vertex cover. For each clause C_i, we introduce a cycle $(u_i^1, \ldots, u_i^{2n+1})$ of odd length. Clearly, $n+1$ of these vertices will be in any feasible vertex cover. Summarizing, the graph described so far will have at least $r(2n+1) + m(n+1)$ vertices in any feasible cover. Since we will now add more edges to this graph, the lower bound on the size of the vertex cover will still be valid. At the same time, we will maintain the property that $I_j = \{v_i^j, v_i^{j+2n+1} \mid 1 \le i \le r\} \cup \{u_i^j \mid 1 \le i \le m\}$, where $1 \le j \le 2n+1$, are all independent sets.

If $\ell_i^1 = y_i^1 = x_q$, then we will make u_i^1 adjacent to v_q^2. If $\ell_i^2 = y_i^2 = x_q$, then we will make u_i^2 adjacent to v_q^{2n+2}. If $\ell_i^3 = y_i^3 = x_q$, then we will make u_i^3 adjacent to v_q^2.

If $\ell_i^1 = \bar{y}_i^1 = \bar{x}_q$, then we will make u_i^1 adjacent to v_q^{2n+3}. If $\ell_i^2 = \bar{y}_i^2 = \bar{x}_q$, then we will make u_i^2 adjacent to v_q^1. If $\ell_i^3 = \bar{y}_i^3 = \bar{x}_q$, then we will make u_i^3 adjacent to v_q^{2n+3}.

Now, if x_i is set to true in a satisfying assignment of the given 3-SAT instance, then we put v_i^j into the vertex cover if and only if j is even. If x_i is set to false in a satisfying assignment of the given 3-SAT instance, then we put v_i^j into the vertex cover if and only if j is odd. It is not difficult to verify that a satisfying assignment of $\mathcal{C}$ can be translated into a feasible vertex cover of size $r(2n+1)+m(n+1)$.

The same identification of vertices from v_i^j with variable settings allows us to translate a feasible vertex cover of size $r(2n+1)+m(n+1)$ into a satisfying assignment for the given 3-SAT instance. □

COROLLARY 3.13. *Unless $\mathcal{P} = \mathcal{NP}$, the* VERTEX COVER *problem restricted to $\mathcal{G}_n$ does not have a kernel of size $(2-\epsilon)k$ for any $\epsilon > 0$.*

Proof. This follows from Lemma 3.11 and Theorem 3.1. □

Remark. The above corollary shows that the lower bound results on the kernel size for VERTEX COVER restricted to $\mathcal{G}_n$ obtained using the techniques in this paper are tight. It also shows that it is unlikely that the VERTEX COVER problem on general graphs admits a kernelization of size $(2-\epsilon)k$ with the property that the produced kernel is a subgraph of the original graph, the reason being that such a kernelization would also be a kernelization for the vertex cover problem restricted to the class $\mathcal{G}_n$ with the same kernel bound.

3.4. A possible two-sided attack for exact algorithms. With problems having both FPT algorithms for their primal and for their dual parameterizations, we have the possibility of converting both algorithms into a nonparameterized algorithm. This is like attacking the problem from two sides. This means that we can use either of the two FPT algorithms, depending on "to which side" our concrete problem instance is closer.

THEOREM 3.14. *Let (P, s) be a parameterized problem and P_d its dual. Assume that both P and P_d are in FPT. Let f be a monotone function. Assume that there is an algorithm A that solves an instance (I, k) of P in time $\mathcal{O}(f(\beta k)p(s(I)))$ for some polynomial p, and that A_d is an algorithm that solves an instance (I, k_d) of P_d in time $\mathcal{O}(f(\beta_d k_d)p_d(s(I)))$ for a polynomial p_d.*

Then, there is an algorithm A' for solving the nonparameterized problem instance I running in time

$$\mathcal{O}\left(f\left(\frac{\beta\beta_d}{\beta+\beta_d}s(I)\right)p'(s(I))\right),$$

for some polynomial p'.

Proof. The idea is to use algorithm A as long as it is better than using A_d. This means that we have to compare

$$f(\beta k) \quad \text{to} \quad f(\beta_d(s(I)-k_d)).$$

Since f is monotone, this means we can simply compare

$$\beta k \quad \text{to} \quad \beta_d(s(I)-k_d).$$

Some simple algebra shows that we can have the following algorithm A' for the nonparameterized problem P, given an instance I:

```
for all parameter values $k$ do:
if $k \leq \frac{\beta_d}{\beta+\beta_d}s(I)$ then compute $A(I,k)$;
else compute $A_d(I, s(I)-k)$;
output the 'best' of all computed solutions.
```

Considering the boundary case $k = \frac{\beta_d}{\beta+\beta_d}s(I)$ gives the claimed worst-case running time. Here, $p'(j) = j(p(j)+p_d(j))$. □

Unfortunately, we currently lack good examples that prove this approach superior to published (problem-tailored) exact algorithms.

4. Reduction and coloring rules.

In this section we show how to improve the upper bound on the kernel size for the PLANAR-DS problem to $67k$. In the remainder of the paper we will always assume that the graph we are dealing with is planar.

In this section we present an $O(n^3)$ time preprocessing scheme that reduces the graph G to a graph G' such that $\gamma(G) = \gamma(G')$ and such that given a minimum dominating set for G', a minimum dominating set for G can be constructed in linear time. We will color the vertices of the graph G with two colors: black and white. Initially, all vertices are colored black. Informally speaking, white vertices will be those vertices for which we know for sure when we color them that there exists a minimum dominating set for the graph excluding all of them. The black vertices are all other vertices. Note that it is possible for white vertices to be in some minimum dominating set, but the point is that there exists at least one minimum dominating set that excludes all white vertices. Hence, the black-and-white coloring is only an auxiliary structure and not part of the problem definition. We start with the following definitions that are adopted from [5] with minor additions and modifications.

For a vertex v in G denote by $N(v)$ the set of neighbors of v, and by $N[v]$ the set $N(v) \cup \{v\}$. By removing a vertex v from G, we mean removing v and all the edges incident on v from G. For a vertex v in G, we partition its set of neighbors $N(v)$ into three sets: $N_1(v) = \{u \in N(v) \mid N(u) - N[v] \neq \emptyset\}$; $N_2(v) = \{u \in N(v) - N_1(v) \mid N(u) \cap N_1(v) \neq \emptyset\}$; and $N_3(v) = N(v) - (N_1(v) \cup N_2(v))$. For two vertices v and w we define $N(v,w) = N(v) \cup N(w)$ and $N[v,w] = N[v] \cup N[w]$. We partition $N(v,w)$ into three sets: $N_1(v,w) = \{u \in N(v,w) \mid N(u) - N[v,w] \neq \emptyset\}$; $N_2(v,w) = \{u \in N(v,w) - N_1(v,w) \mid N(u) \cap N_1(v,w) \neq \emptyset\}$; and $N_3(v,w) = N(v,w) - (N_1(v,w) \cup N_2(v,w))$.

DEFINITION 4.1. *Let $G = (V,E)$ be a plane graph. A region $R(v,w)$ between two vertices v and w is a closed subset of the plane with the following properties:*

1. *The boundary of $R(v,w)$ is formed by two simple paths P_1 and P_2 in G which connect v and w, and the length of each path is at most three.*
2. *All vertices that are strictly inside (i.e., not on the boundary) the region $R(v,w)$ are from $N(v,w)$.*

For a region $R = R(v,w)$, let $V[R]$ denote the vertices in R; i.e.,

$$V[R] := \{u \in V \mid u \textit{ sits inside or on the boundary of } R\}.$$

Let $V(R) = V[R] - \{v,w\}$.

DEFINITION 4.2. *A region $R = R(v,w)$ between two vertices v and w is called* simple *if all vertices in $V(R)$ are common neighbors of both v and w; that is, $V(R) \subseteq N(v) \cap N(w)$.*

We introduce the following definition.

DEFINITION 4.3. *A region $R = R(v,w)$ between two vertices v and w is called* quasi-simple *if $V[R] = V[R'] \cup R^+$, where $R' = R'(v,w)$ is a simple region between v and w, and R^+ is a set of white vertices satisfying the following conditions.*

1. *Every vertex of R^+ sits strictly inside R'.*
2. *Every vertex of R^+ is connected to v and not connected to w, and is also connected to at least one vertex on the boundary of R' other than v.*

A vertex in $V(R)$ is called a *simple* vertex if it is connected to both v and w; otherwise it is called *nonsimple*. The set of vertices R^+, which consists of the nonsimple vertices in $V(R)$, will be referred to as $R^+(v, w)$.

For a vertex $u \in V$, denote by $B(u)$ the set of black vertices in $N(u)$ and by $W(u)$ the set of white vertices in $N(u)$. We describe next the reduction and coloring rules to be applied to the graph G. The reduction and coloring rules are applied to the graph until the application of any of them does not change the structure of the graph or the color of any vertex in the graph. The first two reduction rules, Rules 1 and 2, are slight modifications of Rules 1 and 2 introduced in [5]. The only difference is that in the current paper they are applied only to black vertices, and not to all the vertices as in [5].

Rule 1 (see [5]). If $N_3(v) \neq \emptyset$ for some black vertex v, then remove the vertices in $N_2(v) \cup N_3(v)$ from G and add a new white vertex v' and an edge (v, v') to G.

Rule 2 (see [5]). If $N_3(v, w) \neq \emptyset$ for two black vertices v, w and if $N_3(v, w)$ cannot be dominated by a single vertex in $N_2(v, w) \cup N_3(v, w)$, then we distinguish the following two cases.

Case 1. If $N_3(v, w)$ can be dominated by a single vertex in $\{v, w\}$, then (1) if $N_3(v, w) \subseteq N(v)$ and $N_3(v, w) \subseteq N(w)$, remove $N_3(v, w)$ and $N_2(v, w) \cap N(v) \cap N(w)$ from G and add two new white vertices z, z' and the edges $(v, z), (w, z), (v, z'), (w, z')$ to G; (2) if $N_3(v, w) \subseteq N(v)$ and $N_3(v, w) \not\subseteq N(w)$, remove $N_3(v, w)$ and $N_2(v, w) \cap N(v)$ from G and add a new white vertex v' and the edge (v, v') to G; and (3) if $N_3(v, w) \subseteq N(w)$ and $N_3(v, w) \not\subseteq N(v)$, remove $N_3(v, w)$ and $N_2(v, w) \cap N(w)$ from G and add a new white vertex w' and the edge (w, w') to G.

Case 2. If $N_3(v, w)$ cannot be dominated by a single vertex in $\{v, w\}$, then remove $N_2(v, w) \cup N_3(v, w)$ from G and add two new white vertices v', w' and the edges (v, v'), (w, w') to G.

Rule 3. For each black vertex v in G, if there exists a black vertex $x \in N_2(v) \cup N_3(v)$, color x white and remove the edges between x and all other white vertices in G.

Rule 4. For every two black vertices v and w, if $N_3(v, w) \neq \emptyset$, then for every black vertex $x \in N_2(v, w) \cup N_3(v, w)$ that does not dominate all vertices in $N_3(v, w)$, color x white and remove all the edges between x and the other white vertices in G.

Rule 5. For every quasi-simple region $R = R(v, w)$ between two vertices v and w, if v is black, then for every black vertex $x \in N_2(v, w) \cup N_3(v, w)$ strictly inside R that does not dominate all vertices in $N_2(v, w) \cup N_3(v, w)$ strictly inside R, color x white and remove all the edges between x and the other white vertices in G.

Rule 6. For every two white vertices u and v, if $N(u) \subseteq N(v)$ and $u \in N_2(w) \cup N_3(w)$ for some black vertex w, then remove v.

Rule 7. For every black vertex v, if every vertex $u \in W(v)$ is connected to all the vertices in $B(v)$, then remove all the vertices in $W(v)$ from G.

Rule 8. For every two black vertices v and w, let $W(v, w) = W(v) \cap W(w)$. If $|W(v, w)| \geq 2$ and there is a degree-2 vertex $u \in W(v, w)$, then remove all vertices in $W(v, w)$ except u, add a new degree-2 white vertex u', and connect u' to both v and w.

Figure 2 illustrates Rules 4, 6, and 8.

A graph G is said to be *reduced* if every vertex in G is colored white or black and

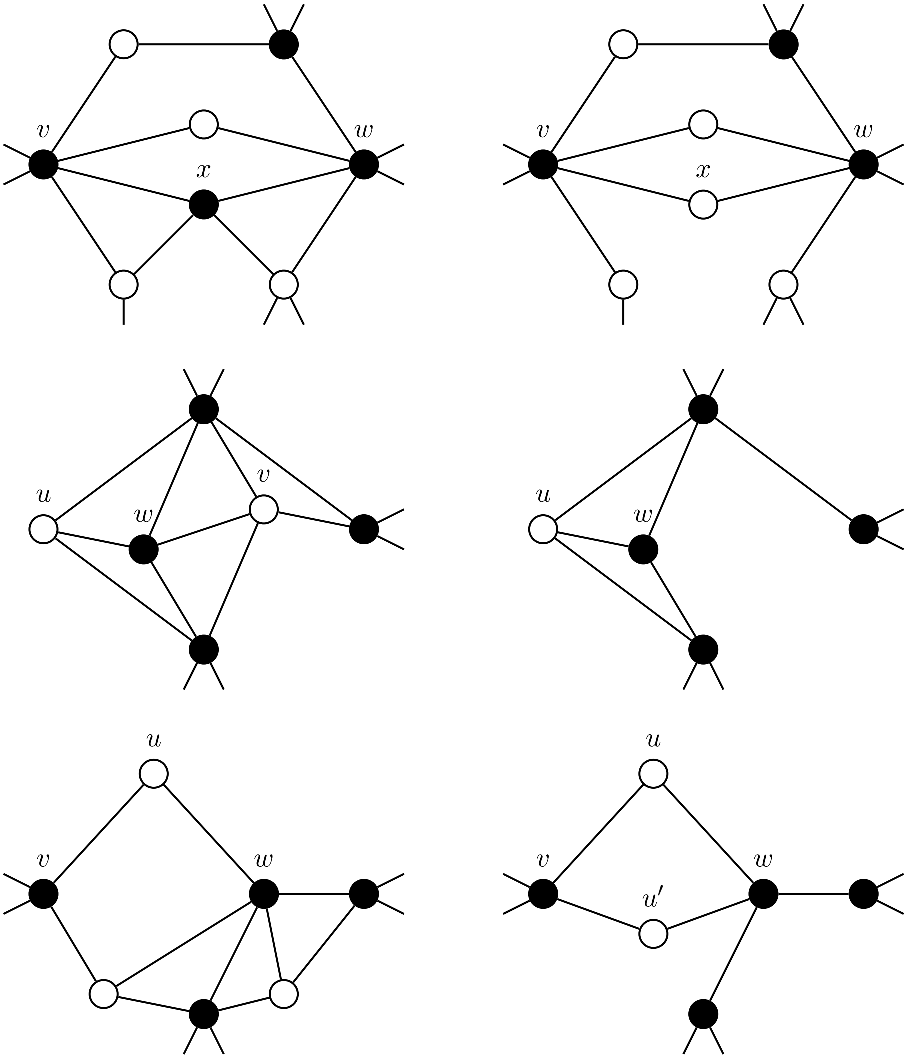


FIG. 2. *Illustrations of Rule* 4 *(top figure), Rule* 6 *(middle figure), and Rule* 8 *(bottom figure).*

the application of Rules 1–8 leaves the graph G unchanged. That is, the application of any of the above rules does not change the color of any vertex in G, nor does it change the structure of G. We have the following theorem.

THEOREM 4.4. *Let G be a planar graph with n vertices. Then in time $O(n^3)$ we can construct a planar graph G' from G such that* (1) *G' is reduced,* (2) $\gamma(G') = \gamma(G)$, (3) *there exists a minimum dominating set for G' that excludes all white vertices of G', and* (4) *from a minimum dominating set for G' a minimum dominating set for G can be constructed in linear time.*

Proof. Given a graph G, we first color all its vertices black. We then apply Rules 1–8 given above until the application of any of these rules leaves G unchanged. Let G' be the resulting graph. Then G' is reduced by the definition of a reduced graph. Alber, Fellows, and Niedermeier [5] noted that each successful application of Rules 1 and 2 (i.e., an application that changes the structure of the graph G) reduces the number of vertices in the graph by at least one. Hence, the total number of applications of these two rules is bounded by n. By looking at Rules 3–7, it is easy to see that each of these rules either reduces the number of vertices in G by at least one or changes the color of at least one black vertex to white without adding any new vertices to the graph. Moreover, none of Rules 1–7 increases the number of edges in the graph. If we look at Rule 8, it is not difficult to see that each successful application of this rule reduces the number of edges in the graph by at least 1. This is true since in a successful application of the rule either $|W(v,w)| > 2$ (and in this case the numbers of vertices and edges decrease after the application of the rule) or $|W(v,w)| = 2$ and there is a vertex in $W(v,w)$ of degree larger than 2 (otherwise the application of the rule does not change the structure of the graph), and hence the removal of $W(v,w)$ decreases the number of edges in the graph. Noting that the number of edges in a planar graph is linear in the number of vertices and that the application of the rules becomes unnecessary if the graph does not contain any black vertices, we conclude that the total number of successful applications of the operations in Rules 1–8 is $O(n)$. Alber, Fellows, and Niedermeier [5] also showed that Rules 1 and 2 can be executed in time $O(n^2)$ when the graph is planar. Similarly, one can show that Rules 3–8 can also be executed in $O(n^2)$ time (we leave the verification of this simple fact to the interested reader). This, together with the fact that the total number of successful applications of all the rules is $O(n)$, implies that the time needed to construct G' is $O(n^3)$.

To show parts (2) and (3) of the theorem, we prove that after the execution of any of the rules, the resulting graph satisfies conditions (2) and (3) in the theorem. The proof will then follow by an inductive argument on the number of applications of the rules. Denote by H the graph before a rule is executed, and by H' the resulting graph after the rule is executed. Denote by D a minimum dominating set for H excluding all white vertices in H. Initially, $H = G$, and all vertices in H are black. Thus, H trivially satisfies conditions (2) and (3) in the theorem. Suppose now that one of the rules is executed on a graph H satisfying conditions (2) and (3) in the theorem to yield the graph H'. We need to show that H' satisfies conditions (2) and (3) as well.

Suppose that Rule 1 is executed. The same argument used in [5] shows that $\gamma(H) = \gamma(H')$.[3] What is left is showing that H' has a minimum dominating set consisting only of black vertices. Let D be a minimum dominating set for H consisting of black vertices. Since $N_3(v) \neq \emptyset$, D must contain a vertex in $N_2(v) \cup N_3(v) \cup \{v\}$. If D contains a vertex in $N_2(v) \cup N_3(v)$, then clearly this vertex can be replaced by v which is black. Thus we can assume that D contains v and no vertex in $N_3(v) \cup N_2(v)$. Then D is also a dominating set for H' consisting only of black vertices, and since $\gamma(G) = \gamma(H) = \gamma(H')$, D is a minimum dominating set for H'. It follows that H' satisfies conditions (2) and (3). The proof of Rule 2 is of the same flavor.

If Rule 3 is executed, then the black vertices in the set $N_2(v) \cup N_3(v)$, where v is black, will be colored white, and the edges between the white vertices are removed. It suffices to show that after the coloring of one vertex x in $N_2(v) \cup N_3(v)$ white and removing the edges between x and the other white vertices, conditions (2) and (3) still hold (the same argument can then be applied repetitively to every such vertex). By our inductive hypothesis, before the application of Rule 3 to H, H had a minimum dominating set D of size equal to $\gamma(G)$ that excludes all white vertices in H. If D contains x, we can replace x by v and have a minimum dominating set of H consisting only of black vertices in H. Thus, we can assume, without loss of generality, that D

[3] The fact that this statement holds true can be easily verified by the reader.

does not include x. Since x is the only vertex whose color has changed to white, D consists only of black vertices in H'. Moreover, it is not difficult to see that D is also a dominating set in H' since the edges removed from H are not used to dominate any vertices in H (these edges were incident on vertices that are not in D). Since by removing edges from the graph the size of the minimum dominating set can only increase, we conclude that D is a minimum dominating set for H' excluding all white vertices, and $\gamma(H') = \gamma(H) = \gamma(G)$.

Suppose Rule 4 is executed. Similarly, we need only show that conditions (2) and (3) still hold after a vertex x has been colored white. If D contains x, then by the assumption in Rule 4, x does not dominate all the vertices in $N_3(v,w)$, and D must also contain at least another vertex x' in $N_2(v,w) \cup N_3(v,w) \cup \{v,w\}$ to dominate $N_3(v,w)$. This is true since only vertices in $N_2(v,w) \cup N_3(v,w) \cup \{v,w\}$ can dominate vertices of $N_3(v,w)$. In such a case we can replace x and x' by v and w and have a minimum dominating set that consists only of black vertices in H. Since x is the only vertex whose color has changed to white, D excludes all white vertices in H'. It is easy to see that the edges that connect white vertices in H are not used by D to dominate any vertex. By an argument similar to the above, it follows that D is a minimum dominating set for H' excluding all white vertices in H', and $\gamma(H') = |D| = \gamma(H) = \gamma(G)$.

Suppose Rule 5 is executed and a vertex x is colored white. Let $R = R(v,w)$ be the quasi-simple region that was being processed in the rule, and note that all the vertices in $R^+(v,w)$ are connected to v. Let the boundary of R be (v,y,w,z,v). Let D be a dominating set for H consisting only of black vertices. If D contains x, then by the assumption in Rule 5, x does not dominate all the vertices in $N_2(v,w) \cup N_3(v,w)$ strictly inside R, and D must contain another black vertex $x' \in R(v,w)$ in $N_2(v,w) \cup N_3(v,w) \cup \{v,w,y,z\}$ to dominate the vertices in $N_2(v,w) \cup N_3(v,w)$ that are strictly inside R. Observe that, by the definition of a quasi-simple region, the only vertex that can be dominated by x and not by v is w. We distinguish the following cases.

Case 1. $x' = v$. Since at least one vertex $r \in \{y,w,z\}$ must be black (w is connected to both y and z, and no edges exist between white vertices; thus it is not possible for all the vertices in $\{y,w,z\}$ to be white) and since all the vertices in $\{y,w,z\}$ dominate w, we can replace x by r (note that x is dominated by v) to obtain a dominating set consisting of black vertices that excludes x.

Case 2. $x' \neq v$. If x' does not dominate w, then x' must be one of those vertices in R^+ that connect only to v and to the vertices on the boundary of R other than w. Since all such boundary vertices are dominated by v, and x' is dominated by v as well, we can replace x' by v in D, and the case reduces to Case 1 above. If x' dominates w, then we can replace x by v to get a dominating set consisting of black vertices that excludes x.

Thus, we can assume, without loss of generality, that D does not include x. Since x is the only vertex whose color has changed to white, D consists only of black vertices in H'. By an argument similar to that above, it follows that D is a minimum dominating set for H' excluding all white vertices in H', and $\gamma(H') = |D| = \gamma(H) = \gamma(G)$.

Suppose Rule 6 is executed and a vertex v is removed as described in the rule. Let D be a minimum dominating set for H excluding all white vertices in H. Thus D does not contain v. Since v is the only vertex removed, and no vertices are colored, it follows that D is a dominating set for H' excluding all white vertices in H'. What is left is proving that D is a minimum dominating set for H'. Suppose that H' has a minimum dominating set D' of size strictly smaller than D. Then D' has to cover u, and hence D' contains either u or a neighbor of u in H'. If D' contains u, since every neighbor of u is also a neighbor of w, and u is a neighbor of w, $(D' \cup \{w\}) - \{u\}$ is a minimum dominating set for H of size smaller than D, a contradiction (note that since w is a neighbor of u, w is a neighbor of v as well and hence dominates v). On the other hand, if D' contains a neighbor of u, since $N(u) \subseteq N(v)$, D' also contains a neighbor of v in H and hence dominates v. Thus, D' is a dominating set for H of size smaller than D, a contradiction. It follows that $|D| = \gamma(H') = \gamma(H) = \gamma(G)$.

Suppose that Rule 7 is executed on a black vertex v and all vertices in $W(v)$ were removed as described in the rule. Let D be a minimum dominating set for H excluding all white vertices in H. Thus, D does not contain any vertex in $W(v)$. Since the vertices in $W(v)$ are the only vertices that were removed, and no vertices in the graph were colored, it follows that D is a dominating set for H' excluding all white vertices in H'. What is left is proving that D is a minimum dominating set for H'. Suppose that H' has a minimum dominating set D' of size strictly smaller than D. Then D' has to cover v. Hence D' contains either v or a neighbor of v in $B(v)$ because all the vertices in $W(v)$ were removed. In either case, D' dominates all the removed vertices in $W(v)$ in H, since every vertex in $W(v)$ is adjacent to all vertices in $B(v)$. Therefore D' is also a dominating set for H of size smaller than D, a contradiction. It follows that $|D| = \gamma(H') = \gamma(H) = \gamma(G)$.

To prove the statement for Rule 8, let D be a minimum dominating set for H excluding all white vertices in H. Again, D is a dominating set for H' excluding all white vertices in H'. Let D' be a minimum dominating set for H' and suppose, to get a contradiction, that $|D'| < |D|$. Without loss of generality, we can assume that D' contains either v or w (or both); otherwise, to dominate u and u', D' has to contain both u and u', which can be replaced by v and w. Now D' is also a dominating set for H of smaller size than D, a contradiction. It follows that D is a minimum dominating set for H' excluding all white vertices in H', and $\gamma(H') = \gamma(H) = \gamma(G)$.

To prove part (4) of the theorem, note the following: (1) from a minimum dominating set for G' one can construct in $O(n)$ time a minimum dominating set for G' containing only black vertices (this can be achieved by associating, during the reduction phase, with the vertices colored white the black vertices that can replace them) and (2) a minimum dominating set for G' consisting only of black vertices is also a minimum dominating set for G. This completes the proof. □

5. A problem kernel. Let G be a reduced graph, and let D be a minimum dominating set for G consisting of black vertices such that $|D| = k$. In this section, we will show that the number of vertices n in G is bounded by $67k$. The following definitions are adopted from [5].

Given any dominating set D in a graph G, a *D-region* decomposition of G is a set $\Re$ of regions between pairs of vertices in D such that the following hold.

1. For any region $R = R(v, w)$ in $\Re$, no vertex in D is in $V(R)$. That is, a vertex in D can only be an endpoint of a region in $\Re$.
2. No two distinct regions R_1, $R_2 \in \Re$ intersect. However, they may touch each other by having common boundaries.

Note that all the endpoints of the regions in a D-region decomposition are vertices in D. For a D-region decomposition $\Re$, define $V[\Re] = \bigcup_{R \in \Re} V[R]$. A D-region decomposition is *maximal* if there is no region R such that $\Re^{'} = \Re \cup R$ is a D-region decomposition with $V[\Re] \subsetneq V[\Re^{'}]$.

For a D-region decomposition $\Re$, associate a planar graph $G_{\Re}(V_{\Re}, E_{\Re})$ with possible multiple edges, where $V_{\Re} = D$, and such that there is an edge between two vertices v and w in $G_{\Re}$ if and only if $R(v,w)$ is a region in $\Re$. A planar graph with multiple edges is called *thin* if there is a planar embedding of the graph such that for any two edges e_1 and e_2 between two distinct vertices v and w in the graph, there must exist two more vertices which sit inside the disjoint areas of the plane enclosed by e_1 and e_2.

Alber, Fellows, and Niedermeier [5] showed that the number of edges in a thin graph of n vertices is bounded by $3n-6$. They also showed that for any plane graph G and a dominating set D of G, there exists a maximal D-region decomposition for G such that $G_{\Re}$ is thin. Since the maximal D-region decomposition in [5] starts with any dominating set D and is not affected by the color a vertex can have, the same results in [5] hold true for our reduced graph G whose vertices are colored black/white, and with a minimum dominating set D consisting of only black vertices. The above discussion is summarized in the following proposition.

PROPOSITION 5.1. *Let G be a reduced graph and D a dominating set of G consisting of black vertices. Then there exists a maximal D-region decomposition $\Re$ of G such that $G_{\Re}$ is thin.*

COROLLARY 5.2. *Let G be a reduced graph with a minimum dominating set D consisting of k black vertices, and let $\Re$ be a maximal D-region decomposition of G such that $G_{\Re}$ is thin. Then the number of regions in $\Re$ is bounded by $3k-6$.*

Proof. The number of regions in $\Re$ is the number of edges in $G_{\Re}$. Since $G_{\Re}$ has $|D| = k$ vertices, by [5], the number of edges in $G_{\Re}$ is bounded by $3k-6$. □

In the remainder of this section, $\Re$ will denote a maximal D-region decomposition of G such that $G_{\Re}$ is thin. Let u and v be two vertices in G. We say that u and v are *boundary-adjacent* if (u,v) is an edge on the boundary of some region $R \in \Re$. For a vertex $v \in G$, denote by $N^*(v)$ the set of vertices that are boundary-adjacent to v. Note that for a vertex $v \in D$, since v is black, by Rule 3, all vertices in $N_2(v) \cup N_3(v)$ must be white. Note also that, by the definition of a D-region decomposition, all the endpoints of the regions in $\Re$ are vertices in D, and hence are colored black.

PROPOSITION 5.3. *Let $v \in D$. The following are true.*

(a) *(Lemma 6, [5].) Every vertex $u \in N_1(v)$ is in $V[\Re]$.*

(b) *The vertex v is an endpoint of a region $R \in \Re$. That is, there exists a region $R = R(x,y) \in \Re$ such that $v = x$ or $v = y$.*

(c) *Every vertex $u \in N_2(v)$ which is not in $V[\Re]$ is connected only to v and to vertices in $N^*(v)$.*

Proof. The proof of part (a) appears in [5].

To prove (b), suppose, to get a contradiction, that v is not the endpoint of any region in $\Re$. Since $v \in D$, and by the definition of a region, v must be outside every region in $\Re$. Now v must have a vertex in $N_1(v)$; otherwise, all vertices in $N(v)$ would be white and hence removed by Rule 7 (we assume, without loss of generality, that G does not contain any isolated vertices). Let $u \in N_1(v)$. By part (a) above, u must belong to some region $R = R(x,y)$. Observe that u must be on the boundary of R; otherwise v would be a vertex in $V[R]$. Again, by the definition of a region, u is either boundary-adjacent to x or to y. Suppose, without loss of generality, that u is boundary-adjacent to x. But then the degenerated region formed by (x,u,v) does not cross $\Re$ (it only touches R), contradicting the maximality of $\Re$.

To prove part (c), let u be a vertex in $N_2(v)$ and note that u is white, and suppose that u is connected to a vertex $w \neq v$ such that $w \notin N^*(v)$. Note that w must be in $N_1(v)$ (otherwise w would be white and u and w cannot be adjacent) and hence, by part (a) above, must belong to some region $R = R(x, y)$. Since $u \notin V[\Re]$, w cannot be inside R and hence is on the boundary of R. Moreover, by the definition of a region, w must be boundary-adjacent to either x or y. Without loss of generality, assume w is boundary-adjacent to x. Now $w \notin N^*(v)$, so w cannot be boundary-adjacent to v, and $x \neq v$. Consider the degenerated region formed by (v, u, w, x). This region cannot cross any region in $\Re$; otherwise it crosses it via (u, w), and u would be in $V[\Re]$. But this contradicts the maximality of $\Re$ since $u \notin V[\Re]$. ▯

Let x be a vertex in G such that $x \notin V[\Re]$. Then by part (b) in Proposition 5.3, $x \notin D$. Thus, $x \in N(v)$ for some black vertex $v \in D \subseteq V[\Re]$. By part (a) in Proposition 5.3, $x \notin N_1(v)$, and hence, $x \in N_2(v) \cup N_3(v)$. By Rule 3, the color of x must be white. Let $R = R(v, w)$ be a region in $V[\Re]$ of which v is an endpoint (such a region must exist by part (b) of Proposition 5.3). We distinguish two cases.

Case A. $x \in N_3(v)$. Since v is black, by Rule 1, this is only possible if $deg(x) = 1$ and $N_2(v) = \emptyset$ (in this case x will be the white vertex added by the rule). In such a case it can be easily seen that we can flip x and place it inside R without affecting the planarity of the graph.

Case B. $x \in N_2(v)$. Note that in this case $N_3(v) = \emptyset$ by Rule 1 (otherwise $N_2(v) \cup N_3(v)$ would be removed), and x is only connected to v and $N^*(v)$ by part (c) of Proposition 5.3. If $deg(x) = 2$, by a similar argument to Case A above, x can be flipped and placed inside R.

According to the above discussion, it follows that the vertices in G can be classified into two categories: (1) those vertices that are in $V[\Re]$ and (2) those that are not in $V[\Re]$, which are those vertices of degree larger than 2 that belong to $N_2(v)$ for some vertex $v \in D$ and in this case must be connected only to vertices in $N^*(v)$. To bound the number of vertices in G we need to bound the number of vertices in the two categories. We start with the vertices in category (2).

Let O denote the set of vertices in category (2). Note that all vertices in O are white, and no two vertices u and v in O are such that $N(u) \subseteq N(v)$. To see why the latter statement is true, note that every vertex in O must be in $N_2(w)$ for some black vertex $w \in D$. So if $N(u) \subseteq N(v)$, then by Rule 6, v would have been removed from the graph. To bound the number of vertices in O, we will bound the number of vertices in O that are in $N_2(v)$ where $v \in D$. Let us denote this set by $N^\dagger(v)$. Let $N^*_\dagger(v)$ be the set of vertices in $N^*(v)$ that are neighbors of vertices in $N^\dagger(v)$. Note that every vertex in $N^\dagger(v)$ has degree ≥ 3 and is connected only to v and to $N^*_\dagger(v)$, and that no two vertices x and y in $N^\dagger(v)$ are such that $N(x) \subseteq N(y)$.

ASSUMPTION 5.4. *For the sake of counting the number of vertices in $N^\dagger(v)$, it is safe to assume that* (1) *every vertex in $N^\dagger(v)$ has degree exactly* 3*;* (2) *no two vertices x and y in $N^\dagger(v)$ are such that $N(x) \subseteq N(y)$; and* (3) *vertices in $N^\dagger(v)$ are only connected to v and to vertices in $N^*_\dagger(v)$.*

Proof. Since properties (2) and (3) are already satisfied by the vertices in $N^\dagger(v)$, we need only show how we can make the vertices in $N^\dagger(v)$ satisfy property (1) without reducing their number and without affecting properties (2) and (3). To satisfy property (1), we will remove some edges between vertices in $N^\dagger(v)$ and $N^*_\dagger(v)$ without affecting the other properties. This can be done as follows. List the vertices in $N^\dagger(v)$ in an arbitrary order $\langle u_1, \ldots, u_r\rangle$. Start by picking u_1; then choose any two neighbors of u_1 in $N^*_\dagger(v)$ and remove all edges that join u_1 to all its neighbors other than v and these two chosen neighbors. Inductively, suppose we have processed vertex u_{i-1}; we process vertex u_i as follows. Pick two neighbors w_1^i and w_2^i of u_i in $N^*_\dagger(v)$ such that no vertex in $\{u_1, \ldots, u_{i-1}\}$ has both w_1^i and w_2^i as its picked neighbors. Delete all the edges that join u_i to all vertices other than v, w_1^i, and w_2^i. We need to show that it is always possible to carry out this step. Suppose not, and let i be the smallest index such that this is not possible. It is easy to verify using the facts that every vertex in $N^\dagger(v)$ has degree larger than 2, no two vertices x and y are such that $N(x) \subseteq N(y)$, and that $i > 3$. Note that it must be the case that $deg(u_i) > 3$; otherwise, since this step cannot be carried out, the only two neighbors of u_i other than v must be neighbors of some other vertex $u_j \in \{u_1, \ldots, u_{i-1}\}$, and hence we would have $N(u_i) \subseteq N(u_j)$ for some $u_j \in \{u_1, \ldots, u_{i-1}\}$, contradicting the properties satisfied by category-(2) vertices as discussed above. Let a, b, c be three neighbors of u_i other than v. Since this step cannot be carried out successfully, there must exist three distinct vertices $u_p, u_q, u_s \in \{u_1, \ldots, u_{i-1}\}$ such that $\{a, b\} \subset N(u_p)$, $\{a, c\} \subset N(u_q)$, and $\{b, c\} \subset N(u_s)$. Consider the subgraph H of G induced by the set of vertices $\{v, u_p, u_q, u_s, u_i, a, b, c\}$. Then the following is true about the vertices in H: (1) $u_i, u_p, u_q, u_s, a, b, c$ are neighbors of v in H; (2) v, a, b, c are neighbors of u_i in H; (3) v, a, b are neighbors of u_p in H; (4) v, a, c are neighbors of u_q in H; (5) v, b, c are neighbors of u_s in H; (6) v, u_i, u_p, u_q are neighbors of a in H; (7) v, u_i, u_p, u_s are neighbors of b in H; and (8) v, u_i, u_q, u_s are neighbors of c in H. Using all this information, it is not difficult to verify that H is nonplanar (identify vertex a with vertex b along the path (a, u_p, b) to obtain a copy of $K_{3,3}$), contradicting the planarity of G. This completes the proof. □

PROPOSITION 5.5. $|N^\dagger(v)| \leq 3/2|N^*_\dagger(v)|$.

Proof. To simplify the counting, by Assumption 5.4, we can assume that every vertex in $N^\dagger(v)$ has degree exactly 3; no two vertices x and y in $N^\dagger(v)$ are such that $N(x) \subseteq N(y)$; and vertices in $N^\dagger(v)$ are connected only to v and $N^*_\dagger(v)$. Let x be the number of vertices in $N^*_\dagger(v)$, and let $f(x) = |N^\dagger(v)|$. We will show that $f(x) \leq 3/2(x-1)$. We proceed by induction on x. If $x = 1$, it is clear that $f(x) = 0 \leq 3/2(x-1)$ since by Assumption 5.4, each vertex in $N^\dagger(v)$ has degree exactly 3. If $x = 2$, then clearly $f(x) \leq 1 \leq 3/2(x-1)$ since at most one vertex can be connected to v and the two vertices in $N^*_\dagger(v)$ without violating properties (1)–(3) in Assumption 5.4. If $x = 3$, then $f(x) \leq 3$ since at most three vertices can be connected to $N^*_\dagger(v)$ without violating properties (1)–(3) in Assumption 5.4, each connected to v and to two other vertices in $N^*_\dagger(v)$. Inductively, suppose that if $N^*_\dagger(v)$ contains y vertices with $3 \leq y < x$, then the number of vertices $f(y)$ in $N^\dagger(v)$ satisfies $f(y) \leq 3/2(y-1)$. Suppose now that $|N^*_\dagger(v)| = x$. Let u be a vertex in $N^\dagger(v)$, and let a, b be its neighbors in $N^*_\dagger(v)$. The vertex u is called *hollow* if the interior of the region enclosed by (u, a, v, b, u) contains no vertices of $N^*_\dagger(v)$. If every vertex in $N^\dagger(v)$ is hollow, then it is clear that $f(x) \leq x \leq 3/2(x-1)$ for $x > 3$, and the bound $f(x) = x$ is attained when there are x vertices in $N^\dagger(v)$, and every vertex u in $N^\dagger(v)$ is adjacent to v and the two neighbors a and b in $N^*_\dagger(v)$ immediately to the left and right in the clockwise (or counterclockwise) ordering, respectively, of u in the embedding. Suppose now that there is a vertex $u \in N^\dagger(v)$ such that u is not hollow. The edges (u, a), (u, v), (u, b), (v, a), (v, b) separate the plane into three faces: F_1 enclosed by the cycle (u, a, v, u), F_2 enclosed by the cycle (u, v, b, u), and F_3, the outer face determined by the cycle (u, a, v, b, u). Let x_1 be the number of vertices in $N^*_\dagger(v)$ that are in F_1 including the boundary, x_2 the number in F_2, and x_3 the number in F_3. Note that $1 \leq x_1 < x$ since $a \in F_1$ and $b \notin F_1$, $1 \leq x_2 < x$ since $b \in F_2$ and $a \notin F_2$, and $2 \leq x_3 < x$ since a and b are in F_3 and at least one vertex in $N^*_\dagger(v)$ is not in F_3 since u is hollow, and hence the interior of the face (u, a, v, b, u) contains at least one vertex in $N^*_\dagger(v)$. Moreover, $x_1 + x_2 + x_3 = x + 2$, since a and b are the only vertices counted twice when we add the vertices in $N^*_\dagger(v)$ that are in F_1, F_2, and F_3. Now every vertex in $N^\dagger(v)$ is either (1) connected to two vertices in $N^*_\dagger(v)$ in F_1, (2) connected to two vertices in $N^*_\dagger(v)$ in F_2, or (3) connected to two vertices in $N^*_\dagger(v)$ in F_3. Note that vertex u satisfies property (3). Since $x_1, x_2, x_3 < x$, by the inductive hypothesis, the number of vertices satisfying (1) is bounded by $f(x_1) \leq 3/2(x_1 - 1)$, the number of vertices satisfying (2) is bounded by $f(x_2) \leq 3/2(x_2 - 1)$, and the number of vertices satisfying (3) is bounded by $f(x_3) \leq 3/2(x_3 - 1)$. Now $f(x) \leq f(x_1) + f(x_2) + f(x_3) \leq 3/2(x_1 + x_2 + x_3) - 9/2 = 3x/2 - 3/2 = 3/2(x - 1)$. This completes the proof. □

LEMMA 5.6. *The number of vertices in category* (2) *(i.e., the number of vertices not in* $V[\Re]$*) is bounded by* $18k$.

Proof. Let v and w be any two distinct vertices in D and observe the following. First, $N^\dagger(v) \cap N^\dagger(w) = \emptyset$, because if $u \in N^\dagger(v) \cap N^\dagger(w)$, then (v, u, w) would be a degenerated region with $u \notin V[\Re]$ contradicting the maximality of $\Re$. Second, from the first observation it follows that $w \notin N^*_\dagger(v)$ and $v \notin N^*_\dagger(w)$, and in general no vertex $a \in D$ belongs to $N^*_\dagger(b)$ for any vertex $b \in D$; otherwise, there exists a vertex $u \in N^\dagger(v)$ that is connected to w, and hence $u \in N^\dagger(v) \cap N^\dagger(w)$, contradicting the first observation. Third, $N^*_\dagger(v) \cap N^*_\dagger(w) = \emptyset$; otherwise, there exists a vertex $u \in N^*_\dagger(v) \cap N^*_\dagger(w)$ that is connected to a category-(2) vertex $a \in N^\dagger(v)$ (or $b \in N^\dagger(w)$), and the degenerated region (v, a, u, w) (or (w, b, u, v)) would contain the vertex $a \notin \Re$ (or $b \notin \Re$), contradicting the maximality of $\Re$.

Let B be the number of vertices not in D that are boundary-adjacent to vertices in D (i.e., in $N^*(v) - D$ for some $v \in D$). Combining the above observations with Proposition 5.5, it follows that the number of category-(2) vertices is

$$\sum_{v \in D} |N^\dagger(v)| \leq \frac{3}{2} \sum_{v \in D} |N^*_\dagger(v)| \leq 3B/2.$$

According to the definition of a region, each region in $\Re$ has at most six vertices on its boundary, two of which are vertices in D. Thus, each region in $\Re$ can contribute with at most four vertices to B. Note that from the above discussion no vertex $a \in D$ belongs to $N^*_\dagger(b)$ for any vertex $b \in D$, and hence the endpoints of the regions do not contribute to B. By Corollary 5.2, the number of regions in $\Re$ is bounded by $3k - 6$. It follows that $B \leq 12k - 24$, and hence the number of category-(2) vertices is bounded by $18k - 36 < 18k$. This completes the proof. □

To bound the number of vertices in category (1), fix a region $R(v, w)$ between $v, w \in D$. We have the following lemma whose proof is technical and based on case-by-case structural analysis. The proof of the lemma appears in the appendix.

LEMMA 5.7 (see Lemma A.5 in the appendix). *Let* $R = R(v, w)$ *be a region in* $V[\Re]$. *The number of vertices in* $V(R)$ *is bounded by* 16.

THEOREM 5.8. *The number of vertices in* G *is bounded by* $67k$.

Proof. By Lemma 5.6, the number of category-(2) vertices in G is bounded by $18k$. Using this bound, we can assume that each region in $\Re$ is nice. By Corollary 5.2, the number of regions in $\Re$ is bounded by $3k - 6$. According to Lemma 5.7, the number of vertices in $V(R)$, where $R \in \Re$ is a nice region, is bounded by 16. It follows that the number of vertices in $V(\Re)$ is bounded by $48k - 96$. Thus, the number of vertices in $V[\Re]$, and hence in category (1), is bounded by $48k - 96$ plus the number of vertices in D which are the endpoints of the regions in $\Re$. Therefore the number of vertices in $V[\Re]$ is bounded by $49k - 96$, and the total number of vertices in G is bounded by $67k - 96 < 67k$. This completes the proof. □

THEOREM 5.9. *Let G be a planar graph with n vertices. Then in time $O(n^3)$, computing a dominating set for G of size bounded by k can be reduced to computing a dominating set of size bounded by k for a planar graph G' of $n' < n$ vertices, where $n' \leq 67k$.*

Proof. According to Theorem 4.4, in time $O(n^3)$ we can construct a reduced graph G' from G, where $\gamma(G') = \gamma(G)$, and such that a dominating set for G can be constructed from a dominating set for G' in linear time. Moreover, the graph G' has no more than n vertices. If G has a dominating set of size bounded by k, then G' has a dominating set of size bounded by k (since $\gamma(G) = \gamma(G')$), and by Theorem 5.9, we must have $n' \leq 67k$. If this is the case, then we can work on computing a dominating set for G' of size bounded by k, from which a dominating set for G can be easily computed. If this is not the case, then G does not have a dominating set of size bounded by k, and the answer to the input instance is negative. This completes the proof. □

6. A simple algorithm. In this section we present a simple algorithm for determining whether a graph G has a dominating set of size bounded by k.

Let $G = (V, E)$ be a planar graph given with an embedding in the plane. The *layer decomposition* of G with respect to the embedding is a partitioning of V into disjoint layers $(L_1, \ldots, L_r)$ defined inductively as follows. Layer L_1 is the set of vertices that lie on the outer face of G, and layer L_i is the set of vertices that lie on the outer face of $G - \bigcup_{j=1}^{i-1} L_j$ for $1 < i \leq r$. It is well known that a layer decomposition of a planar graph G can be computed in linear time in the number of vertices in the graph [4].

A *separator* in a graph G is a set of vertices S whose removal disconnects G. If $(L_1, \ldots, L_r)$ is a layer decomposition of G, then clearly the vertices in any layer L_i form a separator in G, separating the vertices in layers $L_1, \ldots, L_{i-1}$ from those in layers $L_{i+1}, \ldots, L_r$. Let (G, k) be an instance of the PLANAR-DS problem. By Theorem 5.9, we can assume that G is reduced and that the number of vertices n of G satisfies $n \leq 67k$. Let $(L_1, \ldots, L_r)$ be a layer decomposition of G. Let $c > 0$ be a constant which will be determined later, and set $l = \lceil c\sqrt{k} \rceil$. Consider the families of layers F_i, $i = 1, \ldots, l$, where F_i consists of layers $L_i, L_{i+l}, L_{i+2l}, \ldots$. Assume for now that the number of layers $r \geq l$. We will show later how to handle the situation when this is not the case. The families F_i, $i = 1, \ldots, l$, are disjoint, and each family forms a separator separating the graph into connected components that will be called *chunks*, where each chunk consists of at most l consecutive layers. Since these l families are disjoint and partition the layers into l groups, and since the graph has at most $67k$ vertices, there exists an index $1 \leq \mu \leq l$, such that the number of vertices in F_μ is bounded by $67k/l$. Again, observe that the removal of F_μ from G separates G into chunks, each consisting of at most l consecutive layers. Let these chunks be $G_1, \ldots, G_s$.

The basic idea behind the algorithm is to apply a simple divide-and-conquer strategy by removing the vertices in the family F_μ to split the graph into chunks, then to compute a minimum dominating set for the resulting chunks using the algorithm introduced in [33], which is a variation of Baker's algorithm [8]. To do this, for each vertex v in the F_μ, we "guess" whether v is in the minimum dominating set for G or not (basically, what we mean by guessing is enumerating all sequences corresponding to the different possibilities). For each guess of all the vertices in F_μ, we will solve the corresponding instance with respect to that guess. It was shown in [33] how this guessing process can be achieved using at most three statuses per vertex. Hence, guessing the vertices in F_μ can be done by enumerating at most $3^{|F_\mu|} \leq 3^{67k/l}$

ternary sequences. After guessing each vertex in the separator and updating the graph accordingly, the instance becomes an instance of a variation of the minimum dominating set problem due to the constraints placed on some of the vertices in the graph. Kanj and Perković introduced an algorithm in [33], which is a variation of Baker's algorithm [8], to solve this problem. The algorithm introduced in [33] solves this problem on the chunks in time $O(27^{d+1}n)$, where d is the maximum number of layers in a chunk (i.e., the maximum depth of a chunk). Noting that $d \leq l$ and that $n \leq 67k$, we conclude that after guessing all the vertices in F_μ, the problem can be solved in time $O(27^l k)$. If the number of layers r in G is less than l, we can simply call the algorithm in [33] directly on G to solve the problem in time $O(27^l k)$. The algorithm is given in Figure 3 below.

It is not difficult to verify that the running time of the algorithm is $O(3^{67k/l} \cdot 27^l \cdot k + n^3)$, where the $O(n^3)$ time is the time taken to reduce G to its kernel. Niedermeier and Rossmanith showed how to get rid of the k factor corresponding to the kernel size in the running time of such algorithms [36]. Using their techniques, the running time of the algorithm becomes $O(n^3 + 3^{67k/l} \cdot 27^l)$. We choose c, and hence l, so that the above expression is minimized. It can be shown that the expression is minimized when $c = \sqrt{67/3}$, and the running time of the algorithm becomes $O(n^3 + 2^{45\sqrt{k}})$.

```
Algorithm. DS-SOLVER
Input: a planar graph $G$ of $n$ vertices, and a parameter $k$
Output: a dominating set $D$ of size $\leq k$ in case it exists;

1. use the results in Theorem 5.9 to kernelize $G$;
2. if the number of vertices $n$ of $G$ is $> 67k$ then
       Stop("$G$ does not have a dominating set of size $\leq k$");
3. let $c = \sqrt{67/3}$; $l = \lceil c\sqrt{k} \rceil$;
4. if the number of layers in $G$ is $< l$ then
       use the algorithm in [33] to solve the problem in time $O(2^{45\sqrt{k}}k)$; Stop;
5. let $F_\mu$ be a separator of size $\leq 67k/l$ separating the graph into chunks
   $G_1, \ldots, G_s$ each consisting of at most $l$ consecutive layers;
6. for each assignment to the vertices in $F_\mu$ do
       update $D$;
       split the graph into its components;
       compute a minimum dominating set $D'$ for the resulting graph using the
       algorithm in [33];
       $D = D \cup D'$;
7. output the smallest dominating set constructed in step 6 in case its size is
   bounded by $k$; otherwise return ("$G$ does not have a dominating set of size
   $\leq k$").
```

FIG. 3. *A simple algorithm solving* PLANAR-DS.

THEOREM 6.1. *In time $O(n^3 + 2^{45\sqrt{k}})$, it can be determined whether or not a planar graph on n vertices has a dominating set of size bounded by k.*

Theorem 6.1 shows that our algorithm for solving the PLANAR-DS problem is competitive with the previous algorithms using the similar technique of layer decomposition of a planar graph [4, 33]. The above algorithm improves the original $O(2^{70\sqrt{k}}n)$ time algorithm given in [4] for the problem. At the same time, our algorithm is much simpler than the algorithms in [4, 23, 25, 33], illustrating the power of kernelization in the process of designing efficient algorithms for parameterized $\mathcal{NP}$-hard problems. Finally, after a kind of race resulting in better and better algorithms [4, 23, 25, 33], Fomin and Thilikos recently presented an $O(n^3 + 2^{15.13\sqrt{k}})$ time algorithm to solve the PLANAR-DS problem based on the concept of branchwidth [25], the best treewidth based algorithm being only slightly worse [23].

7. Summary and extensions. In this paper we exhibited the first lower bound results on kernel sizes and, motivated by these findings, we strived to improve on the (still huge) constants involved in the known linear kernel for PLANAR-DS.

Are there other, possibly more sophisticated arguments for showing lower bounds on kernel sizes? In particular, it would be interesting to have arguments ruling out, for example, the existence of a kernel of size $o(k^3)$ in a situation when a kernel of size $\mathcal{O}(k^3)$ has been obtained. The algebra we used in the proof of Theorem 3.1 does not extend to such cases.

We mention that the concept of a black-and-white graph we used for deriving the kernel upper bound results for PLANAR-DS also allows us to exhibit a small kernel for a variation of the DOMINATING SET problem, called the RED/BLUE DOMINATING SET problem, as studied by Weihe [41]: Given a graph $G = (V, E)$, with V partitioned into $V_{\text{red}} \cup V_{\text{blue}}$, and a positive integer k, is there a *red/blue dominating set* $D \subseteq V_{\text{red}}$ with $|D| \leq k$, i.e., $V_{\text{blue}} \subseteq N(D)$? Namely, if we consider the red vertices as "black" in our notation and the blue vertices as "white," and if we reanalyze our reduction rules, we can state the following corollary.

COROLLARY 7.1. PLANAR RED/BLUE DOMINATING SET *admits a problem kernel of size* $67k$.

As exhibited in [4, 23], the possibly better known FACE COVER problem can be solved with the help of PLANAR RED/BLUE DOMINATING SET, by introducing "face vertices." However, we are still investigating if we could claim a small kernel for FACE COVER, since we are not keeping the (face) structure of the original problem. Notice that a cubic kernel was derived in [1, Thm. 1]. Based on this sort of problem kernel, we could then arrive at an $O^*(c^{\sqrt{k}})$ algorithm for FACE COVER that is significantly better than what was obtained in [4], close to being competitive with [23], along the lines of the preceding section.

Appendix. In this section we prove Lemma 5.7. We first start with some observations and preliminary results.

Fix a region $R(v, w)$ between $v, w \in D$. Without loss of generality, assume the boundary of R is determined by the two paths (v, v_1, w_1, w) and (v, v_2, w_2, w). Note that all vertices in $V(R)$ belong to $N(v, w)$, and that v_1, $v_2 \in N^*(v)$ and $w_1, w_2 \in N^*(w)$. If there is a degree-1 vertex x connected to v (resp., w), then this vertex is in $N_3(v)$ (resp., $N_3(w)$) and must be colored white by Rule 3. Similarly, if there exists a degree-2 vertex y that is connected to v and either v_1 or v_2 (resp., w and either w_1 or w_2), then y is in $N_2(v)$ (resp., $N_2(w)$) and must be colored white by Rule 3. Now if a degree-1 white vertex is connected to v, then since the vertices in $N^\dagger(v)$ are white and are neighbors of v, by Rule 6 we must have $N^\dagger(v) = \emptyset$. During the process of counting the number of vertices in $N^\dagger(v)$, we bounded the number of vertices in $N^\dagger(v)$ by $3|N^*(v)|/2$. This can be looked at as each vertex in $N^*(v)$ contributing 3/2 vertices to $|N^\dagger(v)|$. So if a degree-1 white vertex is connected to v (note that at most one degree-1 vertex can be connected to v), this means that $N^*(v)$ which contains at least two vertices will not contribute to the number of vertices in $N^\dagger(v)$, and hence, the bound on $|N^\dagger(v)|$ will be decreased by at least three. Similarly, if a degree-2 white vertex is connected to v and v_1 (or v_2) (again, note that there can be at most one degree-2 vertex connected to v and to v_1 (or v_2)), then no vertex in $N^\dagger(v)$ can be connected to v_1 (or v_2). This can be regarded as a reduction to the bound on $|N^\dagger(v)|$ by 3/2. Thus, if we use the upper bound on the number of vertices in category (2) computed above, we may assume without loss of generality that no degree-1 vertex is connected to v or w and that no degree-2 vertex is connected to v and v_1, v and v_2,

w and w_1, or w and w_2. We will also assume that the boundary of a region $R(v, w)$ consists of exactly six distinct vertices; that is, the region is not a degenerated region. The case of a degenerated region obviously yields a better bound on the number of vertices in the region. Let us call a region with all the above properties *nice*. We start with the following propositions.

Proposition A.1. *If there is no simple black vertex strictly inside a quasi-simple region $R = R(v, w)$, then $V(R)$ contains at most two simple white vertices.*

Proof. Suppose, to get a contradiction, that $V(R)$ contains more than two simple white vertices, and let a, b, and c be three such vertices. Since all three vertices are simple, one vertex must be engulfed within the area determined by v, w, and the other two vertices. Suppose that b is situated within the area (v, a, w, c, v). Since, by the assumption of the proposition, all the simple vertices strictly inside $V(R)$ must be white, and since all the nonsimple vertices inside $V(R)$ (i.e., vertices in R^+) are white by definition, and since no edges exist between white vertices, it follows that the white simple vertex b, engulfed by the area (v, a, w, c, v), is connected only to v and w and hence has degree 2. Note that the color of both v and w must be black since there are simple white vertices that are connected to both v and w. Now $|W(v, w)| > 2$ because $\{a, b, c\} \subseteq W(v, w)$. But this makes Rule 8 applicable, contradicting the fact that G is reduced. This completes the proof. ∎

Proposition A.2. *Let $R = R(v, w)$ be a quasi-simple region where the color of v is black; then $V(R)$ has at most four simple vertices.*

Proof. Suppose first that R has six or more simple vertices. Let S be the set of those simple vertices that are strictly inside R. Then $|S| \geq 4$. Since the vertices in S are simple and hence connect to both v and w, it is obvious that no vertex lying strictly inside R can dominate all vertices in S. Since S is a subset of those vertices in $N_2(v, w) \cup N_3(v, w)$ that are strictly inside R, it follows that no vertex that is strictly inside R can dominate all vertices in $N_2(v, w) \cup N_3(v, w)$. Now all vertices that lie strictly inside R belong to $N_2(v, w) \cup N_3(v, w)$; thus, by Rule 5, all vertices strictly inside R must be white. Noting that $|S| \geq 4$, and that all the vertices in S are simple white vertices, this contradicts Proposition A.1.

Suppose now that R has five simple vertices. Let a, b, and c be the three simple vertices that lie strictly inside R. By an argument similar to the above, we can assume that vertex b is engulfed within the area determined by v, a, w, and c. Again all the vertices strictly inside R must belong to $N_2(v, w) \cup N_3(v, w)$. Since a does not dominate c, and vice versa, it follows that a and c are colored white by Rule 5. Now a, b, and c are the only simple vertices strictly inside R, but by Proposition A.1, no three simple white vertices can be contained in R. This forces b to be black and to be connected to both a and c (otherwise b would be colored white by Rule 5). Now all other nonsimple vertices in R must be connected to the boundary and hence cannot be connected to b (all the vertices other than a and c which can be connected to b have to belong to the area engulfed by (v, a, w, c) and cannot be connected to the boundary). Thus, $W(b) = \{a, c\}$, and every vertex in $W(b)$ is connected to all vertices in $B(b) = \{v, w\}$ (note that since a and c are white and are connected to w, w must be black). By Rule 7, $W(b) = \{a, c\}$ should have been removed at this point, a contradiction. Therefore, R has at most four simple vertices and the proof is complete. ∎

Proposition A.3. *Let (v, y, w, z, v) be the boundary of a quasi-simple region $R = R(v, w)$, and suppose that v and y are black. Then there can be at most one white vertex in $R^+ = R^+(v, w)$ that is connected to both v and y.*

Proof. Suppose, to get a contradiction, that there are at least two white vertices in R^+ that are connected to both v and y. Since all the vertices in R^+ are white and hence cannot be connected together, there must exist two white vertices a and b in R^+ satisfying that the area engulfed by (v, a, y, v) is empty, and that the area engulfed by (v, b, y, v) contains only the vertex a. Clearly, the degree of a is exactly 2, and a belongs to $N_2(v) \cup N_3(v)$. Now since both a and b are connected to both v and y, we have $N(a) \subseteq N(b)$. Given the fact that v is black, this is a contradiction to Rule 6. □

Proposition A.4. *Let $R = R(v, w)$ be a quasi-simple region, and suppose that v is black. Let (v, y, w, z, v) be the boundary of R. If*

(a) *there are no simple vertices strictly inside R or*

(b) *there are simple vertices strictly inside R and all vertices in $R^+ = R^+(v, w)$ are connected to y,*

then $V(R) \cup \{w\}$ contains at most three white vertices. Moreover, if there are three white vertices in $V(R) \cup \{w\}$, then either $R^+ \neq \emptyset$ or there is a simple black vertex interior to R.

Proof. To prove that part (a) implies the statement of the proposition, suppose that there are no simple vertices lying strictly inside R. Then clearly all the white vertices in $V(R)$ come from $R^+ \cup \{y, z\}$. If y is white, then no vertex in R^+ can be connected to y because the vertices in R^+ are all white. On the other hand, since $R^+ \subseteq N_2(v) \cup N_3(v)$, if y is black, by Proposition A.3, at most one white vertex in R^+ can be connected to y, and similarly if z is black. Since every vertex in R^+ has to be connected to either y or z by the definition of a quasi-simple region, it follows from the above that $V(R)$ contains at most two white vertices, and hence $V(R) \cup \{w\}$ contains at most three white vertices. Now when $V(R) \cup \{w\}$ contains three white vertices, w must be white, and hence, y and z are black. Thus, the two white vertices other than w in $V(R) \cup \{w\}$ come from R^+, and $R^+ \neq \emptyset$.

To prove part (b), note first that, by Proposition A.2, the number of simple vertices in R including y and z is bounded by four. We will assume that the number of simple vertices in R is exactly four. The cases when there are less than four simple vertices in R are simpler and yield the desired bound. Let a and b be the other two simple vertices, and assume that the four simple vertices y, a, b, z appear in the preceding sequence in a clockwise order around v. Observe that the white vertices in $V(R)$ come from $R^+ \cup \{y, a, b, z\}$. Also observe that since all the vertices in R^+ are connected to y by the hypothesis of part (b), either y is white and R^+ is empty or y is black and by Proposition A.3, R^+ contains at most one vertex. It follows that the number of white vertices in $R^+ \cup \{y\}$ is bounded by one. Now suppose to get a contradiction that $V(R) \cup \{w\}$ contains four white vertices. Since no two white vertices are connected and since all vertices in $\{a, b, z\}$ are connected to w, w must be black, and all three vertices a, b, and z must be white. But then the degree of b is exactly 2, and $|W(v, w)| > 2$, contradicting Rule 8. To complete the proof, supposing that $V(R) \cup \{w\}$ contains exactly three white vertices, we need to show that either $R^+ \neq \emptyset$ or there exists a simple black vertex inside R. Suppose, to get a contradiction, that $R^+ = \emptyset$ and the interior vertices to R, a, b, are all white. Then w must be black in this case, and either y or z is white. Without loss of generality, assume y is white. Since there are no edges between white vertices, the degree of a must be 2, and $\{y, a, b\} \subseteq W(v, w)$, again a contradiction to Rule 8. □

Lemma A.5 (see Lemma 5.7). *Let $R = R(v, w)$ be a nice region in $V[\Re]$. The number of vertices in $V(R)$ is bounded by* 16.

Proof. Every vertex in $V(R)$ is in $N(v,w)$, and hence is connected to either v or w. We distinguish two cases.

Case 1. $N_3(v,w)=\emptyset$. In this case every vertex in $V(R)-\{v_1,v_2,w_1,w_2\}$ has to be connected to at least one vertex in $\{v_1,v_2,w_1,w_2\}$ because v_1,v_2,w_1,w_2 are the only vertices in $V(R)$ that possibly belong to $N_1(v,w)$. Since R is nice, the vertices in $V(R)-\{v_1,v_2,w_1,w_2\}$ can be classified into the following categories, where a vertex is assigned to the first category that it satisfies:

(i) vertices connected to v and v_1, but not connected to w_1;
(ii) vertices connected to v and w_1;
(iii) vertices connected to v and w_2;
(iv) vertices connected to v and v_2;
(v) vertices connected to w and w_2, but not connected to v_2;
(vi) vertices connected to w and v_2;
(vii) vertices connected to w and v_1; and
(viii) vertices connected to w and w_1.

Note that one of categories (ii) and (vii) must be empty; otherwise, according to our placement of the vertices in the categories, we have two distinct vertices in $V(R)$ other than v_1 and w_1: one of them is connected to v and w_1 and the other to w and v_1, contradicting the planarity of the graph. Similarly, one of categories (iii) and (vi) must be empty. Without loss of generality, assume that categories (iii) and (vii) are empty. If, in addition, either category (ii) or category (vi) is empty, then the situation becomes simpler, leading to a better bound on the number of vertices in $V(R)$. Thus, we will assume that both categories (ii) and (vi) are nonempty. Note also that since R is nice, no vertex interior to R has degree 2, and every vertex in category (i) must be connected to some vertex interior to R. Since category (ii) is nonempty, and by the planarity of G, a vertex in category (i) can be connected only to vertices in category (ii). Moreover, since the vertices in category (i) are connected only to v and to neighbors of v including v_1 (since these vertices can be connected only to v and to vertices in category (ii)), all these vertices belong to $N_2(v)\cup N_3(v)$. Since v is black, by Rule 3, all vertices in category (i) must be white. Now the vertices in categories (i) and (ii), plus v, v_1, and w_1, form a quasi-simple region between v and w_1, $Q_1=R(v,w_1)$. Moreover, all the vertices in $V(Q_1)$, except those in category (i), are simple vertices because all vertices in $V(Q_1)$, except those in category (i), have to be connected to both v and w_1. Since v is black (all the endpoints of regions in $\Re$ are black), by Proposition A.2, the number of vertices in $V(Q_1)$ except those vertices in category (i) is bounded by 4. Now we bound the number of vertices in category (i). Every vertex in category (i) is white and is connected to v and v_1. If category (i) is nonempty, then v_1 must be black, and the vertices in category (i) are white vertices in $Q_1^+(v,w_1)$ that are connected to v and v_1. It follows from Proposition A.3 that the number of vertices in category (i) is bounded by 1. This shows that the number of vertices in $V(Q_1)$ is bounded by 5. By symmetry, the number of vertices in $V(Q_2)$, where Q_2 is the quasi-simple region between w and v_2 consisting of the vertices in categories (v) and (vi), plus the vertices w, w_2, v_2, is bounded by 5. Now we bound the number of vertices in categories (iv) and (viii). We have the following claim.

Claim. The number of vertices in category (iv) (resp., category (viii)) is bounded by 2. Moreover, at most one vertex in category (iv) (resp., category (viii)) is white.

Consider the vertices in category (iv). Suppose that there are three or more vertices in category (iv), and let a_1, a_2, a_3 be three vertices in category (iv) such that no vertex is engulfed in the area of the embedding determined by (v,a_1,v_2), a_1

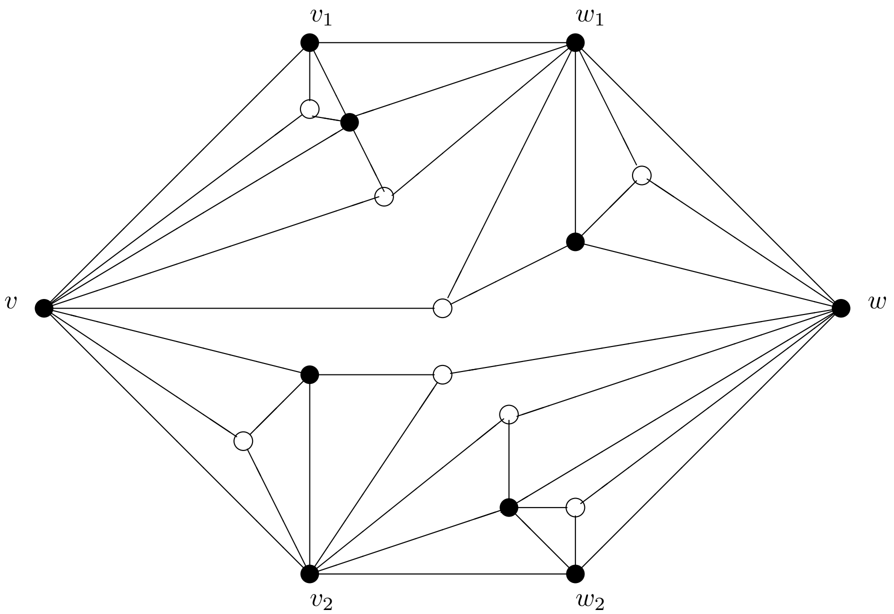


FIG. 4. *Illustration of a possible worst-case scenario for Case* 1. *Empty circles represent white vertices, and filled circles black vertices.*

is the only vertex engulfed in the area determined by (v, a_2, v_2), and a_1 and a_2 are the only two vertices engulfed in the area determined by (v, a_3, v_2). Now a_1 and a_2 must belong to $N_2(v) \cup N_3(v)$. Since v is black, it follows from Rule 3 that a_1 and a_2 must be white and that no edge exists between a_1 and a_2. But this means that $N(a_1) \subseteq N(a_2)$, and since $a_1 \in N_2(v)$ and v is black, according to Rule 6, this leads to a contradiction. It follows that at most two vertices can be in category (iv). By symmetry, at most two vertices can be in category (viii). Note also that it follows from the above proof that if there are exactly two vertices a_1 and a_2 in category (iv) (resp., category (viii)), then at most one vertex in $\{a_1, a_2\}$ can be white. This proves the claim.

Now the vertices in $V(R)$ consist of vertices of $V(Q_1)$, vertices of $V(Q_2)$, category (iv) and category (viii) vertices, and the two vertices v_2 and w_1 (in case these two vertices were not included in any of the other categories). It follows that the number of vertices in R is bounded by 16. See Figure 4 for an illustration of such a possible scenario.

Case 2. $N_3(v, w) \neq \emptyset$. Let X be the set of white vertices in $N_2(v, w)$ that are in $V(R)$, Y the set of black vertices in $N_2(v, w) \cup N_3(v, w)$ that are in $V(R)$, and Z the set of white vertices in $N_3(v, w)$ that are in $V(R)$. We first make a few observations.

Observation 1. $|X| \leq 7$. We first show that $|X| \leq 8$. Remove the vertices in $N_3(v, w)$ interior to R; then define categories (i)–(viii) as above. Similar to Case 1, we can assume that the vertices in categories (i) and (ii), plus the vertices v, v_1, and w_1, form a quasi-simple region $Q_1 = R(v, w_1)$, and that those in categories (v) and (vi), plus the vertices w, w_2, and v_2, form a quasi-simple region $Q_2 = R(w, v_2)$. Since $X \subseteq N_2(v, w)$, every vertex in X must belong to one of categories (i)–(viii) or possibly to the set $\{v_1, v_2, w_1, w_2\}$. From the definition of categories (i) and (ii), vertices in category (i) form the set $Q_1^+ = Q_1^+(v, w_1)$ in the quasi-simple region Q_1, and all the vertices in Q_1^+ are connected to v_1. Now add the vertices in $N_3(v, w)$ back, and note that no black vertex in $N_3(v,w)$ that is not connected to w_1 resides in Q_1. The reason is that such a vertex would be in $N_2(v) \cup N_3(v)$ (otherwise, this vertex will have to be connected to w_1—the only vertex in Q_1 possibly not in $N(v)$) and hence colored white by Rule 3. Now Q_1 plus the set of black vertices in $N_3(v,w)$ that reside in Q_1, minus the set of white vertices in $N_3(v,w)$ that reside in Q_1, satisfies condition (b) in Proposition A.4, and the number of white vertices in Q_1 is bounded by 3. Since no two white vertices are connected together and hence the presence of the white vertices from $N_3(v,w)$ in Q_1 cannot increase the number of possible white vertices in Q_1, we conclude that the number of white vertices in Q_1 that are not in $N_3(v,w)$, and hence, the number of vertices in X that belong to Q_1, is bounded by 3. Similarly, the number of vertices in X that belong to Q_2 is bounded by 3. Moreover, the statement of the claim in Case 1 carries in a straightforward manner to Case 2, and categories (iv) and (viii) contain at most one white vertex each. It follows that the number of white vertices in the set X is bounded by 8. Now if $|X| = 8$, then both Q_1 and Q_2 (plus the black vertices in $N_3(v,w)$ that reside in Q_1 and Q_2) contain three white vertices. Since Q_1 contains exactly three white vertices, by Proposition A.4, either $Q_1^+ \neq \emptyset$, or Q_1 must contain an interior black vertex. If $Q_1^+ \neq \emptyset$, since R is nice, the vertex in Q_1^+ must be connected to some vertex interior to Q_1 which must be black because the vertices in Q_1^+ are white. Therefore, if Q_1 contains exactly three white vertices, then there must exist an interior black vertex p in Q_1. Similarly, there must exist an interior black vertex q in Q_2. Since both p and q are black and are in $N_2(v,w) \cup N_3(v,w)$, by Rule 4, p and q must dominate all vertices in $N_3(v,w) \neq \emptyset$. In particular, p which is interior to Q_1 must dominate q which is interior to Q_2. This is a contradiction to the planarity of the graph. It follows that $|X| \leq 7$.

Observation 2. Every vertex in Y must dominate all vertices in $N_3(v,w)$.

This observation follows from Rule 4 since the vertices in Y are black and are a subset of $N_2(v,w) \cup N_3(v,w)$.

Let H be the graph obtained from G by identifying the vertex v with w along the path (v, v_1, w_1, w). Clearly, H is planar. Let u be the resulting vertex by this identification. Let Y' be the set of vertices in Y that are in H, and let $y = |Y'|$. Similarly, let Z' be the set of vertices in Z that are in H, and let $z = |Z'|$. Observe that the vertex u is connected to all the vertices in Y' and Z' in H, and that the only vertices that have been removed by this identification are boundary vertices to R that belong to $\{v_1, v_2, w_1, w_2\}$.

Observation 3. If $y > 1$ and $z > 1$, then the number of vertices in $V(R)$ is bounded by 16.

Suppose that $y > 1$ and $z > 1$. If $y > 2$, since every vertex in Y' must dominate the vertices in Z', it follows that the subgraph of H induced by the set of vertices $Y' \cup Z' \cup \{u\}$ contains a copy of $K_{3,3}$, contradicting the planarity of H (the vertices in Y' form the first bipartition, and the other vertices form the second bipartition). Suppose now that $y = 2$. If $z > 2$, then similarly, the subgraph induced by $Z' \cup \{u\} \cup Y'$ contains a copy of $K_{3,3}$ (the vertices in $Y' \cup \{u\}$ form the first bipartition, and those in Z' form the second bipartition). Suppose now that $y = z = 2$. Then the number of vertices in $X \cup Y' \cup Z'$ is bounded by 11. Since $|V(R)| \leq |X \cup Y' \cup Z' \cup \{v_1, v_2, w_1, w_2\}|$, it follows that the number of vertices in $V(R)$ is bounded by 16.

Now we distinguish the following two subcases.

Subcase 2.1. $z \leq 1$. Let $Y_1 = Y' \cap N_2(v,w)$ be the set of black vertices in Y' that are in $N_2(v,w)$, $y_1 = |Y_1|$, $Y_2 = Y' - Y_1$ be the set of black vertices in Y' that are in $N_3(v,w)$, and $y_2 = |Y_2|$. Note that every vertex in Y' must be connected to all vertices in $Y_2 \cup Z$ by Rule 4. If $y = y_1 + y_2 < 5$, then since $z \leq 1$ and the number of vertices in X is bounded by 7 by Observation 1, the number of vertices in $V(R)$ is bounded by 16. So we can assume that $y \geq 5$. If $y_2 + z \geq 4$, then the subgraph induced by the vertices $\{u\} \cup Y_2 \cup Z$ is a copy of K_5. Thus, $y_2 + z < 4$. If $y_2 + z = 3$, then the subgraph induced by the bipartition $(Y_2 \cup Z, Y_1 \cup \{u\})$ contains a copy of $K_{3,3}$, whereas if $y_2 + z = 2$, then the subgraph induced by the bipartition $(\{u\} \cup Y_2 \cup Z, Y_1)$ contains a copy of $K_{3,3}$. Suppose now that $y_2 + z = 1$. If $y_1 \leq 4$, then $y + z \leq 5$, and hence, the number of vertices in R is bounded by 16. Assume now that $y_1 \geq 5$. Let p be the vertex in $Y_2 \cup Z$; then p is connected to all vertices in Y_1 in H and hence in G. Moreover, every vertex in Y_1 is connected to either v or w (or both) in G. Since $y_1 \geq 5$, there must exist at least three vertices in Y_1 that are connected to either v or to w in G. Let these vertices be p_1, p_2, and p_3, and assume, without loss of generality, that these vertices are connected to v. Since p_1, p_2, and p_3, are also connected to p, there must exist a vertex in $\{p_1, p_2, p_3\}$, say p_2, that is interior to the region determined by v, p, and the other two vertices. But $p_2 \in Y_1 \subseteq N_2(v, w)$, and hence p_2 must be connected to the boundary of R (because p_2 must be connected to some vertex in $N_1(v, w)$), a contradiction. Thus, the number of vertices in $V(R)$ is bounded by 16.

Subcase 2.2. $y \leq 1$. If $z \leq 4$, then $y + z \leq 5$, and given that $|X| \leq 7$ by Observation 1, the total number of vertices in $V(R)$ is bounded by 16. Suppose now that $z \geq 5$. Observe first that $Y \neq \emptyset$; otherwise, $N_2(v, w) \cup N_3(v, w)$ consists of only white vertices, and $N_3(v, w)$, which contains at least five white vertices ($Z \subseteq N_3(v, w)$), could not be dominated by a single vertex in $N_2(v, w) \cup N_3(v, w)$. This would make Rule 2 applicable, a contradiction. Let p_1, p_2, p_3, p_4, and p_5 be vertices in Z. Since each of these vertices must be connected to either w or v, at least three vertices in $\{p_1, p_2, p_3, p_4, p_5\}$ are connected to either v or w. Suppose, without loss of generality, that $\{p_1, p_2, p_3\}$ are connected to v, and note that by Observation 2, every vertex in Y must be connected to all vertices in Z. If $|Y| \geq 2$, then $(\{v\} \cup Y, \{p_1, p_2, p_3\})$ would be a copy of $K_{3,3}$. Suppose now that $|Y| = 1$, and let q be the single vertex in Y. Since p_1, p_2, p_3 are white and belong to $N_3(v, w)$, these vertices can connect only to vertices in $\{v, q, w\}$. Again, by planarity, at least one vertex in $\{p_1, p_2, p_3\}$ is not connected to w and hence must be of degree 2. But then $|W(v, q)| \geq 3$, and $W(v, q)$ contains a degree-2 vertex. This contradicts Rule 8.

It follows that in all cases the number of vertices in $V(R)$ is bounded by 16. This completes the proof. □

REFERENCES

[1] F. Abu-Khzam and M. Langston, *A direct algorithm for the parameterized face cover problem*, in Proceedings of the 1st International Workshop on Parameterized and Exact Computation, Lecture Notes in Comput. Sci. 3162, Springer, Berlin, 2004, pp. 213–222.

[2] F. Abu-Khzam, M. Langston, and P. Shanbhag, *A High-Performance Toolkit for Fast Exact Algorithms*, invited talk to the Workshop on Fixed-Parameter Tractability, Ottawa, ON, Canada, 2003; available online at http://citeseer.ist.psu.edu/560785.html.

[3] J. Alber, N. Betzler, and R. Niedermeier, *Experiments on data reduction for optimal domination in networks*, Ann. Oper. Res., 146 (2006), pp. 105–117.

[4] J. Alber, H. Bodlaender, H. Fernau, T. Kloks, and R. Niedermeier, *Fixed parameter algorithms for dominating set and related problems on planar graphs*, Algorithmica, 33 (2002), pp. 461–493.

[5] J. Alber, M. Fellows, and R. Niedermeier, *Polynomial-time data reduction for dominating set*, J. ACM, 51 (2004), pp. 363–384.

[6] J. Alber, H. Fernau, and R. Niedermeier, *Parameterized complexity: Exponential speedup for planar graph problems*, J. Algorithms, 52 (2004), pp. 26–56.

[7] J. Alber, H. Fernau, and R. Niedermeier, *Graph separators: A parameterized view*, J. Comput. System Sci., 67 (2003), pp. 808–832.

[8] B. Baker, *Approximation algorithms for NP-complete problems on planar graphs*, J. ACM, 41 (1994), pp. 153–180.

[9] R. Balasubramanian, M. R. Fellows, and V. Raman, *An improved fixed parameter algorithm for vertex cover*, Inform. Process. Lett., 65 (1998), pp. 163–168.

[10] J. Cheetham, F. Dehne, A. Rau-Chaplin, U. Stege, and P. J. Taillon, *Solving large FPT problems on coarse grained parallel machines*, J. Comput. System Sci., 67 (2003), pp. 691–706.

[11] J. Chen and I. Kanj, *Constrained minimum vertex cover in bipartite graphs: Complexity and parameterized algorithms*, J. Comput. System Sci., 67 (2003), pp. 833–847.

[12] J. Chen and I. Kanj, *Improved exact algorithms for* Max-Sat, Discrete Appl. Math., 142 (2004), pp. 17–27.

[13] J. Chen, I. Kanj, and W. Jia, *Vertex cover: Further observations and further improvement*, J. Algorithms, 41 (2001), pp. 280–301.

[14] J. Chen, I. Kanj, and G. Xia, *Improved parameterized upper bounds for vertex cover*, in Proceedings of the 31st International Symposium on the Mathematical Foundations of Computer Science, Lecture Notes in Comput. Sci. 4162, Springer, Berlin, 2006, pp. 238–249.

[15] F. Dehne, M. Fellows, H. Fernau, E. Prieto, and F. Rosamond, *Nonblocker: Parameterized algorithmics for* Minimum Dominating Set, in Software Seminar SOFSEM, Lecture Notes in Comput. Sci. 3831, Springer, Berlin, 2006, pp. 237–245.

[16] R. Diestel, *Graph Theory*, Springer, Berlin, 1997.

[17] R. Downey and M. Fellows, *Parameterized Complexity*, Springer, New York, 1999.

[18] R. Downey, M. Fellows, and U. Stege, *Parameterized complexity: A framework for systematically confronting computational intractability*, in Contemporary Trends in Discrete Mathematics, DIMACS Ser. Discrete Math. Theoret. Comput. Sci. 49, R. Graham, J. Kratochvíl, J. Nešetřil, and F. Roberts, eds., AMS, Providence, RI, 1999, pp. 49–99.

[19] M. Fellows, *Parameterized complexity: The main ideas and connections to practical computing*, in Electronic Notes in Theoretical Computer Science, Vol. 61, 2002.

[20] M. Fellows, C. McCartin, F. Rosamond, and U. Stege, *Coordinatized kernels and catalytic reductions: An improved FPT algorithm for max leaf spanning tree and other problems*, in FST TCS 2000, Lecture Notes in Comput. Sci. 1974, Springer, Berlin, 2000, pp. 240–251.

[21] H. Fernau, *On parameterized enumeration*, in Computing and Combinatorics, Lecture Notes in Comput. Sci. 2387, Springer, Berlin, 2002, pp. 564–573.

[22] H. Fernau, *ROMAN Domination: A parameterized perspective*, in Software Seminar SOFSEM 2006, Lecture Notes in Comput. Sci. 3831, Springer, Berlin, 2006, pp. 262–271.

[23] H. Fernau and D. Juedes, *A geometric approach to parameterized algorithms for domination problems on planar graphs*, in Mathematical Foundations of Computer Science 2004, Lecture Notes in Comput. Sci. 3153, Springer, Berlin, 2004, pp. 488–499.

[24] H. Fernau and R. Niedermeier, *An efficient exact algorithm for constraint bipartite vertex cover*, J. Algorithms, 38 (2001), pp. 374–410.

[25] F. V. Fomin and D. M. Thilikos, *Dominating sets in planar graphs: Branch-width and exponential speed-up*, in Proceedings of the Fourteenth ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, Philadelphia, 2003, pp. 168–177.

[26] F. V. Fomin and D. Thilikos, *Fast parameterized algorithms for graphs on surfaces: Linear kernel and exponential speed-up*, in Automata, Languages and Programming (ICALP), Lecture Notes in Comput. Sci. 3142, Springer, Berlin, 2004, pp. 581–592.

[27] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

[28] J. Gimbel and C. Thomassen, *Coloring triangle-free graphs with fixed size*, Discrete Math., 219 (2000), pp. 275–277.

[29] H. Grötzsch, *Ein Dreifarbensatz für dreikreisfreie Netze auf der Kugel*, Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Ma.-Nat. Reihe, 8 (1958/1959), pp. 109–120.

[30] F. Henglein and H. G. Mairson, *The complexity of type inference for higher order typed lambda calculi*, J. Funct. Programming, 4 (1994), pp. 435–477.

[31] J. Hopcroft and R. Tarjan, *Efficient planarity testing*, J. ACM, 21 (1974), pp. 549–568.

[32] R. Hyafil and R. L. Rivest, *Constructing optimal binary trees is NP-complete*, Inform. Process. Lett., 5 (1976), pp. 15–17.

[33] I. Kanj and L. Perković, *Improved parameterized algorithms for planar dominating set*, in Mathematical Foundations of Computer Science 2002, Lecture Notes in Comput. Sci. 2420, Springer, Berlin, 2002, pp. 399–410.

[34] S. Khot and V. Raman, *Parameterized complexity of finding subgraphs with hereditary properties*, Theoret. Comput. Sci., 289 (2002), pp. 997–1008.

[35] O. Lichtenstein and A. Pneuli, *Checking that finite-state concurrent programs satisfy their linear specification*, in Proceedings of the 12th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL), 1985, pp. 97–107.

[36] R. Niedermeier and P. Rossmanith, *A general method to speed up fixed-parameter-tractable algorithms*, Inform. Process. Lett., 73 (2000), pp. 125–129.

[37] O. Ore, *Theory of Graphs*, Amer. Math. Soc. Colloq. Publ. XXXVIII, AMS, Providence, RI, 1962.

[38] E. Prieto and C. Sloper, *Either/or: Using vertex cover structure in designing FPT-algorithms—the case of k-Internal Spanning Tree*, in Algorithms and Data Structures (WADS), Lecture Notes in Comput. Sci. 2748, Springer, Berlin, 2003, pp. 474–483.

[39] C. Thomassen, *Grötzsch's* 3*-color theorem and its counterparts for the torus and the projective plane*, J. Combin. Theory Ser. B, 62 (1994), pp. 268–279.

[40] R. Uehara, *Probabilistic Algorithms and Complexity Classes*, Ph.D. thesis, Department of Computer Science and Information Mathematics, The University of Electro-Communications, Japan, 1998.

[41] K. Weihe, *Covering trains by stations or the power of data reduction*, in Proceedings of the Workshop on Algorithms and Experiments (ALEX '98), 1998, pp. 1–8; available online at http://rtm.science.unitn.it/alex98/proceedings.html.

SIAM J. Comput.
Vol. 37, No. 4, pp. 1107–1138

DISTRIBUTION-FREE PROPERTY-TESTING*

SHIRLEY HALEVY[†] AND EYAL KUSHILEVITZ[†]

Abstract. We consider the problem of distribution-free property-testing of functions. In this setting of property-testing, the distance between functions is measured with respect to a *fixed but unknown* distribution D on the domain. The testing algorithms are given oracle access to random sampling from the domain according to this distribution D. This notion of distribution-free testing was previously defined, but no distribution-free property-testing algorithm was known for any (non-trivial) property. We present the first such distribution-free property-testing algorithms for two of the central problems in this field. The testers are obtained by extending some known results (from "standard," uniform distribution, property-testing): (1) A distribution-free testing algorithm for low-degree multivariate polynomials with query complexity $O(d^2 + d \cdot \epsilon^{-1})$, where d is the total degree of the polynomial. The same approach that is taken for the distribution-free testing of low-degree polynomials is shown to apply also to several other problems; (2) a distribution-free monotonicity testing algorithm for functions $f : [n]^d \to A$ for low dimensions (e.g., when d is a constant) with query complexity similar to the one achieved in the uniform setting. On the negative side, we prove an exponential gap between the query complexity required for uniform and distribution-free monotonicity testing in the high-dimensional case.

Key words. property-testing, distribution-free property-testing, low-degree testing, monotonicity testing, distribution-free testing lower bounds

AMS subject classifications. 68W20, 68Q17

DOI. 10.1137/050645804

1. Introduction.

The classical notion of *decision problems* requires an algorithm to distinguish objects having some property $\mathcal{P}$ from those objects which do not have the property. *Property-testing* is a relaxation of decision problems, where algorithms are only required to distinguish objects having the property $\mathcal{P}$ from those which are at least "ϵ-far" from every such object. The notion of property-testing was introduced by Rubinfeld and Sudan [46] for the general case and was first studied in the context of combinatorial properties by Goldreich, Goldwasser, and Ron [25]. Since then it has attracted a considerable amount of attention. Property-testing algorithms were introduced for problems in graph theory (see, e.g., [2, 25, 28, 41]), monotonicity testing (see, e.g., [12, 16, 17, 21, 22, 24, 31]), and other properties (see, e.g., [1, 4, 5, 6, 7, 8, 10, 13, 14, 15, 18, 20, 26, 32, 34, 35, 39, 40, 42, 43, 45]).[1] The main goal of property-testing algorithms is to avoid "reading" the whole object, which requires complexity at least linear in the size of its representation, i.e., to make the decision by reading a small, possibly selected at random, fraction of the input (e.g., a fraction of size polynomial in $1/\epsilon$ and polylogarithmic in the size of the representation) and still having a good (say, at least 2/3) probability of success.

A crucial component in the definition of property-testing is that of the *distance* between two objects. For the purpose of this definition, it is common to think of objects as being *functions* over some domain $\mathcal{X}$. For example, a graph G may be

*Received by the editors November 22, 2005; accepted for publication (in revised form) March 9, 2007; published electronically November 21, 2007. This paper includes the results of the conference papers [30] and [33].

http://www.siam.org/journals/sicomp/37-4/64580.html

[†] Computer Science Department, Technion – Israel Institute of Technology, Haifa 32000, Israel (shirleyh@cs.technion.ac.il, eyalk@cs.technion.ac.il).

[1] The reader is referred to surveys by Ron [44], Goldreich [23], and Fischer [19] for a presentation of some of this work, including some connections between property-testing and other topics.

thought of as a function $f_G : V \times V \rightarrow \{0,1\}$ indicating for each possible edge e whether it exists in the graph. The distance between functions f and g is then measured by considering the set $\mathcal{X}_{f \neq g}$ of all points x where $f(x) \neq g(x)$ and comparing the size of this set $\mathcal{X}_{f \neq g}$ to that of $\mathcal{X}$;[2] equivalently, one may introduce a uniform distribution over $\mathcal{X}$ and measure the probability of picking $x \in \mathcal{X}_{f \neq g}$. Note that property-testing algorithms access the input function (object) via *membership queries* (i.e., the algorithm gives a value x from the domain and gets as an answer $f(x)$).[3]

It is natural to consider situations where not all the points of the domain are equivalently important. For example, when dealing with graph properties it is possible that removing or adding some edges may be more expensive than removing or adding others, depending on various factors of the problem (and not only on the number of edge removals and additions). We are interested in taking these differences between the edges into consideration when measuring the distance between a given graph and the property (which is not the case when dealing with standard uniform testing). In other words, we wish to put different weights (probabilities) on different elements of the domain. That is, our goal is to generalize the definition of distance between two functions, to deal with arbitrary probability distributions D over $\mathcal{X}$, by utilizing the probability measure of $\mathcal{X}_{f \neq g}$ according to D. Ideally, one would hope to get *distribution-free* property-testing algorithms (in short, distribution-free testers). A distribution-free tester for a given property $\mathcal{P}$ accesses the function using membership queries, as above, and by randomly sampling the *fixed but unknown* distribution D (this mimics similar definitions from *learning theory* and is implemented via an oracle access to D; see, e.g., [37]).[4] As before, the testing algorithm is required to accept the given function f with probability at least $\frac{2}{3}$ if f satisfies the property $\mathcal{P}$, and to reject it with probability at least $\frac{2}{3}$ if f is at least ϵ-far from $\mathcal{P}$ with respect to the distribution D.

Another natural question that arises when dealing with distribution-free testing, beyond the mere existence of such testers, is whether the testing (if possible) can always be accomplished using the same query complexity as in the uniform setting, or is there an inherent gap between the query complexity required for the different settings?

This definition of distance with respect to an arbitrary distribution D was already considered in the context of property-testing [25]. However, it is important to notice that the transformation of uniform testers into distribution-free ones is not straightforward. Many of the previously known testers use the fact that an object that is far from satisfying the property in question contains many small witnesses of this fact. As a result, many of the uniform testers use random sampling of the input object in order to find such a witness. On the other hand, when dealing with distribution-free property-testing, it is possible that an object contains very few such witnesses (hence, is very close to the property with respect to the uniform distribution), while at the same time is very far from the property with respect to some arbitrary distribution D.

[2] We usually compare objects of the same size, and hence a common definition for the distance between two objects is $|\mathcal{X}_{f \neq g}|$ (instead of $\frac{|\mathcal{X}_{f \neq g}|}{\mathcal{X}}$). However, defining the distance as described above is more suitable when dealing with general distribution measures.

[3] For example, in the case of a graph G, represented as a function f_G as described above, the algorithm gives a pair (u_1, u_2) and receives an answer of whether the edge (u_1, u_2) exists in the graph G or not.

[4] More precisely, distribution-free property-testing is the analogue of the PAC+MQ model of learning (that was studied by the learning theory community mainly via the EQ+MQ model); standard property-testing is the analogue of the uniform+MQ learning model.

Indeed, to the best of our knowledge, no distribution-free property-testing algorithm was known for any (nontrivial) property (besides testing algorithms that follow from the existence of proper learning algorithms in learning theory [25]). Moreover, discouraging impossibility results, due to [25], show that for many graph-theoretic properties (for which testers that work with respect to the uniform distribution are known) no such distribution-free algorithm exists. As a result, most previous work focused on testing algorithms for the uniform distribution; some of these algorithms can be generalized to deal with certain (quite limited) classes of distributions (e.g., product distributions [25]), and very few can be modified to be testers with respect to any *known* distribution (as was observed by [19] regarding the tester presented in [39]), but none is shown to be a distribution-free tester.

We start by reviewing some of the central problems, studied in the context of property-testing, which are relevant to our work.

Low-degree tests for polynomials. The first problem studied in the field of property testing is that of low-degree testing for multivariate polynomials over a finite field, where one wishes to test whether a given function can be represented by a multivariate polynomial of total degree d, or it is ϵ-far from any such polynomial. Later, the problem of low-degree testing played a central role in the development of probabilistic checkable proofs (PCPs), where the goal is to probabilistically verify the validity of a given proof. For the problem of low-degree testing, Rubinfeld and Sudan [46] presented a tester with query complexity of $O(d^2 + d \cdot \epsilon^{-1})$. This test was further analyzed in [11]. The reader is also referred to [13], where a linearity test (which tests whether a given function acts as a homomorphism between groups) is presented, and to [3, 9, 10, 26, 36] for other related work.

Monotonicity testing. Monotonicity has also been the subject of a significant amount of work in the property-testing literature (see, e.g., [12, 16, 17, 18, 21, 22, 24]). In monotonicity testing, the domain $\mathcal{X}$ is usually the d-dimensional cube $[n]^d$. A partial order is defined on this domain in the natural way (for $\vec{y}, \vec{z} \in [n]^d$, we say that $\vec{y} \leq \vec{z}$ if each coordinate of $\vec{y}$ is bounded by the corresponding coordinate of $\vec{z}$).[5] A function f over the domain $[n]^d$ is *monotone* if whenever $\vec{z} \geq \vec{y}$, then $f(\vec{z}) \geq f(\vec{y})$. Testing algorithms were developed to deal with both the low-dimensional and the high-dimensional cases (with respect to the uniform distribution over the domain). In what follows, we survey some of the known results on this problem which are most relevant to our work:

- *The low-dimensional case.* In this case, d is considered to be small compared to n (and, in fact, it is typically a constant); a successful algorithm for this case is typically one that is polynomial in $1/\epsilon$ and in $\log n$. The first paper to deal with this case is by Ergün et al. [17], who presented an $O(\frac{\log n}{\epsilon})$ algorithm for the line (i.e., the case $d = 1$) and showed that this query complexity cannot be achieved without using membership queries (that is, such query complexity cannot be achieved using only queries of the function at randomly drawn points of the domain; we have to allow the tester to make queries at points of its choice, computed based on the information it learned during the previous steps of its execution). This algorithm was generalized for any fixed d in [12]. For the case $d = 1$, there is a lower bound showing that testing monotonicity (for some constant ϵ) indeed requires $\Omega(\log n)$ queries [18].
- *The high-dimensional case.* In this case, d is considered as the main parameter (and n might be as low as 2); a successful algorithm is typically one that

[5] In the case $d = 1$ this yields a linear order.

is polynomial in $1/\epsilon$ and d. This case was first considered by Goldreich et al. [24], who showed an algorithm for testing monotonicity of functions from the Boolean ($n = 2$) d-dimensional hypercube to a Boolean range using $O(\frac{d}{\epsilon})$ queries. This result was generalized in [16] to arbitrary values of n, showing that $O(\frac{d\cdot\log^2 n}{\epsilon})$ queries suffice for testing monotonicity of general functions over $[n]^d$, which is the best known result so far.

- *Other related work.* Lower bounds for monotonicity testing of functions $f : \{0,1\}^d \to \{0,1\}$ (i.e., Boolean functions over $[n]^d$ for $n = 2$) were shown by Fisher et al. in [21]: an $\Omega(\sqrt{d})$ lower bound for nonadaptive, one-sided error algorithms, and an $\Omega(\log d)$ lower bound for nonadaptive two-sided error algorithms (these bounds imply the corresponding adaptive lower bounds of $\Omega(\log d)$ and $\Omega(\log\log d)$, respectively). These authors also considered graphs other than the hypercube, proving, for example, that testing monotonicity with a constant (depending only on $\frac{1}{\epsilon}$) number of queries is possible for certain classes of graphs.

1.1. Our contributions. Our main positive contributions are distribution-free testers for the two properties mentioned above: low-degree multivariate polynomials and low-dimensional monotone functions. Both testers use query complexity similar to that known for the uniform setting. We also show that the low-degree test is a special case of a general scheme for a wider class of properties including properties such as dictatorship and junta functions [20, 43]. These algorithms are the first known distribution-free testers for nontrivial properties. By presenting these algorithms, we answer a natural question that has been raised explicitly by Fischer [19, subsection 9.3] and is implicit in [25]. We emphasize that our algorithms work for any distribution D without having any information about D.

On the negative side, we show that, although in the uniform setting testing of Boolean functions defined over the Boolean hypercube can be done using query complexity that is polynomial in $\frac{1}{\epsilon}$ and in the dimension d, in the distribution-free setting such testing requires a number of queries that is exponential in d. We elaborate below.

Distribution-free low-degree testing for polynomials (and more). We consider a generalization of the notion of self-correctors for single functions (see [13]) to classes of functions (this was previously introduced in [46]). We observe that if the property in question has such a self-corrector, then the existence of a uniform tester implies the existence of a distribution-free one. This approach yields distribution-free testers for a variety of properties, among which is the property of low-degree multivariate polynomials (see section 3).

In addition, for the problem of testing low-degree multivariate polynomials, we show how to generalize the tester presented in [46] to a distribution-free tester with the same (up to a multiplicative factor of 2) query complexity ($O(d^2+d\cdot\epsilon^{-1})$) without using self-correctors. The algorithm and its analysis are presented in subsection 3.1.

Distribution-free monotonicity testing in the low-dimensional case. We present a distribution-free monotonicity tester in the low-dimensional hypercube case with query complexity of $O(\frac{\log^d n\cdot 2^d}{\epsilon})$. This is done by first considering the one-dimensional case (the "line"). In this case, we prove that an algorithm of [17] can be slightly modified to handle the distribution-free case with the same query complexity of $O(\frac{\log n}{\epsilon})$. Our proof is presented for the sake of completeness and it shares similar ideas with the original proof [17] for the uniform case (although it is possible to modify the original analysis for the distribution-free case). We then show how to appropriately generalize this algorithm to deal with higher (yet low compared to n) dimensions (a similar generalization was used in [12] for the uniform distribution case). The tester for the one-dimensional case and its generalization for higher dimensions appear in section 4.

It is typical for known property-testing algorithms to be quite simple, and it is in the analysis of why these algorithms work that the property $\mathcal{P}$ in question requires understanding; indeed, Goldreich and Trevisan [29] proved that in certain settings this is an inherent phenomenon: they essentially showed (with respect to the uniform distribution) that any graph-theoretic property that can be tested can also be tested (with a small penalty in the complexity) by a "generic" algorithm that samples a random subgraph and decides whether it has some property. Our work is no different in this aspect: our algorithms are similar to previously known algorithms, and our main contribution is their analysis, in particular, the analysis for the distribution-free case. Moreover, it is somewhat surprising that our distribution-free testers do not require dramatically different techniques than those used in the construction and analysis of previous algorithms (that work for the uniform distribution case). We remark, however, that although all the distribution-free testers presented in this work can be viewed as variations of testers for the uniform distribution, in each of the problems the modification of the uniform distribution test is different.[6]

Lower bound for distribution-free monotonicity testing in the high-dimensional case. In section 5, we show that while $O(\frac{d}{\epsilon})$ queries suffice for testing monotonicity of Boolean functions in the uniform case [24], in the distribution-free case it requires a number of queries that is exponential in the dimension d (that is, there exists a constant c such that $\Omega(2^{cd})$ queries are required for the testing). Hence, such testing is infeasible in the high-dimensional case. Our lower bound can be trivially extended to deal with monotonicity testing of Boolean functions that are defined over the domain $[n]^d$ for general values of n. By this, we show another gap between the query complexity of uniform and distribution-free testers for a natural testing problem.

Organization. In section 2, we formally define the notions used in this work. In section 3, we deal with properties that have both a uniform tester and a property self-corrector; then, in subsection 3.1, we present an alternative distribution-free tester for low-degree multivariate polynomials that does not use the notion of property self-correctors. Section 4 describes the distribution-free monotonicity tester for the low-dimensional case, and section 5 describes the lower bound on the query complexity of distribution-free monotonicity testing in the high-dimensional case. Finally, section 6 presents some open questions with respect to property-testing in general and distribution-free testing in particular.

2. Definitions.

We start by formally defining the notion of being ϵ-far from a property $\mathcal{P}$, with respect to a given distribution D defined over the domain $\mathcal{X}$, and of distribution-free testing. Assume that the range of the functions in question is $\mathcal{A}$.

DEFINITION 2.1. *Denote by $\mathcal{F}_{\mathcal{A}}^{\mathcal{X}}$ the set of all functions $f : \mathcal{X} \to A$. A* property *$\mathcal{P}$ is a subset $\mathcal{P} \subseteq \mathcal{F}_{\mathcal{A}}^{\mathcal{X}}$. We say that f* satisfies *$\mathcal{P}$ if $f \in \mathcal{P}$.*

DEFINITION 2.2. *Let $\mathcal{X}$ be a domain and D be a distribution measure defined over $\mathcal{X}$. The D-*distance *between functions $f, g \in \mathcal{F}_{\mathcal{A}}^{\mathcal{X}}$ is defined by $dist_D(f, g) \overset{def}{=} \Pr_{x\sim D}\{f(x) \neq g(x)\}$.*

*The D-*distance *of a function f from a property $\mathcal{P}$ is $dist_D(f, \mathcal{P}) \overset{def}{=} \inf_{g\in\mathcal{P}} dist_D(f, g)$.*

[6] Indeed, in light of [25], there can be no generic transformation of uniform distribution testers into distribution-free ones.

We say that f is (ϵ, D)-far from a property $\mathcal{P}$ *if $dist_D(f, \mathcal{P}) \geq \epsilon$. Similarly, we say that f is* (ϵ, D)-close to a property $\mathcal{P}$ *if $dist_D(f, \mathcal{P}) \leq \epsilon$.*

When the distribution in question is the uniform distribution over $\mathcal{X}$, we either use U instead of D or (if clear from the context) we omit any reference to the distribution; e.g., the phrase "f is ϵ-far from $\mathcal{P}$" refers to distance according to the uniform distribution.

Next, we define the notion of a distribution-free tester for a given property $\mathcal{P}$.

DEFINITION 2.3. *A* distribution-free two-sided error tester *for a property $\mathcal{P}$ is a probabilistic oracle machine M, which is given a distance parameter $\epsilon > 0$, an oracle access to an arbitrary function $f : \mathcal{X} \to \mathcal{A}$, and an oracle access to sampling of a fixed but unknown distribution D over $\mathcal{X}$, and satisfies the following two conditions:*

1. *If f satisfies $\mathcal{P}$, then* $\Pr\{M^{f,D} = \textit{ Accept}\} \geq \frac{2}{3}$.
2. *If f is (ϵ, D)-far from $\mathcal{P}$, then* $\Pr\{M^{f,D} = \textit{ Accept}\} \leq \frac{1}{3}$.

All our testers, like many previously known testers, have one-sided error; i.e., they always accept any function satisfying the property $\mathcal{P}$ in question.

DEFINITION 2.4. *A* distribution-free one-sided error tester *for a property $\mathcal{P}$ is a probabilistic oracle machine M, which is given a distance parameter $\epsilon > 0$, an oracle access to an arbitrary function $f : \mathcal{X} \to \mathcal{A}$, and an oracle access to sampling of a fixed but unknown distribution D over $\mathcal{X}$, and satisfies the following two conditions:*

1. *If f satisfies $\mathcal{P}$, then* $\Pr\{M^{f,D} = \textit{ Accept}\} = 1$.
2. *If f is (ϵ, D)-far from $\mathcal{P}$, then* $\Pr\{M^{f,D} = \textit{ Accept}\} \leq \frac{1}{3}$.

The definition of a uniform distribution (two-sided or one-sided error) tester for a property $\mathcal{P}$ can be derived from the previous definitions by omitting the sampling oracle (since the tester can sample in the uniform distribution by itself) and by measuring distance with respect to the uniform distribution.

Notice that, because the distribution D in question is arbitrary, there is a difference between satisfying $\mathcal{P}$ and having distance 0 from $\mathcal{P}$. That is, it is possible that there are two distinct functions f and g such that $dist_D(f, g) = 0$. Specifically, it is possible that $f \in \mathcal{P}$ and $g \notin \mathcal{P}$. Since the notion of testing is meant to be a relaxation of the notion of decision problems, it is required that the algorithm accept (with probability at least $\frac{2}{3}$) functions that satisfy $\mathcal{P}$, but it may reject functions that have distance 0 from $\mathcal{P}$ (but do not satisfy $\mathcal{P}$). In addition, note that membership queries allow the algorithm to query the value of the input function also in points with probability 0 (which is also the case with membership queries in learning theory).[7] This definition of distribution-free testing was previously introduced in [25].

3. Distribution-free testing of properties with self-corrector. In this section, we focus on a specific class of properties for which the existence of a uniform tester implies the existence of a distribution-free one. These properties are characterized by the fact that we are able to efficiently "correct" every object that is close to the property. That is, given a function f that is close to a property $\mathcal{P}$, we are able to efficiently compute (with high probability), in every point x of the domain, the correct value of the function $g \in \mathcal{P}$ that is close to the input function f. We refer to this ability as "property self-correction." First, we formally define the notion of a property self-correction (it has already been defined implicitly and used in [38, 46]), and then introduce a general scheme for obtaining distribution-free testers for a variety of properties that satisfy these conditions.

[7] It is not known whether membership queries are essential in general for testing even in the uniform case (see [44]); this is known only for specific problems such as monotonicity testing (see [17]).

The notion of a property self-corrector is a generalization of the notion of self-correctors for functions that was introduced by Blum, Luby, and Rubinfeld in [13]. A self-corrector for a specific function f is a randomized algorithm that, given oracle access to a function g which is ϵ-close to f, is able to compute the value $f(x)$ in every point x of the domain. This definition can be generalized to classes of functions, specifically demanding that all the functions in the class are self-correctable using the same algorithm.

DEFINITION 3.1. *An ϵ'* self-corrector *for a property $\mathcal{P}$ is a probabilistic oracle machine M, which is given an oracle access to an arbitrary function $f : \mathcal{X} \rightarrow \mathcal{A}$ and satisfies the following conditions:*

- *If $f \in \mathcal{P}$, then $\Pr\{M^f(x) = f(x)\} = 1$ for every $x \in \mathcal{X}$.*
- *If there exists a function $g \in \mathcal{P}$ such that $dist_U(f, g) \leq \epsilon'$ (i.e., f is ϵ'-close to $\mathcal{P}$), then $\Pr\{M^f(x) = g(x)\} \geq \frac{2}{3}$ for every $x \in \mathcal{X}$.*

Note that the definition of a property self-corrector refers to distance measured only with respect to the uniform distribution; however, we still use these correctors for the construction of distribution-free testers. Observe that a necessary condition for the existence of an ϵ-self-corrector for a property $\mathcal{P}$ is that for every function f such that $dist_U(f, \mathcal{P}) \leq \epsilon$ (i.e., f is ϵ-close to $\mathcal{P}$ with respect to the uniform distribution), there exists a *unique* function $g \in \mathcal{P}$ that is ϵ-close to $\mathcal{P}$. Notice that the property of monotonicity does not fulfill this requirement.[8] Hence, the distribution-free monotonicity tester that is presented in the next section requires a different approach.

Next, in Figure 3.1 we describe our general distribution-free testing scheme. Let $\mathcal{P}$ be a property, let $T_{\mathcal{P}}$ be a uniform distribution tester for $\mathcal{P}$ with query complexity Q_T that has one-sided error, and let $C_{\mathcal{P}}$ be an ϵ' property self-corrector for $\mathcal{P}$ with query complexity Q_C. Let $\epsilon \leq \epsilon'$, and $f : \mathcal{X} \rightarrow A$.

```
Tester$_D(\epsilon)$
Run $T^f_{\mathcal{P}}(\epsilon)$. If $T^f_{\mathcal{P}}(\epsilon)$ = FAIL, then return FAIL.
Repeat $\frac{2}{\epsilon}$ times:
    Choose $x \in_D \mathcal{X}$.
    Repeat twice: Run $C^f_{\mathcal{P}}(x)$; If $f(x) \neq C^f_{\mathcal{P}}(x)$, then return FAIL.
return PASS
```

FIG. 3.1. *Generalized distribution-free testing scheme.*

THEOREM 3.2. *Algorithm Tester$_D(\epsilon)$ is a one-sided error distribution-free tester for $\mathcal{P}$ with query complexity $Q_T(\epsilon) + \frac{4}{\epsilon} \cdot Q_C$.*

Proof. It is obvious that the query complexity of the algorithm Tester$_D(\epsilon)$ is indeed as required. Hence, we have only to prove the correctness of the algorithm. To do so, we prove the following two facts:

- If $f \in \mathcal{P}$, then f is accepted by the algorithm with probability 1.
- If f is (ϵ, D)-far from $\mathcal{P}$, then f is rejected by Tester$_D(\epsilon)$ with probability at least $\frac{2}{3}$.

If f is indeed in $\mathcal{P}$, then it passes the uniform test with probability 1, and the value returned by the self-corrector is always identical to the value of f. Hence, f is accepted by the algorithm. Assume, from now on, that f is (ϵ, D)-far from $\mathcal{P}$. We distinguish between two possibilities as follows:

[8] Consider, for example, the following function $f : [n] \rightarrow \{0, 1\}$: Set $f(i) = 0$ if i is even, and set $f(i) = 1$ if i is odd. f is $\frac{1}{2}$-far from monotone, and it is $\frac{1}{2}$-close to $O(n)$ monotone functions.

- If f is (ϵ, U)-far from $\mathcal{P}$, then the probability that it passes the uniform test is at most $\frac{1}{3}$.
- If f is (ϵ, U)-close to $\mathcal{P}$, then there exists a function $g \in \mathcal{P}$ such that $dist(f, g) \leq \epsilon$. However, since $dist_D(f, \mathcal{P}) \geq \epsilon$, we deduce that $dist_D(f, g) \geq \epsilon$ (in other words, $\Pr_{x \sim D}\{f(x) \neq g(x)\} \geq \epsilon$). If f was accepted by the algorithm, then one of the two following events happened: Either we failed to sample a point in which f and g differ, or we succeeded in sampling such a point x, but both executions of the self-corrector failed to compute the value $g(x)$. The probability of the first event is at most $(1-\epsilon)^{\frac{2}{\epsilon}} \leq \frac{1}{e}^2 \leq \frac{1}{6}$ and, by the definition of a property self-corrector, the probability of the second event is at most $\frac{1}{3}^2 = \frac{1}{9}$. Therefore, the total probability that f is accepted by the algorithm is at most $\frac{1}{3}$.

Hence, in both cases the probability that f is accepted by the algorithm is at most $\frac{1}{3}$. □

Remark 3.3. We used the assumption that there exists a uniform distribution tester for the property $\mathcal{P}$ that has one-sided error. However, the same transformation can be applied also when the uniform distribution tester has two-sided error, except that the resulting distribution-free tester also has two-sided error.

This scheme implies the existence of distribution-free property testers for a few properties. Among these properties are low-degree multivariate polynomials (which will be the focus of the next subsection) and junta and dictatorship functions. A function $f : \{0,1\} \to \{0,1\}$ is said to be a k-junta if there exists a subset of $\{x_1, \ldots, x_n\}$ of size k that determines the value of f (i.e., f is independent of the other variables). A special case of juntas are dictatorship functions, where a single variable determines the value of the function. These properties (and other related properties) have uniform distribution testers, as was shown in [46, 20, 43]. In addition, they are subsets of the class of low-degree polynomials (for example, k-juntas are a special case of degree k multivariate polynomials) and thus are self-correctable [38, 46]. Therefore, the scheme described in this section gives distribution-free testers for these properties.

Remark 3.4. Given two properties $\mathcal{P}$ and $\mathcal{P}'$ such that $\mathcal{P}' \subseteq \mathcal{P}$, the fact that $\mathcal{P}$ is testable in the uniform distribution does not imply that $\mathcal{P}'$ is thus testable (to see this observe that every property is a subset of the class of all functions that is clearly testable). However, the fact that $\mathcal{P}$ is self-correctable implies that $\mathcal{P}'$ is self-correctable (using the same correction algorithm).

3.1. An alternative distribution-free test for low-degree multivariate polynomials. In this subsection, we present a distribution-free test for the property of low-degree multivariate polynomials that does not use property self-correctors. This test is obtained by modifying the known uniform tester presented by [46]. Hence, it might be used, along with other distribution-free testing algorithms, to deduce a general transformation of uniform testers into distribution-free ones (in case such a distribution-free tester exists).

We first define the problem formally. Let F be a finite field. In the problem of low-degree testing, with respect to the uniform distribution, the tester is given access to a function $f : F^m \to F$, a distance parameter ϵ, and a degree d, and has to decide whether f is a multivariate polynomial of total degree d, or is at least ϵ-far (with respect to the uniform distribution) from any degree d multivariate polynomial (i.e., one has to change the values of at least $\epsilon \times |F|^m$ points in order to transform f into a degree d multivariate polynomial; this implies that for every degree d multivariate polynomial g, the probability that a *uniformly* drawn point x has a value $g(x)$ different than $f(x)$ is at least ϵ). Rubinfeld and Sudan [46] presented a tester for this problem with query complexity $O(d^2 + d \cdot \epsilon^{-1})$. We show how to modify this tester to a distribution-free one with the same query complexity (up to a factor of 2).

Basic tool. Fix some value for d and assume from now on that $|F| > 10d$. To describe the testers (both the one for the uniform distribution and our distribution-free one), we use the following terminology from [46]:

- A *line* in F^m is a set of $10d + 1$ points of the form $\{x, x + h, \ldots, x + 10dh\}$ for some $x, h \in F^m$. The line defined by x and h is denoted $\ell_{x,h}$.
- We say that a line $\ell_{x,h}$ *is an f-polynomial* if there exists a univariate polynomial $P_{x,h}(i)$ of degree d such that $f(x + ih) = P_{x,h}(i)$ for every $0 \leq i \leq 10d$.

Notice that if f is a multivariate polynomial of total degree at most d, then for every x and h, the line $\ell_{x,h}$ is an f-polynomial.[9] Given the values of f on a line $\ell_{x,h}$, testing whether this line is an f-polynomial can be done as follows:

- Find, using interpolation, a univariate polynomial $P(i)$ of degree d, consistent with the values of f at the $d+1$ points $x, x+h, \ldots, x+dh$ (i.e., $P(i) = f(x+hi)$ for every $0 \leq i \leq d$).
- Check, for every $(d + 1) \leq i \leq 10d$, that $f(x + ih) = P(i)$. If so, accept; otherwise reject.

We show how this basic test is used to build a uniform and a distribution-free low-degree test.

Low-degree test for the uniform distribution. The low-degree test for the uniform distribution works by randomly sampling $O(d+\epsilon^{-1})$ lines (i.e., by uniformly choosing $x, h \in F^m$) and checking that each of these lines is an f-polynomial. The correctness of this algorithm follows immediately from the following theorem (see [46, Theorem 9]).

THEOREM 3.5 (see [46, Theorem 9]). *There exists a constant c_U such that for $0 \leq \delta \leq \frac{1}{c_U \cdot d}$, if f is a function from F^m to F, such that all but at most a δ fraction of the lines $\{\ell_{x,h} | x, h \in F^m\}$ are f-polynomials, then there exists a polynomial $g : F^m \to F$ of total degree at most d such that $dist_U(f, g) \leq (1 + o(1))\delta$ (provided that $|F| > 10d$).*

Distribution-free low-degree testing algorithm. Denote the class of multivariate polynomials of total degree d by $\mathcal{P}^d_{deg}$. We show that the tester described above can be modified into a distribution-free tester for low-degree multivariate polynomials. That is, we present an algorithm with query complexity $O(d^2 + d \cdot \epsilon^{-1})$ that, given a distance parameter ϵ, a degree parameter d, and access to random sampling of F^m according to D and to membership queries of a function $f : F^m \to F$, distinguishes, with probability at least $\frac{2}{3}$, between the case that f is in $\mathcal{P}^d_{deg}$ and the case that f is (ϵ, D)-far from $\mathcal{P}^d_{deg}$.

The natural generalization of the uniform distribution tester above for the distribution-free case would be to replace the sampling of the tested lines by sampling according to the distribution D; i.e., sample the $O(d + \epsilon^{-1})$ lines by choosing $x \sim D$ and $h \sim U$ and check that these lines are f-polynomials. However, we could not show that this modification actually works. Instead, the algorithm we present consists of two stages: In the first stage, we simply run the uniform distribution test as is and check that the function f is ϵ-close to $\mathcal{P}^d_{deg}$ with respect to the uniform distribution;

[9] To see this, assume $f(x) = \Sigma_j a_j \Pi_{l=1}^{d_j} x_{k_l^j}$, where a_j is the coefficient of the jth term in f, d_j is the degree ($d_j \leq d$), and k_l^j is the index of the lth variable in that term. In this case, for every fixed $x = (x_1, \ldots, x_m)$ and $h = (h_1, \ldots, h_m)$, the value $f(x + ih) = \Sigma_j a_j \Pi_{l=1}^{d_j}(x_{k_l^j} + ih_{k_l^j})$, which, of course, is a degree d univariate polynomial in i.

the second stage is the generalization suggested above. This combined strategy is presented in Figure 3.2 and analyzed below.

```
Poly($\epsilon, d$)

Set $k \triangleq \max\{\epsilon^{-1}, c_U \cdot d\}$.
Repeat $5k$ times:
    • Choose $x, h \in_U F^m$. If the line $\ell_{x,h}$ is not an $f$-polynomial, return FAIL.
    • Choose $x \in_D F^m$, $h \in_U F^m$. If the line $\ell_{x,h}$ is not an $f$-polynomial, return FAIL.
return PASS
```

FIG. 3.2. *Distribution-free low-degree tester.*

THEOREM 3.6. *Algorithm Poly*(ϵ, d) *is a distribution-free one-sided error tester for* $\mathcal{P}^d_{deg}$*; its query complexity is* $O(d^2 + d \cdot \epsilon^{-1})$.

The correctness of the algorithm relies on the following lemma.

LEMMA 3.7. *Let* c_U *be the constant from Theorem* 3.5. *For every* $0 \leq \delta \leq \frac{1}{c_U \cdot d}$, *if* f *is a function from* F^m *to* F *such that*

- $\Pr_{x,h \sim U}\{\ell_{x,h}$ *is not an* f*-polynomial*$\} \leq \delta$, *and*
- $\Pr_{x \sim D, h \sim U}\{\ell_{x,h}$ *is not an* f*-polynomial*$\} \leq \delta$,

then there exists a polynomial $g : F^m \to F$ *of total degree at most* d *such that* $dist_D(f, g) \leq \frac{\delta}{1-40\delta} = (1 + o(1))\delta$ *(provided that* $|F| > 10d$*).*

Proof. We prove that if f satisfies the two conditions for $0 \leq \delta \leq \frac{1}{c_U \cdot d}$, then it is indeed $(\frac{\delta}{1-40\delta}, D)$-close to $\mathcal{P}^d_{deg}$. For the purpose of the analysis, we construct a function g based on f in the same way as in [46, proof of Theorem 9]:

- For every $x, h \in F^m$, denote by $P_{x,h}(i)$ the univariate polynomial that satisfies $P_{x,h}(i) = f(x + ih)$ for at least $6d$ of the values $i \in \{0, \ldots, 10d\}$. If no such polynomial exists, define $P_{x,h}$ to be an "error." Notice that there can be at most one such polynomial, since every two polynomials that satisfy the demand have to agree on at least $2d$ points, implying that they are identical.
- For every $x \in F^m$, define $g(x) = \text{plurality}_h\{P_{x,h}(0)\}$, where the plurality is taken over all values of h for which $P_{x,h}$ is not an error ($g(x)$ is well defined for every x since, by [46, Lemma 12], $\Pr_{h_1,h_2 \sim U}\{P_{x,h_1}(0) = P_{x,h_2}(0)\} \geq 1 - 20\delta$.

It was shown that the function g is indeed a degree d multivariate polynomial (see [46, proof of Theorem 9] and notice that f satisfies the condition of that theorem). Moreover, for every $x \in F^m$, $i \in \{0, \ldots, 10d\}$, $\Pr_{h \sim U}[g(x + ih) = P_{x,h}(i)] \geq 1 - 40\delta$ (see [46, Corollary 13]).

To complete the proof, it remains to show that $dist_D(f, g) \leq \frac{\delta}{1-40\delta}$. Let B be the set of x's that satisfy $f(x) \neq P_{x,h}(0)$ for at least a $1 - 40\delta$ fraction of the h's in F^m (including the values of h for which $P_{x,h}$ was defined to be an error). Notice that if $x \notin B$, then $f(x) = P_{x,h}(0)$ for more than a 40δ fraction of the h's in F^m (this is due to the fact that $f(x) \neq P_{x,h}(0)$ for less than a $1 - 40\delta$ fraction of the h's, and hence it has to be equal to $P_{x,h}(0)$ for more than a 40δ fraction of the h's). Therefore, for every $x \notin B$, we can claim the following:

1. $g(x) = P_{x,h}(0)$ for at least a $1 - 40\delta$ fraction of the h's (this holds for every x, and thus for every $x \notin B$). Denote this fraction by g_x.
2. $f(x) = P_{x,h}(0)$ for more than a 40δ fraction of the h's in F^m. Denote this fraction by f_x.

Therefore, for at least one value of h both $f(x) = P_{x,h}(0)$ and $g(x) = P_{x,h}(0)$ (otherwise, $f_x + g_x > 1$, since $g_x \geq 1 - 40\delta$ and $f_x > 40\delta$). Hence, for every $x \notin B$, we get $f(x) = g(x)$. Let us bound the probability of picking a point in B according to the distribution D as follows:

$$\delta \geq \Pr_{x \sim D, h \sim U}\{\ell_{x,h} \text{ is not an } f\text{-polynomial}\} \geq (1 - 40\delta) \Pr_{x \sim D}\{x \in B\},$$

where the first inequality is due to the second condition of the lemma, and the second inequality follows the previous discussion, implying that

$$\Pr_{x \sim D}\{x \in B\} \leq \frac{\delta}{1 - 40\delta}. \qquad \square$$

Proof of Theorem 3.6. To prove that the algorithm is indeed a distribution-free tester for $\mathcal{P}^d_{deg}$, we prove the following two facts:

1. If f is in $\mathcal{P}^d_{deg}$, then the algorithm accepts f with probability 1.
2. If f is (ϵ, D)-far from $\mathcal{P}^d_{deg}$, then the algorithm $Poly(\epsilon, d)$ rejects f with probability at least $\frac{2}{3}$.

As explained before, if f is indeed a multivariate polynomial of total degree d, then every line is an f-polynomial. Hence, it follows that such an f is accepted by the tester with probability 1. Assume from now on that f is (ϵ, D)-far from $\mathcal{P}^d_{deg}$. Notice that, by the definition of $k = \max\{\epsilon^{-1}, c_U \cdot d\}$, for $\epsilon' = \frac{1}{k}$, f is (ϵ', D)-far from $\mathcal{P}^d_{deg}$. By Lemma 3.7, either $\Pr_{x,h \sim U}\{\ell_{x,h}$ is not an f-polynomial$\} > \frac{\epsilon'}{2+40\epsilon'}$, or $\Pr_{x \sim D, h \sim U}\{\ell_{x,h}$ is not an f-polynomial$\} > \frac{\epsilon'}{2+40\epsilon'}$ (otherwise, it follows that there exists a degree d polynomial g such that $dist_D(f,g) \leq \frac{\epsilon'}{(2+40\epsilon')\cdot(1-40\frac{\epsilon'}{2+40\epsilon'})} = \frac{\epsilon'}{2} < \epsilon'$, contradicting the fact that the D-distance of f from any such polynomial is at least ϵ'). Assume that the first event occurs. Therefore, the probability that a randomly chosen line $\ell_{x,h}$ is an f-polynomial is at most $(1 - \frac{\epsilon'}{2+40\epsilon'})$. Hence, the probability that the algorithm accepts f is at most $(1 - \frac{\epsilon'}{2+40\epsilon'})^{5k} = (1 - \frac{1}{2k+40})^{5k} \leq \frac{1}{e}^2 \leq \frac{1}{3}$ (the first inequality follows from the fact that c_U, the constant of Theorem 3.5, is at least 100, implying that $k \geq 100$). Similarly, if the second event occurs, the probability that a randomly chosen line $\ell_{x,h}$, where $x \sim D$ and $h \sim U$, is an f-polynomial is at most $(1 - \frac{\epsilon'}{2+40\epsilon'})$. Hence, as before, the probability that the algorithm accepts f is at most $(1 - \frac{\epsilon'}{2+40\epsilon'})^{5k} \leq \frac{1}{3}$. $\square$

4. Distribution-free monotonicity testing on the *d*-dimensional cube.

In this section, we present monotonicity testers for the d-dimensional hypercube with respect to an arbitrary distribution D. As before, we assume D to be fixed but unknown and, other than the ability to sample according to D, we assume no knowledge of D. For simplicity, we begin our discussion with the case $d = 1$, and show that, given access to random sampling according to D and to membership queries, there is a distribution-free tester for monotonicity over $[n]$, whose query complexity is $O(\frac{\log n}{\epsilon})$ (the algorithm for the case $d = 1$ is presented without proof). In subsection 4.2, we generalize this algorithm to a distribution-free tester for monotonicity over the d-dimensional hypercube, whose query complexity is $O(\frac{\log^d n \cdot 2^d}{\epsilon})$ (this algorithm for the case $d = 1$ is the basic algorithm shown in subsection 4.1).

We begin with a few notations and definitions. Denote by $[n]$ the set $\{1, \ldots, n\}$ and by $[n]^d$ the set of d-tuples over $[n]$. For every two points $\vec{i}$ and $\vec{j}$ in $[n]^d$ we say that $\vec{i} \leq \vec{j}$ if for every $1 \leq k \leq d$, $i_k \leq j_k$. Let $(A, <_A)$ be some linear order.

DEFINITION 4.1. *We say that a function* $f : [n]^d \to A$ *is* monotone *if for every* $\vec{i}$ *and* $\vec{j}$ *if* $\vec{i} \leq \vec{j}$, *then* $f(\vec{i}) \leq_A f(\vec{j})$.

DEFINITION 4.2. *Let* $f : [n]^d \to A$ *be a function. A pair* $(\vec{i}, \vec{j})$ *is said to be an* f-violation *if* $\vec{i} < \vec{j}$ *and* $f(\vec{i}) >_A f(\vec{j})$.

Let D be any distribution on $[n]^d$, and let S be a subset of $[n]^d$. Define $\Pr_D\{\vec{i}\} \overset{\text{def}}{=} \Pr_{X \sim D}\{X = \vec{i}\}$ and $\Pr_D\{S\} \overset{\text{def}}{=} \sum_{\vec{i} \in S} \Pr_D\{\vec{i}\}$.

4.1. Testing monotonicity for the line ($d = 1$). We consider the case $d = 1$. Our algorithm is a variant of the algorithm presented in [17] for testing monotonicity (with respect to the uniform distribution). The analysis for the special case $d = 1$ shares some of the ideas presented in the proof of [17] and is omitted; it can be obtained as a special case of the general proof that appears in the next subsection. The algorithm works in phases; in each phase a center point is selected according to the distribution D (in the original algorithm, the center point is selected uniformly), and the algorithm looks for a violation of the monotonicity with this center point. The search for a violation is done by randomly sampling in growing neighborhoods of the center point. In other words, in the case $d = 1$, the only change made in the original algorithm in order to adjust it to be distribution-free is that the choice of center points is made according to D. However, the search for violations remains unchanged. It is important to observe that, when dealing with an arbitrary distribution, there is no connection between the distance of the function from monotone and the number of pairs that form a violation of monotonicity (unlike the case of testing with respect to the uniform distribution, when a small number of violations implies closeness to monotonicity). Hence, the correctness of the algorithm for the uniform distribution does not imply its correctness for the general case.

As stated above, we present a tester for monotonicity in the one-dimensional case with query complexity $O(\frac{\log n}{\epsilon})$. The algorithm appears in Figure 4.1 and is a special case of the algorithm for general d that is proved in the next subsection. A separate correctness proof for the case $d = 1$ can be found in [30].

```
Algorithm-monotone-1-dim$_D(f, \epsilon)$:
repeat $\frac{2}{\epsilon}$ times
   choose $i \in_D [n]$
   for $k \leftarrow 0 \ldots \lceil \log i \rceil$ do
      repeat 8 times
         choose $a \in_U [2^k]$
         if $f(i - a) >_A f(i)$ then return FAIL
   for $k \leftarrow 0 \ldots \lceil \log(n - i) \rceil$ do
      repeat 8 times
         choose $a \in_U [2^k]$
         if $f(i) >_A f(i + a)$ then return FAIL
return PASS
```

FIG. 4.1. *Distribution-free monotonicity tester for* $d = 1$.

Remark 4.3. In the journal version of [17], an additional testing algorithm for monotonicity on the line, called Sort-Check-II, is presented. This algorithm can also be transformed into a distribution-free monotonicity tester over the line. However, we do not know if it can be generalized to higher dimensions.

4.2. Distribution-free monotonicity testing for general *d*. In the previous section, we saw how to test monotonicity over the one-dimensional hypercube (the line) when the distance is measured with respect to an arbitrary distribution D.

Next, we show how to generalize this algorithm to the d-dimensional case and prove its correctness. We begin by adjusting the definitions and notations of the previous section to the d-dimensional case. Notice that, unlike in the one-dimensional case, in the d-dimensional case $[n]^d$ is a partial order (not every two points are comparable). Denote by $\vec{i}$ the point $i_1, \dots, i_d \in [n]^d$. For every two points $\vec{i}$ and $\vec{a}$, we denote by $\vec{i} + \vec{a}$ (respectively, $\vec{i} - \vec{a}$) the point $i_1 + a_1, \dots, i_d + a_d$ (respectively, $i_1 - a_1, \dots, i_d - a_d$), where every coordinate smaller than 1 is set to 1 and every coordinate larger than n is set to n.

We will show that, given access to random sampling according to D and to membership queries, there is a distribution-free tester for monotonicity, whose query complexity is $O(\frac{\log^d n \cdot 2^d}{\epsilon})$. The algorithm is presented in Figure 4.2.

```
Algorithm monotone-d-dim$_D(d, f, \epsilon)$:
repeat $\frac{2}{\epsilon}$ times
  choose $\vec{i} \in_D [n]^d$
   for $k_1 \leftarrow 0 \cdots \lceil \log i_1 \rceil$ do
        ⋮
     for $k_d \leftarrow 0 \dots \lceil \log i_d \rceil$ do
       repeat $4 \cdot 2^d$ times
         choose $\vec{a} \in_U [2^{k_1}] \times \cdots \times [2^{k_d}]$
         if $f(\vec{i} - \vec{a}) >_A f(\vec{i})$ then return FAIL
  for $k_1 \leftarrow 0 \dots \lceil \log(n - i_1) \rceil$ do
        ⋮
     for $k_d \leftarrow 0 \dots \lceil \log(n - i_d) \rceil$ do
       repeat $4 \cdot 2^d$ times
         choose $\vec{a} \in_U [2^{k_1}] \times \cdots \times [2^{k_d}]$
         if $f(\vec{i}) >_A f(\vec{i} + \vec{a})$ then return FAIL
return PASS
```

FIG. 4.2. *Distribution-free monotonicity tester for general d.*

THEOREM 4.4. *Algorithm monotone-d-*$\mathrm{dim}_D(d, f, \epsilon)$ *is a one-sided error distribution-free monotonicity tester over* $[n]^d$ *with query complexity* $O(\frac{\log^d n \cdot 2^d}{\epsilon})$.

To prove the above theorem, we need the following lemmas.

LEMMA 4.5. *Let* $f : [n]^d \to A$ *be a function, and let* $S \subseteq [n]^d$ *be a set. If for every* f*-violation* $(\vec{i}, \vec{j})$ *either* $\vec{i} \in S$ *or* $\vec{j} \in S$, *then there exists a monotone function* f' *that differs from* f *only on points in* S.

Proof. We will show that, by modifying the value of f only on points in S, we can obtain a monotone function f'. Define $\bar{S} = [n]^d \setminus S$. The construction of f' is done through the following iterative process. In each step, modify the value of one of the points $\vec{i} \in S$ that has not yet been modified and so that the values of all the points below it (i.e., all $\vec{k} < \vec{i}$) have already been set. The process ends after redefining the values of all the points in S.

The value of point $\vec{i} \in S$ is set in the following manner. We distinguish between two cases: $\vec{i} = (1, \dots, 1)$ and $\vec{i} \in [n]^d \setminus (1, \dots, 1)$. If $\vec{i} = (1, \dots, 1)$, then we set $f'(\vec{i}) = f(\vec{i})$ if $S = [n]^d$, and $f'(\vec{i}) = \min_{\vec{j}:\vec{i}<\vec{j},\vec{j}\in\bar{S}}\{f(\vec{j})\}$ otherwise. If $\vec{i} \in S \setminus (1, \dots, 1)$, we set $f'(i) = \max_{\vec{j}:\vec{j}<\vec{i}}\{f(\vec{j})\}$ (note that the maximum is taken over *all* points smaller than $\vec{i}$ and not only those in S) . The values of points in $\bar{S}$ are unchanged (i.e., for all these points $f'(\vec{i}) = f(\vec{i})$).

It is left to prove that f' is indeed monotone, or equivalently, that there are no f'-violations. Consider a pair of points $\vec{i}, \vec{j}$ such that $\vec{i} < \vec{j}$; we will prove that $f'(\vec{i}) \leq_A f'(\vec{j})$. There are three possible cases.

Case 1. $\vec{i}, \vec{j} \in \bar{S}$. Since f' equals f for all points not in S, and there were no f-violations with both endpoints in $\bar{S}$, then $f'(\vec{i}) = f(\vec{i}) \leq_A f(\vec{j}) = f'(\vec{j})$.

Case 2. $\vec{j} \in S$ and $\vec{i} \in [n]^d$. By the construction of f', the value of $f'(\vec{j})$ was set after the value at point $\vec{i}$ had already been set (since $\vec{i} < \vec{j}$), and therefore $f'(\vec{i}) \leq_A f'(\vec{j})$.

Case 3. $\vec{i} \in S$ and $\vec{j} \in \bar{S}$. Assume towards a contradiction that $(\vec{i}, \vec{j})$ is an f'-violation; moreover, let $(\vec{i}, \vec{j})$ be the minimal such pair (in lexicographic order). Since we already know that there are no f'-violations other than the ones referred to in this case, this is the minimal f'-violation. By the definition of f', clearly if $\vec{i} = (1, \ldots, 1)$, then $f'(\vec{i}) = \min_{\vec{j}: \vec{i} < \vec{j}, \vec{j} \in \bar{S}} \{f(\vec{j})\}$ and therefore $f'(\vec{i}) \leq_A f(\vec{j}) = f'(\vec{j})$. Hence $\vec{i} \neq (1, \ldots, 1)$. By the construction of f', the value of $f'(\vec{i})$ is the maximal value of f' at points $\vec{k}$ such that $\vec{k} < \vec{i}$. Thus $f'(\vec{i}) = f'(\vec{k})$ for some $\vec{k} < \vec{i}$, implying that $(\vec{k}, \vec{j})$ is also an f'-violation, contradicting the minimality of $(\vec{i}, \vec{j})$. □

A similar argument was used in [16]. An immediate conclusion of the above lemma is the following.

Lemma 4.6. *Let $f : [n]^d \to A$ be a function (ϵ, D)-far from monotone. Given $S \subseteq [n]^d$, if for every f-violation $(\vec{i}, \vec{j})$ either $\vec{i} \in S$ or $\vec{j} \in S$, then $\Pr_D\{S\} \geq \epsilon$.*

Definition 4.7. *For an f-violation $(\vec{i}, \vec{j})$, we say that $\vec{i}$ is* active *in this violation if*

$$|\{\vec{k} \ : \ \vec{i} < \vec{k} < \vec{j} \ , \ f(\vec{i}) >_A f(\vec{k})\}| \ \geq \ \frac{|\{\vec{k} : \vec{i} < \vec{k} < \vec{j}\}|}{2}.$$

That is, $\vec{i}$ is active in an f-violation $(\vec{i}, \vec{j})$ if, for at least half of the points $\vec{i} < \vec{k} < \vec{j}$, $(\vec{i}, \vec{k})$ is also an f-violation (i.e., $f(\vec{i}) >_A f(\vec{k})$). Similarly, $\vec{j}$ is active *in this violation if*

$$|\{\vec{k} \ : \ \vec{i} < \vec{k} < \vec{j} \ , \ f(\vec{k}) >_A f(\vec{j})\}| \ \geq \ \frac{|\{\vec{k} : \vec{i} < \vec{k} < \vec{j}\}|}{2}.$$

Lemma 4.8. *For every f-violation $(\vec{i}, \vec{j})$, at least one of $\vec{i}$ and $\vec{j}$ is active in $(\vec{i}, \vec{j})$.*

Proof. Let $(\vec{i}, \vec{j})$ be an f-violation. Namely, $\vec{i} < \vec{j}$ and $f(\vec{i}) >_A f(\vec{j})$. For every $\vec{i} < \vec{k} < \vec{j}$, if $f(\vec{i}) \leq_A f(\vec{k})$ (i.e., $(\vec{i}, \vec{k})$ is not an f-violation), then $f(\vec{k}) >_A f(\vec{j})$ (since $(A, <_A)$ is a linear order), and hence $(\vec{k}, \vec{j})$ is an f-violation. Similarly, if $(\vec{k}, \vec{j})$ is not an f-violation, then $(\vec{i}, \vec{k})$ is. Therefore, each of the points between $\vec{i}$ and $\vec{j}$ forms an f-violation with either $\vec{i}$ or $\vec{j}$. □

Define the active set of f (denoted A_f) as the set of all points that are active in some f-violation. Applying Lemma 4.6 to the set A_f, if f is (ϵ, D)-far from monotone, then $\Pr_D\{A_f\} \geq \epsilon$. We now prove Theorem 4.4.

Proof of Theorem 4.4. Clearly, the query complexity of the algorithm is $O(\frac{\log^d n \cdot 2^d}{\epsilon})$. It is left to show that this algorithm is indeed a distribution-free one-sided error monotonicity tester. To prove this, the following two facts have to be shown: (a) If f is monotone, it is accepted by the algorithm with probability 1; and (b) if f is (ϵ, D)-far from monotone, it is rejected by the algorithm with probability at least $\frac{2}{3}$.

As for (a), it follows immediately from the definition of the algorithm. Therefore, assume from now on that f is (ϵ, D)-far from monotone; we prove that f is rejected with probability at least $\frac{2}{3}$. Our algorithm may fail to detect that f is not monotone if either one of the following two events occurs:

1. None of the $\frac{2}{\epsilon}$ points $\vec{i}$ sampled by the algorithm according to D are in A_f.

2. The algorithm picked at least one point $\vec{i} \in A_f$, but failed to detect that $\vec{i}$ belongs to some f-violation.

The probability of the first event is at most $(1-\epsilon)^{\frac{2}{\epsilon}} \leq \frac{1}{e^2} \leq \frac{1}{6}$. To estimate the probability of the second event, let $\vec{i}$ be a point in A_f and assume, without loss of generality, that there exists $\vec{j}$ such that $(\vec{i},\vec{j})$ is an f-violation and $\vec{i}$ is active in this violation. Denote by k_l, $1 \leq l \leq d$, the minimal value such that $i_l + 2^{k_l} \geq j_l$. By the minimality of k_l, it follows that $j_l - i_l > \frac{1}{2} \cdot 2^{k_l}$ and therefore $(j_1 - i_1) \cdot \ldots \cdot (j_d - i_d) > \frac{1}{2^d} \cdot 2^{k_1} \cdot \ldots \cdot 2^{k_d}$. Since $\vec{i}$ is active in the f-violation $(\vec{i},\vec{j})$, at least half of the values between $\vec{i}$ and $\vec{j}$ form an f-violation with $\vec{i}$. Therefore at least $\frac{1}{2^{d+1}}$ of the values $\vec{a}$ in $[2^{k_1}] \times \cdots \times [2^{k_d}]$ will reveal a violation of monotonicity. Since for every possible value for $k_1, \ldots, k_d$ we sample $4 \cdot 2^d$ points in $[2^{k_1}] \times \cdots \times [2^{k_d}]$, the probability that no f-violation is detected is at most $(1 - \frac{1}{2^{d+1}})^{4 \cdot 2^d} = (1 - \frac{1}{2^{d+1}})^{2 \cdot 2^{d+1}} \leq \frac{1}{e^2} \leq \frac{1}{6}$. Therefore, the total probability of the algorithm failing to detect a violation, and hence to accept f, is at most $\frac{1}{3}$. □

Remark 4.9. A similar distribution-free monotonicity tester can be shown for Boolean functions of the form $f : B \to \{0,1\}$, where $(B, <_B)$ is a partial order that can be described as an "almost balanced" tree (i.e., a tree of depth $O(\log n)$ when $n = |B|$); monotonicity testing of functions defined over trees was also studied in [21], where a constant time algorithm was presented for testing Boolean functions defined over trees. The testing is again performed in stages. In each stage, as before, the tester selects a center point x according to the distribution D in question and starts looking for a violation of monotonicity with that center point. However, in this case, we distinguish between two possibilities: If $f(x) = 0$, then the tester uniformly samples $O(\log n)$ points in the path from x to the root (the tester can actually read the whole path using $O(\log n)$ queries). Otherwise, if $f(x) = 1$, then the tester uniformly samples $O(1)$ points in growing neighborhoods of x, where the neighborhoods are defined to be trees rooted at x, and at each time we increase the depth of the tree by 1 (thereby at most doubling the size of the tree).

The correctness proof of the tester uses the same ideas, with the following adjustment. For every violation of monotonicity (i,j), where $i < j$ (necessarily $f(i) = 1$ and $f(j) = 0$), we say that i is heavy in this violation if at least a $1 - \frac{1}{O(\log n)}$ fraction of the values in the path from i to j are 0, and we say that j is heavy if at least a $\frac{1}{O(\log n)}$ fraction of the vertices in the path from i to j are 1. Using arguments similar to those used in the above proof, it is easy to show that this is indeed a one-sided error monotonicity tester.

5. A lower bound for distribution-free monotonicity testing in the high-dimensional case.

In this section we show that distribution-free testing of Boolean functions, defined over the Boolean hypercube $\{0,1\}^d$, requires a number of queries that is exponential in the dimension. In subsection 5.1, we give an overview of the lower bound proof and present the families of functions used in this proof. Then, for simplicity, we first prove, in subsection 5.2, the lower bound for one-sided error testing, and later in subsection 5.3 we extend our lower bound for the two-sided error case.

Hereafter, we identify any point $x \in \{0,1\}^d$ with the corresponding set $x \subseteq [d]$. This allows us to apply set theory operations, such as union and intersection, to points.

In addition, for any two points $p, p' \in \{0,1\}^{\frac{d}{2}}$, denote by $p||p'$ the point $x \in \{0,1\}^d$, that is, the concatenation of p and p' (i.e., it is identical to p in its first $\frac{d}{2}$ coordinates and to p' in its last $\frac{d}{2}$ coordinates). Given a point $x \in \{0,1\}^d$, we say that $p \in \{0,1\}^{\frac{d}{2}}$ is the prefix of x if $x = p||p'$ for some p'. For every two points $x, y \in \{0,1\}^d$, denote by $H(x,y)$ the Hamming distance between x and y. For every point x, define $U(x) \stackrel{\text{def}}{=} \{y : y \geq x, |y| \leq \frac{d}{2}\}$; similarly define $L(x) \stackrel{\text{def}}{=} \{y : y \leq x, |y| > \frac{d}{2}\}$. Denote by $B^d_{\lambda d}$ the set of points in $\{0,1\}^d$ of weight λd, that is, $B^d_{\lambda d} = \{x \in \{0,1\}^d : |x| = \lambda d\}$. We always assume that d is divisible enough and $\lambda \cdot d$ is an integer.

5.1. Overview of the construction. To prove the lower bound on the query complexity, we show that any two-sided error distribution-free monotonicity tester with subexponential query complexity is unable to distinguish between monotone functions and functions that are $\frac{1}{2}$-far from monotone. To this aim, we consider two families $\mathcal{F}_1$ and $\mathcal{F}_2$ of pairs (f, D_f), where f is a Boolean function and D_f is a probability distribution corresponding to f, both defined over $\{0,1\}^d$, such that the following three requirements are satisfied:

1. Every function in $\mathcal{F}_1$ is monotone. Hence, every tester has to accept every pair $(f, D_f) \in \mathcal{F}_1$ with probability at least $\frac{2}{3}$.
2. Every function f in $\mathcal{F}_2$ is $\frac{1}{2}$-far from monotone with respect to D_f. Therefore, every tester has to reject every pair $(f, D_f) \in \mathcal{F}_2$ with probability at least $\frac{2}{3}$ (regardless of the choice of the distance parameter ϵ).
3. There exists a constant c such that no algorithm A that makes fewer than 2^{cd} membership queries and samples the distribution less than 2^{cd} times satisfies that
   - A accepts every pair (f, D_f) in $\mathcal{F}_1$ with probability at least $\frac{2}{3}$.
   - A rejects every pair (f, D_f) in $\mathcal{F}_2$ with probability at least $\frac{2}{3}$.

We conclude that there exists no two-sided error distribution-free tester for monotonicity of Boolean functions defined over $\{0,1\}^d$ with subexponential query complexity. (A similar approach was previously used for lower bound proofs in property-testing with respect to the uniform distribution [28]. However, since the previous proofs considered the uniform distribution setting, the two families in those proofs consist of functions and not pairs of functions and distributions.)

In the rest of this subsection, we describe the two families of functions (and the corresponding distributions), $\mathcal{F}_1$ and $\mathcal{F}_2$, defined over $\{0,1\}^d$. Let $\alpha < \frac{1}{32}$ be a parameter used for the construction of both families and define $m \stackrel{\text{def}}{=} 2^{\alpha d}$. In the construction, we use a set $\mathcal{M} \subset B^{d/2}_{(1/4-\alpha)\cdot d}$ of size $2m$ such that every two points in $\mathcal{M}$ are "far apart." This property of $\mathcal{M}$ will be used in the proof that the third requirement is satisfied. We first define the exact requirements from such a set $\mathcal{M}$ and claim its existence.

Lemma 5.1. *There exists a set $\mathcal{M} \subset B^{d/2}_{(1/4-\alpha)\cdot d}$ that satisfies the following conditions:*

- $|\mathcal{M}| = 2m$.
- $H(p, p') > 2\alpha d$ *for every two distinct points* $p, p' \in \mathcal{M}$.

Proof. First, note that the second condition of the lemma is equivalent to asking that $|p \cap p'| < (\frac{1}{4} - 2\alpha)d$ for every two distinct points $p, p' \in \mathcal{M}$. This holds since $H(p,p') = 2((\frac{1}{4} - \alpha)d - |p \cap p'|)$; hence, $|p \cap p'| < (\frac{1}{4} - 2\alpha)d$ iff $H(p,p') > 2\alpha d$.

We show that a randomly drawn set of $2m$ points in $B^{d/2}_{(1/4-\alpha)\cdot d}$ satisfies the small intersection condition with nonzero probability. Let P be the probability that two randomly drawn points $x, y \in B^{d/2}_{(1/4-\alpha)\cdot d}$ have intersection of size at least $(\frac{1}{4} - 2\alpha)d$.

By the union bound, the probability that a set of $2m$ randomly drawn points does not satisfy the small intersection condition is at most $4m^2P$. We prove the existence of such a set $\mathcal{M}$ by showing that $4m^2P < 1$. It is also possible to see P as the probability for a given point $x \in B^{d/2}_{(1/4-\alpha)\cdot d}$ that a randomly drawn point $y \in B^{d/2}_{(1/4-\alpha)\cdot d}$ has an intersection of size at least $(\frac{1}{4} - 2\alpha)d$ with x, and hence

$$\begin{aligned} P &= \frac{\sum_{i=(\frac{1}{4}-2\alpha)d}^{(\frac{1}{4}-\alpha)d} \binom{(\frac{1}{4}-\alpha)d}{i}\binom{(\frac{1}{4}+\alpha)d}{(\frac{1}{4}-\alpha)d-i}}{\binom{\frac{d}{2}}{(\frac{1}{4}-\alpha)d}} < (\alpha d+1)\frac{\binom{(\frac{1}{4}-\alpha)d}{(\frac{1}{4}-2\alpha)d}\binom{(\frac{1}{4}+\alpha)d}{\alpha d}}{\binom{\frac{d}{2}}{(\frac{1}{4}-\alpha)d}} \\ &= \frac{(\alpha d+1)((\frac{1}{4}-\alpha)d)!^2((\frac{1}{4}+\alpha)d)!^2}{\frac{d}{2}!((\frac{1}{4}-2\alpha)d)!(\frac{1}{4}d)!(\alpha d)!^2}. \end{aligned}$$

A simple calculation, based on Stirling formula, shows that $P < \frac{1}{4m^2}$ for sufficiently large d, implying the claimed result. To see this, we write

$$\begin{aligned} P &< \frac{(\alpha d+1)\cdot((\frac{1}{4}-\alpha)d)!^2((\frac{1}{4}+\alpha)d)!^2}{\frac{d}{2}!((\frac{1}{4}-2\alpha)d)!(\frac{1}{4}d)!(\alpha d)!^2} \\ &\approx \frac{(\alpha d+1)(2\pi)\cdot((\frac{1}{4}-\alpha)d\cdot((\frac{1}{4}-\alpha)d)^{2(\frac{1}{4}-\alpha)d}\cdot e^{-2(\frac{1}{4}-\alpha)d})}{(2\pi)^{\frac{5}{2}}\cdot\left(\sqrt{\frac{d}{2}}\cdot\frac{d}{2}^{\frac{d}{2}}\cdot e^{-\frac{d}{2}}\right)\cdot\left(\sqrt{(\frac{1}{4}-2\alpha)d}\cdot\left(\left(\frac{1}{4}-2\alpha\right)d\right)^{(\frac{1}{4}-2\alpha)d}\cdot e^{-(\frac{1}{4}-2\alpha)d}\right)} \\ &\quad\cdot\frac{((\frac{1}{4}+\alpha)d\cdot((\frac{1}{4}+\alpha)d)^{2(\frac{1}{4}+\alpha)d}\cdot e^{-2(\frac{1}{4}+\alpha)d})}{\left(\sqrt{\frac{d}{4}}\cdot(\frac{d}{4})^{\frac{d}{4}}\cdot e^{-\frac{d}{4}}\right)\cdot(\alpha d\cdot(\alpha d)^{2\alpha d}\cdot e^{-2\alpha d})} \\ &= \frac{(\alpha d+1)(\frac{1}{16}-\alpha^2)\cdot((\frac{1}{4}-\alpha)d)^{2(\frac{1}{4}-\alpha)d}\cdot((\frac{1}{4}+\alpha)d)^{2(\frac{1}{4}+\alpha)d}\cdot e^{-d}}{\sqrt{\frac{\pi}{4}\cdot((\frac{1}{4}-2\alpha)d)}\frac{d}{2}^{\frac{d}{2}}\cdot\frac{d}{4}^{\frac{d}{4}}\cdot\alpha\cdot\left(\left(\frac{1}{4}-2\alpha\right)d\right)^{(\frac{1}{4}-2\alpha)d}\cdot e^{-d}\cdot(2\alpha d)^{2\alpha d}} \\ &= \frac{(\alpha d+1)(\frac{1}{16}-\alpha^2)\cdot(\frac{1}{4}-\alpha)^{2(\frac{1}{4}-\alpha)d}\cdot(\frac{1}{4}+\alpha)d^{2(\frac{1}{4}+\alpha)d}}{\sqrt{\frac{\pi}{4}\cdot(\frac{1}{4}-2\alpha)}\cdot\alpha\cdot\frac{d}{2}^{\frac{d}{2}}\cdot\frac{d}{4}^{\frac{d}{4}}(\frac{1}{4}-2\alpha)^{(\frac{1}{4}-2\alpha)d}\cdot\alpha^{2\alpha d}}. \end{aligned}$$

For $\alpha < \frac{1}{32}$, we have $P < \frac{1}{4m^2}$. □

Using a set $\mathcal{M}$, as in Lemma 5.1, we define the two families. For each pair (f, D_f), we first define the function $f : \{0,1\}^d \to \{0,1\}$ and then, based on the definition of f, we define the corresponding distribution D_f over $\{0,1\}^d$.

5.1.1. The family $\mathcal{F}_1$.

Each function f is defined by first choosing (in all possible ways) two sets X_1 and X_2, each of size m, such that $X_1 \subset B^d_{1/2-\alpha}$ and $X_2 \subset B^d_{1/2+\alpha}$. The choice of the two sets is done as follows:

- Choose a set of m points $\mathcal{M}' \subset \mathcal{M}$.
- For each point p in $\mathcal{M}'$, randomly choose a point $y \in B^{d/2}_{d/4}$ and add the point $x = p||y$ to X_1 (as needed, $|x| = (1/2 - \alpha)d$).
- For each point p in $\mathcal{M} \setminus \mathcal{M}'$, randomly choose a point $y \in B^{d/2}_{(1/4+2\alpha)\cdot d}$ and add the point $x = p||y$ to the set X_2 (as needed, $|x| = (1/2 + \alpha)d$).

The function f is now defined in the following manner:

- For every point $x_1 \in X_1$ and every point y such that $y \geq x_1$, set $f(y) = 1$.
- For every point $x_2 \in X_2$ and every point y such that $y \leq x_2$, set $f(y) = 0$.

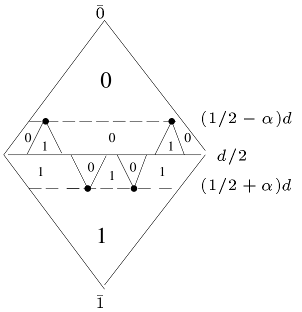


FIG. 5.1. *An example of a function in $\mathcal{F}_1$ for $m = 2$.*

- For every point y, such that $f(y)$ was not defined above, if $|y| \le \frac{d}{2}$, then $f(y) = 0$; otherwise $f(y) = 1$.

See Figure 5.1 for an example of such a function for $m = 2$.

The distribution D_f corresponding to the function f is the uniform distribution over the $2m$ points in $X_1 \cup X_2$. That is, $D_f(x) = \frac{1}{2m}$ for every $x \in X_1 \cup X_2$ and $D_f(x) = 0$ for any other point in $\{0,1\}^d$.

The fact that $\mathcal{F}_1$ is not empty follows from the existence of such a set $\mathcal{M}$. To see that the function f is well defined, and that every function in $\mathcal{F}_1$ is monotone (and hence that the family $\mathcal{F}_1$ satisfies the first requirement of the construction), we observe the following simple lemma.

LEMMA 5.2. $\{y : y \ge x_1\} \cap \{y : y \le x_2\} = \phi$*, for every* $x_1 \in X_1$ *and* $x_2 \in X_2$*.*

Proof. If there exists a point $z \in \{y : y \ge x_1\} \cap \{y : y \le x_2\}$, then $x_1 < x_2$. By the construction of f, there exist two distinct points $p_1, p_2 \in \mathcal{M}$ such that $x_1 = p_1||y_1$ and $x_2 = p_2||y_2$ for some y_1 and y_2. Thus, $p_1 \le p_2$, contradicting the fact that $p_1 \ne p_2$ and $|p_1| = |p_2|$. □

5.1.2. The family $\mathcal{F}_2$. Each function f in $\mathcal{F}_2$ is also defined by first choosing (in all possible ways) two sets X_1 and X_2, each of size m such that $X_1 \subset B^d_{(1/2-\alpha)\cdot d}$ and $X_2 \subset B^d_{(1/2+\alpha)\cdot d}$. The choice of the two sets is done as follows:

- Choose a set of m points $\mathcal{M}' \subset \mathcal{M}$.
- For each point p in $\mathcal{M}'$, randomly choose a pair of points (y_1, y_2) such that
  - $y_1 \in B^{d/2}_{d/4}$.
  - $y_2 \in B^{d/2}_{(1/4+2\alpha)\cdot d}$.
  - $y_1 < y_2$.
- Add the point $x_1 = p||y_1$ to X_1 and the point $x_2 = p||y_2$ to X_2. The two points x_1 and x_2 will be referred to as a *couple* in f.

The function f is now defined as follows:

- For every point $x_1 \in X_1$ and every point y such that $y \in U(x_1)$, set $f(y) = 1$.
- For every point $x_2 \in X_2$ and every point y such that $y \in L(x_2)$, set $f(y) = 0$.
- For every point y such that $f(y)$ was not defined above, if $|y| \le \frac{d}{2}$ then $f(y) = 0$; otherwise $f(y) = 1$.

See Figure 5.2 for an example of such a function for $m = 2$.

The distribution D_f is again the uniform distribution over $X_1 \cup X_2$, as in the definition of $\mathcal{F}_1$.

The fact that $\mathcal{F}_2$ is not empty follows immediately from the existence of the set $\mathcal{M}$. We now argue that the family $\mathcal{F}_2$ satisfies the second requirement of the construction;

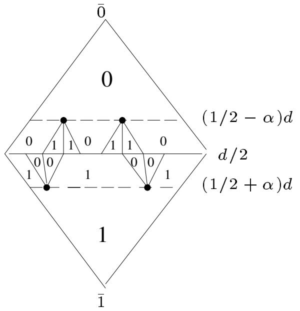


FIG. 5.2. *An example of a function in $\mathcal{F}_2$ for $m = 2$.*

that is, every function f in $\mathcal{F}_2$ is $\frac{1}{2}$-far from monotone with respect to D_f.

LEMMA 5.3. *f is $\frac{1}{2}$-far from monotone, with respect to D_f, for every $(f, D_f) \in \mathcal{F}_2$.*

Proof. To transform f into a monotone function, we have to alter the value of f, either in x_1 or in x_2, for every couple (x_1, x_2) in f. By doing this, we increase the distance by $\frac{1}{2m}$. Since there are m disjoint couples, the total distance is at least $\frac{1}{2}$. (In fact, since the distance of a Boolean function from monotone can be at most $\frac{1}{2}$, the distance is exactly $\frac{1}{2}$.) □

Remark 5.4. The fact that the points in $\mathcal{M}$ are far apart was not used so far, but it will be used later in the proof that the third requirement is satisfied. The first two requirements from $\mathcal{F}_1$ and $\mathcal{F}_2$ are satisfied for every choice of a set $\mathcal{M}$ satisfying the other conditions of Lemma 5.1.

5.2. A lower bound for one-sided error testing.

In this section, to give some insight into the proof and a better understanding of the two families, we first prove our lower bound for the simpler case of one-sided error testers. The arguments used in this section will be later generalized to the case of two-sided error testers.

Specifically, we prove that there exists no one-sided error distribution-free monotonicity tester with subexponential query complexity that accepts every pair (f, D_f) in $\mathcal{F}_1$, with probability 1, and rejects every pair (f, D_f) in $\mathcal{F}_2$, with high probability. To this aim, we show that for every tester A, there exists a pair $(f, D_f) \in \mathcal{F}_2$, such that with high probability, the execution of A on (f, D_f) is also consistent with some monotone function from $\mathcal{F}_1$. Since A has to accept every monotone function, then A has to accept f with high probability.

The above claim is simple if the tester is not allowed to use membership queries but only to sample the distribution D_f. In this case, to distinguish between the two families, $\mathcal{F}_1$ and $\mathcal{F}_2$, the tester has to detect a couple in the function from $\mathcal{F}_2$. Given i samples from the distribution, the probability that the $(i+1)$st sample will be a couple of one of the already known i points is at most $\frac{i}{2m}$. Therefore, the probability of distinguishing between the two families using n samples is at most $\sum_{i=1}^{n} \frac{i}{m} = O(\frac{n^2}{m})$. Since m is exponential in d, distinguishing between the two families requires an exponential number of queries. For proving the lower bound, we assume that all testers in question have full knowledge of the two families $\mathcal{F}_1$ and $\mathcal{F}_2$ and specifically the set of prefixes $\mathcal{M}$. Hence, the difference between one tester and another is in the tester's choice of membership queries, and dealing with the tester's membership queries is where the difficulty of this proof lies.

To state our claim formally, we introduce some notation. Let A be a tester with query and sample complexity $n = n(d, \frac{1}{\epsilon})$. Assume that the tester A works in two stages. In the first stage, the tester performs n samplings of the distribution D_f, and in the second stage, A performs n membership queries.[10] Such a tester can also be viewed as a mapping from a sequence $\{(p_j, v_j)\}_{j=1}^{i}$ of labelled points either to "sample the distribution" (if $i \leq n$) or to "query q_{i+1}" (if $n < i < 2n$) and to "accept" or "reject" if $i = 2n$. We refer to the sequence of labelled points obtained by A during any execution as a *knowledge sequence*. Given a pair (f, D_f), denote by S_f^i the knowledge sequence learnt by A during the first i steps of its run on (f, D_f) (for some possible execution of A on (f, D_f)). That is, S_f^i is a random variable that depends both on the outcome of the samplings from D_f and on A's internal random coins. Hereafter, we use the notation S_f to denote S_f^{2n}; that is, S_f is a knowledge sequence learnt by A during a full execution of A on (f, D_f). We say that a function f is consistent with a knowledge sequence S_f^i (in short, S_f^i consistent) if $f(p_j) = v_j$ for every $1 \leq j \leq i$. We say that a weight $(\frac{1}{2} - \alpha)d$ point x is consistent with S_f^i (in short, an S_f^i consistent point) if there exists an S_f^i consistent function f' such that x is in the set $\mathcal{X}'_1$ used for the construction of f'. Similarly, we say that a weight $(\frac{1}{2} + \alpha)d$ point x is consistent with S_f^i if there exists an S_f^i consistent function f' such that x is in the set $\mathcal{X}'_2$ used for the construction of f'.

As stated above, we will show that for some constant c, for every tester A with query and sample complexity $n < 2^{cd}$, there exists a pair $(f, D_f) \in \mathcal{F}_2$ such that with high probability, over the possible executions of A and the sampling of the domain according to D_f, the sequence S_f is consistent with some monotone function $f' \in \mathcal{F}_1$. Hence, with high probability S_f causes A to accept, contradicting the requirement that A must reject (f, D_f) with probability at least $\frac{2}{3}$.

For this, we show that for every tester A and for every possible choice of random coins for A, the probability that, for a uniformly drawn pair (f, D_f) from $\mathcal{F}_2$, the sequence S_f is consistent with some monotone function from $\mathcal{F}_1$ is very high (where the probability is taken over the choice of (f, D_f) and the sampling of D_f). Hence, by Yao's principle, for every tester A, there exists a pair $(f, D_f) \in \mathcal{F}_2$ such that for most choices of random coins for A, the probability that S_f is consistent with a monotone function is very high.

LEMMA 5.5. *For every tester A with query and sample complexity n, there exists a pair $(f, D_f) \in \mathcal{F}_2$ such that*

$$\Pr\{S_f \textit{ is consistent with a monotone function in } \mathcal{F}_1\} \geq 1 - \frac{3n^2}{m},$$

where the probability is taken over the choice of random coins for A and the samplings of the distribution D_f.

Based on the above lemma, we can state our lower bound.

THEOREM 5.6. *Let $c = \alpha/3$. For every one-sided error monotonicity tester A that makes fewer than 2^{cd} membership queries and samples the distribution less than 2^{cd} times, there exists a pair $(f, D_f) \in \mathcal{F}_2$ such that $\Pr\{A \text{ accepts } (f, D_f)\} > \frac{1}{3}$.*

Proof. Set $n = 2 \cdot 2^{cd}$. By Lemma 5.5, there exists a pair $(f, D_f) \in \mathcal{F}_2$ such that $\Pr\{S_f$ is consistent with a monotone function$\} \geq 1 - \frac{3n^2}{m}$. By our choice of c and n, for this pair $(f, D_f) \in \mathcal{F}_2$ we have $\Pr\{S_f$ is consistent with a monotone function$\} \geq 1 - \frac{3 \cdot 4 \cdot 2^{2\alpha d/3}}{2^{\alpha d}} = 1 - \frac{12}{2^{\alpha d/3}} > \frac{1}{3}$ (for sufficiently large d). □

[10] This is without loss of generality, although at a cost of doubling the tester query complexity.

It remains to prove Lemma 5.5. To do so, we first state the condition that a knowledge sequence has to satisfy in order to be consistent with some function in $\mathcal{F}_1$. Note that in order for a knowledge sequence to be extendable to a monotone function in $\mathcal{F}_1$, it is not enough that S_f is consistent with some monotone function. If for some couple (x_1, x_2) with prefix p, the sequence S_f contains points $y_1 \in U(x_1)$ and $y_2 \in L(x_2)$, both with the same prefix p, then we can deduce that $f \in \mathcal{F}_2$ regardless of whether y_1 and y_2 are comparable or not. Therefore, we have to define a relaxed notion of a witness for nonmonotone functions for our setting such that every sequence that does not contain a witness is indeed extendable to a function in $\mathcal{F}_1$. (In general, it is enough to show that there exists an *arbitrary* monotone function that is consistent with the knowledge sequence and is not necessarily a function in $\mathcal{F}_1$. However, we will use this fact later when extending the proof of the lower bound for the two-sided error case.)

DEFINITION 5.7. *Let S_f^i be a knowledge sequence learnt by A during the first i steps of some execution on the pair $(f, D_f) \in \mathcal{F}_2$. We say that S_f^i contains a* witness *if there exist points y_1, y_2 in S_f^i such that $y_1 \in U(x_1)$ and $y_2 \in L(x_2)$ for some couple (x_1, x_2) in f.*

LEMMA 5.8. *If the knowledge sequence S_f, learnt by A during some execution on a pair $(f, D_f) \in \mathcal{F}_2$, does not contain a witness, then there exists a function $f' \in \mathcal{F}_1$ that is consistent with S_f.*

Proof. To prove the existence of such a function f', we show how to construct the sets X_1' and X_2' such that f', the function induced by X_1' and X_2', is in $\mathcal{F}_1$ and is consistent with S_f. The construction of X_1' and X_2' is as follows:

1. For every couple (x_1, x_2) in f, if S_f contains points in $U(x_1)$, then add x_1 to X_1'; otherwise, if S_f contains points in $L(x_2)$, then add x_2 to X_2'. Let $\mathcal{M}_S$ be the set of $\frac{d}{2}$ length prefixes of all the points that were added to X_1' and to X_2'. Note that since S_f does not contain a witness, $\mathcal{M}_S = |X_1'| + |X_2'|$.
2. Choose $m - |X_1'|$ points in $\mathcal{M} \setminus \mathcal{M}_S$; denote the set of selected points by $\mathcal{M}_1$. For each prefix $p \in \mathcal{M}_1$, choose a point $y \in B_{d/4}^{d/2}$ and add $p||y$ to X_1'.
3. Denote by $\mathcal{M}_2$ the set $\mathcal{M} \setminus (\mathcal{M}_S \bigcup \mathcal{M}_1)$. For each prefix $p \in \mathcal{M}_2$, choose a point $y \in B_{(1/4+2\alpha)\cdot d}^{d/2}$ and add $p||y$ to X_2'.

Clearly, the function f' is S_f consistent. It remains to show that it is indeed a function in $\mathcal{F}_1$. Since S_f does not contain a witness, none of the prefixes in $\mathcal{M}$ was used more than once in the construction of X_1' and X_2'. Hence, the sets X_1' and X_2' can also be chosen in the process described in the construction of $\mathcal{F}_1$. Therefore, the function f' defined by the two sets X_1' and X_2' is indeed in $\mathcal{F}_1$. □

We now prove that the probability is very small that the knowledge sequence S_f, learnt by A during its run on a uniformly chosen pair $(f, D_f) \in \mathcal{F}_2$, contains a witness. To do so, we prove that the probability is very small that for every step of A the knowledge sequence contains a witness after that step. For this, we show that for every query that A may ask, there can be only one prefix in $\mathcal{M}$ such that this query belongs to $U(x_1)$ or $L(x_2)$ for some couple (x_1, x_2) with that prefix. We first prove the following technical lemma.

LEMMA 5.9. *Let f be a function in $\mathcal{F}_2$. Then, $U(x_1) \cap U(x_1') = \phi$ for every $x_1, x_1' \in X_1$ such that $x_1 \neq x_1'$. Similarly, $L(x_2) \cap L(x_2') = \phi$ for every $x_2, x_2' \in X_2$ such that $x_2 \neq x_2'$.*

Proof. Consider $x_1, x_1' \in X_1$ such that $x_1 \neq x_1'$ and assume, towards a contradiction, that there exists a point $y \in U(x_1) \cap U(x_1')$. Let p and p' be the prefixes of x_1 and x_1', respectively, and denote by p_y the $\frac{d}{2}$ length prefix of y. Since $y \in U(x_1)$, then $H(x_1, y) \leq \alpha d$, and hence $H(p, p_y) \leq \alpha d$. Similarly, $H(p', p_y) \leq \alpha d$. Therefore, $H(p, p') \leq 2\alpha d$, contradicting the requirements from $\mathcal{M}$. The proof for X_2 is similar. □

We can now deduce from the above lemma the following.

Lemma 5.10. *For every point $z \in \{y \in \{0,1\}^d : (\frac{1}{2} - \alpha)d \leq |y| \leq \frac{d}{2}\}$, there exists at most one prefix $p \in \mathcal{M}$ such that $z \in U(x)$ for some point $x \in B^d_{1/2-\alpha}$ with prefix p. Similarly, for every point $z \in \{y \in \{0,1\}^d : \frac{d}{2} < |y| \leq (\frac{1}{2} + \alpha)d\}$, there exists at most one prefix $p \in \mathcal{M}$ such that $z \in L(x)$ for some point $x \in B^d_{1/2+\alpha}$ with prefix p.*

Definition 5.11. *We say that the ith labelled point in a length i sequence S^i_f is* destructive *if S^{i-1}_f does not contain a witness and S^i_f does.*

Lemma 5.12. *For every tester A and every possible sequence of random coins for A, the following holds:*

$$\Pr\{S^{i+1}_f \textit{ contains a witness} \mid S^i_f \textit{ does not contain a witness}\} \leq \frac{i+2}{m},$$

where the probability is taken over the random choice of (f, D_f) from $\mathcal{F}_2$ that is consistent with S^i_f, and over the possible samplings of D_f for the chosen pair (f, D_f). (In other words, the probability that the $(i+1)$st labelled point in S^{i+1}_f is destructive can be bounded by $\frac{i+2}{m}$.)

Proof. We distinguish between the two stages of the tester: the sampling stage and the queries stage. In the first case, for every possible choice of (f, D_f) from $\mathcal{F}_2$, we show that the probability that S^{i+1}_f contains a witness is at most $\frac{i}{2m}$. Let (f, D_f) be a pair in $\mathcal{F}_2$ that is S^i_f consistent and let X_1 and X_2 be the sets used for its construction. The sampled point can be destructive only if S^i_f already contains a related point. That is, for some couple (x_1, x_2) in f, either the sampled point is x_1 and there are points in S^i_f from $L(x_2)$, or the sampled point is x_2 and S^i_f contains points from $U(x_1)$. By Lemma 5.10, every point in S^i_f can be in $U(x_1)$ for at most one point $x_1 \in X_1$, and similarly, every point can be in $L(x_2)$ for at most one point $x_2 \in X_2$. Hence, for every point x in S^i_f, only sampling of the couple of x can cause the knowledge sequence to contain a witness (recall that all the points in S^i_f are either from X_1 or from X_2).[11] The probability for this event is $\frac{1}{2m}$. Therefore, the probability that the sampled point returned by the oracle is destructive is bounded by $\frac{i}{2m}$.

In the second case, let q_{i+1} be the query asked by A in step $i+1$. Note that the query q_{i+1} can be computed by the algorithm using the information obtained during the first i steps of its execution, that is, the locations of the sampled points and the values of the queried points. If $|q_{i+1}| < (\frac{1}{2} - \alpha)d$ or $|q_{i+1}| > (\frac{1}{2} + \alpha)d$, then S^{i+1}_f cannot contain a witness. Assume, without loss of generality, that $|q_{i+1}| \leq \frac{d}{2}$ (the proof is similar for the case that $|q_{i+1}| > \frac{d}{2}$).

By Lemma 5.10, there can be at most one relevant prefix $p \in \mathcal{M}$ such that a couple (x_1, x_2) with prefix p can include q_{i+1} in $U(x_1)$. If S^i_f contains no 0-labelled points that can possibly be in $L(x_2)$ for a point x_2 with prefix p, then by the definition of a witness q_{i+1} cannot be destructive. Hence, assume there exists a 0-labelled point z in S^i_f that can possibly be in $L(x_2)$ for a point x_2 with prefix p.

[11] Note that, in general, if the points in $\mathcal{M}$ are not "far apart," it is possible for the couples to be very close to one another, and hence almost every sampled point is destructive.

The sequence S_f^{i+1} will contain a witness only if $q_{i+1} \in U(x_1)$ for the point x_1 corresponding to x_2. Note that S_f^i does not necessarily include x_2 itself, but only points in $L(x_2)$ (however, these points may affect the number of possible choices for x_1). We will bound the probability over the possible choices of a pair (f, D_f) such that f is S_f^i consistent, that indeed $q_{i+1} \in U(x_1)$ for the pair (x_1, x_2) for which S_f^i contains points in $L(x_2)$, and we will show that this probability is at most $\frac{2}{m}$.

We prove this bound by showing that for every value for x_2 with prefix p that is S_f^i consistent, the probability that $q_{i+1} \in U(x_1)$ for x_1 (the couple of x_2) that is S_f^i consistent can be upper bounded by $\frac{2}{m}$. We claim that this is sufficient to prove the bound for a uniformly chosen pair $(f, D_F) \in \mathcal{F}_2$, and we delay the proof of this claim for now.

The bound will be shown through the following three steps. In step 1, we give an upper bound on the number of possible choices for x_1 such that $q_{i+1} \in U(x_1)$. In step 2, we show a lower bound on the number of choices for x_1, given the point x_2, that are S_f^i consistent. Finally, in step 3 we combine these two bounds to get an upper bound on the probability of the desired event.

1. Since x_1 must have the prefix p, we bound the number of points $x \in B^d_{(1/2-\alpha)\cdot d}$ with prefix p such that $x < q_{i+1}$. Clearly, this number is maximal when q_{i+1} also has the prefix p and $|q_{i+1}| = \frac{d}{2}$. Therefore, the number of possible choices for x_1 such that $q_{i+1} \in U(x_1)$ is at most $k \overset{\text{def}}{=} \binom{(\frac{1}{4}+\alpha)d}{\alpha d}$.
2. Our choice of x_1 is constrained by the point x_2 and by 0-labelled points which we know could have been in $U(x_1)$ for possible choices of x_1. The number of choices for x_1 given x_2 is bounded by $K \overset{\text{def}}{=} \binom{(\frac{1}{4}+2\alpha)d}{2\alpha d}$ (the first $\frac{d}{2}$ coordinates are always p). However, our choice is also constrained by the 0-labelled points known in S_f^i that could have been in $U(x_1)$. Using arguments similar to the ones that were used in step 1, each of these points can eliminate at most k choices for x_1. Hence, there are at least $K - ik$ possible choices for x_1 that are S_f^i consistent.
3. We can deduce that the probability that, given x_2, a uniformly drawn point x_1, which is S_f^i consistent, satisfies $q_{i+1} \in U(x_1)$ can be upper bounded by $\frac{k}{K - i \cdot k}$. To bound this probability, we first bound the ratio $\frac{k}{K}$. We have

$$\frac{k}{K} = \frac{((\frac{1}{4}+\alpha)d)!(2\alpha d)!}{((\frac{1}{4}+2\alpha)d)!(\alpha d)!} = \frac{(\alpha d+1)\dots 2\alpha d}{((\frac{1}{4}+\alpha)d+1)\dots(\frac{1}{4}+2\alpha)d}$$
$$< \left(\frac{2\alpha d}{(\frac{1}{4}+2\alpha)d}\right)^{\alpha d} < (8\alpha)^{\alpha d}.$$

Therefore, the probability that $q_{i+1} \in U(x_1)$ is bounded by $\frac{(8\alpha)^{\alpha d}}{1 - i(8\alpha)^{\alpha d}}$. By the choice of $\alpha < \frac{1}{16}$, we have $(8\alpha)^{\alpha d} < \frac{1}{2}^{\alpha d} = \frac{1}{m}$. Hence, $\frac{(8\alpha)^{\alpha d}}{1-i(8\alpha)^{\alpha d}} < \frac{\frac{1}{m}}{1-\frac{i}{m}} = \frac{1}{m-i}$. Since $i < \frac{m}{2}$, this probability is bounded by $\frac{2}{m}$.

It remains to show why, in order to bound the probability that a uniformly drawn pair in $\mathcal{F}_2$ that is S_f^i consistent satisfies $q_{i+1} \in U(x_1)$, it is enough to bound the probability that $q_{i+1} \in U(x_1)$ for a uniformly drawn S_f^i consistent point x_1, given x_2 (the couple of x_1) S_f^i consistent.

We first observe that all the functions in $\mathcal{F}_2$ that are S_f^i consistent contain a couple with the prefix p (otherwise, q_{i+1} cannot be destructive). We now argue that the distribution induced by a random choice of an S_i^f consistent pair in $\mathcal{F}_2$ over the prefix p couples (x_1, x_2) is uniform. To see this, note that every couple (x_1, x_2) with prefix p that is S_f^i consistent can be extended to a pair (f, D_f) that is S_f^i consistent (by choosing all the other points in X_1 and X_2) in the same number of ways. That is, the number of possibilities for extending the sample S_i^f to a pair in $\mathcal{F}_2$ is independent of the specific choice of the pair (x_1, x_2). Therefore, once the prefix of the couple is determined, given the point x_2, the distribution induced over the possible choices of x_1 that are S_f^i consistent is uniform (since the distribution induced over all choices is uniform). In addition, for every such function f, the question of whether indeed $q_{i+1} \in U(x_1)$ is fully determined by the choice of the prefix p couple (x_1, x_2). Hence, it is enough to bound the probability that, given any point x_2 that is S_f^i consistent, indeed $q_{i+1} \in U(x_1)$ for a uniformly chosen x_1 that is S_f^i consistent. □

Based on the above lemma, we can now prove the following proposition.

Proposition 5.13. *For every tester A with query and sample complexity n and every possible choice of random coins for A,*

$$\Pr\{S_f \textit{ contains a witness}\} \leq \frac{3n^2}{m},$$

where the probability is taken over the random choice of the pair (f, D_f) from $\mathcal{F}_2$ and over the possible sampling of D_f for the chosen pair (f, D_f).

Proof. Given a tester A and a choice of random coins for A, the probability that S_f contains a witness for a uniformly chosen pair $(f, D_f) \in \mathcal{F}_2$ is bounded by

$$\sum_{i=1}^{2n} \Pr\{S_f^{i+1} \text{ contains a witness} \mid S_f^i \text{ does not contain a witness}\}.$$

By Lemma 5.12, this probability can be bounded by $\sum_{i=1}^{2n} \frac{i+2}{m} = \frac{n(2n+1)+4n}{m} < \frac{3n^2}{m}$. □

Lemma 5.5 follows immediately from the above proposition (notice that Proposition 5.13 implies Lemma 5.5 for every choice of random coins and not just with high probability over the choice of random coins for A, as stated in Lemma 5.5).

5.3. Lower bound for two-sided error testing. In this section we extend Theorem 5.6 to the two-sided error case; that is, we prove that there exists no two-sided error distribution-free monotonicity tester with subexponential query complexity that accepts every pair $(f, D_f) \in \mathcal{F}_1$, with high probability, and rejects every pair $(f, D_f) \in \mathcal{F}_2$, with high probability. To do so, we show that there exists a constant c such that for every tester A that makes fewer than 2^{cd} membership queries and samples D_f less than 2^{cd} times, the distributions induced on the set of possible knowledge sequences by running A on a random pair from $\mathcal{F}_1$ and from $\mathcal{F}_2$ are statistically close. By Yao's principle, the claim follows.

Let A be a tester with query and sample complexity $n < 2^{cd}$. Denote by $\mathcal{P}_1^A$ the distribution induced over length n knowledge sequences by running A on a uniformly drawn pair (f, D_f) from $\mathcal{F}_1$ (note that $\mathcal{P}_1^A$ is induced by the random choice of the pair (f, D_f) in $\mathcal{F}_1$, the choice of random coins for A, and the sampling of the domain according to D_f); the distribution $\mathcal{P}_2^A$ is defined similarly. We show that for every such tester A, the statistical difference between the two distributions $\mathcal{P}_1^A$ and $\mathcal{P}_2^A$ is exponentially small.

In the previous section we saw that for every tester A, the probability, measured with respect to $\mathcal{P}_2^A$, that a knowledge sequence contains a witness is very small. At first glance, it may seem that this is enough to show that the two distributions $\mathcal{P}_1^A$ and $\mathcal{P}_2^A$ are statistically close. However, this is not true. In the case of functions in $\mathcal{F}_1$, for every prefix $p \in \mathcal{M}$ there exists a point either in X_1 or in X_2 with the prefix p (m of the prefixes in $\mathcal{M}$ are used as prefixes for points in X_1, while the other m are used as prefixes for points in X_2). This is not the case for functions in $\mathcal{F}_2$, where we choose only m out of the $2m$ prefixes and select a couple (x_1, x_2) with each prefix. Hence, one can suggest the following testing approach:

1. Repeat the following two steps:
   (a) Choose a prefix $p \in \mathcal{M}$.
   (b) Decide, using membership queries, whether there exists a point either in X_1 or in X_2 with the prefix p.
2. If for at least $\frac{1}{4}$ of the prefixes chosen in step 1, no point was found in X_1 or in X_2 with that prefix, decide that the function is in $\mathcal{F}_2$; otherwise, decide that the function is in $\mathcal{F}_1$.

(Such an approach is not relevant in the case of one-sided error testing, since the tester has to accept every function in $\mathcal{F}_1$ with probability 1.) Clearly, if the tester is only allowed to sample the distribution D_f, this testing approach requires an exponential number of queries. However, it may be hypothetically possible to use membership queries to significantly reduce the query complexity. Hence, it is not enough to show that with high probability the knowledge sequence S_f does not contain a witness, and there are other undesirable events that we have to eliminate.

We formally define each of the undesirable events and prove that with high probability, both under $\mathcal{P}_1^A$ and under $\mathcal{P}_2^A$, these events do not occur in the knowledge sequence learnt by A. As before, given a pair (f, D_f), we denote by S_f^i the knowledge sequence learnt by A during the first i steps of its execution on (f, D_f) (for some possible execution of A on (f, D_f)). The first undesirable event is that the tester succeeds in learning useful information about the given pair (f, D_f) during the querying stage (and not only in the sampling stage).

Definition 5.14. *Let S_f^i be a knowledge sequence learnt by A during the first i steps of some execution on the pair (f, D_f). We say that S_f^i contains* nonsampled points *if either, for some $x_1 \in X_1$ that was not sampled in the first stage of the tester, S_f^i contains points in $U(x_1)$, or for some $x_2 \in X_2$ that was not sampled in the first stage of the tester, S_f^i contains points in $L(x_2)$.*

We first prove that this event is unlikely under $\mathcal{P}_1^A$. The proof is similar to the proof of Lemma 5.12.

Lemma 5.15. *For every tester A and every possible sequence of random coins for A, the following holds:*

$$\Pr\{\textit{nonsampled points in } S_f^{i+1} \mid \textit{no nonsampled points in } S_f^i\} \leq \frac{2}{m},$$

where the probability is taken over the random choice of (f, D_f) from $\mathcal{F}_1$ that is consistent with S_f^i, and over the possible samplings of D_f for the chosen pair (f, D_f).

Proof. By definition, in order for S_f^{i+1} to contain nonsampled points, $i+1$ must be in the queries stage. That is, in step $i+1$, the tester queries the function. Let q_{i+1} be the query performed by the tester in step $i+1$ based on all the information obtained by the tester during the first i steps of the execution and assume, without loss of generality, that $(\frac{1}{2} - \alpha)d \leq |q_{i+1}| \leq \frac{d}{2}$. We show that the probability of $q_{i+1} \in U(x_1)$, for some point $x_1 \in X_1$ not sampled by the tester in the sampling stage, is bounded by $\frac{2}{m}$. By Lemma 5.10, there exists at most one prefix $p \in \mathcal{M}$ such that q_{i+1} can be in $U(x_1)$ for a point $x_1 \in X_1$ with prefix p. Hence, S_f^i includes no point $x \in X_1 \cup X_2$ with prefix p (otherwise, the probability that S_f^{i+1} contains nonsampled points is 0). As in the proof of Lemma 5.12, once the prefix of a point $x_1 \in X_1$ is determined, the distribution induced by the random choice of the pair $(f, D_f) \in \mathcal{F}_1$ on the choice of x_1 is uniform. Thus, to bound the probability for a random pair $(f, D_f) \in \mathcal{F}_1$ that is S_f^i consistent that $q_{i+1} \in U(x_1)$ for $x_1 \in X_1$, it is enough to bound the probability of $q_{i+1} \in U(x_1)$ for a uniformly chosen x_1 with prefix p that is S_f^i consistent. The arguments used for this bound are simpler than those used in the proof of Lemma 5.12, since this time the choice of x_1 is constrained only by 0-labelled points that could have possibly been in $U(x_1)$, and not by the choice of the couple x_2 (since the functions in question are all in $\mathcal{F}_1$).

Our bound will be obtained through the following three steps. In step 1, we bound from above the number of possible choices for x_1 such that $q_{i+1} \in U(x_1)$. In step 2, we bound from below the number of possible choices for x_1 that are S_f^i consistent. Finally, in step 3 we combine the two bounds to obtain an upper bound on the probability of $q_{i+1} \in U(x_1)$ for a random x_1 that is S_f^i consistent.

1. Since x_1 must have the prefix p, we bound the number of points $x \in B^d_{1/2-\alpha}$ with prefix p such that $x < q_{i+1}$. As before, this number is maximal when q_{i+1} also has the prefix p and $|q_{i+1}| = \frac{d}{2}$. Therefore, the number of possible choices for x_1 such that $q_{i+1} \in U(x_1)$ is at most $k \stackrel{\text{def}}{=} \binom{(\frac{1}{4}+\alpha)d}{\alpha d}$.
2. The number of possible choices for x_1 given only the prefix p is $K \stackrel{\text{def}}{=} \binom{d/2}{d/4}$ (the first $d/2$ coordinates are always p). However, our choice of x_1 is constrained by the 0-labelled points in S_f^i that could have possibly been in $U(x_1)$ for possible choices of x_1. Using arguments similar to step 1, each of these points can eliminate at most k choices for x_1. Hence, there are at least $K - ik$ possible choices for x_1 that are S_f^i consistent.
3. We conclude that the probability of a uniformly drawn point x_1 that is S_f^i consistent satisfying $q_{i+1} \in U(x_1)$ can be upper bounded by $\frac{k}{K-i\cdot k}$. We have
$$\frac{k}{K} = \frac{((\frac{1}{4}+\alpha)d)!(\frac{d}{4})!^2}{(\frac{d}{2})!(\frac{d}{4})!(\alpha d)!} = \frac{1\dots\frac{d}{4}}{1\dots\alpha d\cdot(\frac{1}{4}+\alpha)d\dots\frac{d}{2}} < \left(\frac{1}{2}\right)^{(\frac{1}{4}-\alpha)d} < \left(\frac{1}{2}\right)^{\frac{d}{8}}.$$
Therefore, the probability of $q_{i+1} \in U(x_1)$ can be bounded by $\frac{(\frac{1}{2})^{\frac{d}{8}}}{1-i(\frac{1}{2})^{\frac{d}{8}}}$. By the choice of $\alpha < \frac{1}{16}$ and the fact that $i < m$, we have $i\cdot\frac{1}{2}^{\frac{d}{8}} < \frac{1}{2}$ for sufficiently large values of d. Hence, $\frac{(\frac{1}{2})^{\frac{d}{8}}}{1-i(\frac{1}{2})^{\frac{d}{8}}} < 2(\frac{1}{2})^{\frac{d}{8}} < \frac{2}{m}$. □

Based on the above lemma, we now prove the following proposition.

PROPOSITION 5.16. *For every tester A and every possible choice of random coins for A,*

$$\Pr\{S_f \text{ contains nonsampled points}\} \le \frac{4n}{m},$$

where the probability is taken over the random choice of the pair (f, D_f) from $\mathcal{F}_1$ and over the possible sampling of D_f for the chosen pair (f, D_f).

Proof. Given a tester A and a choice of random coins for A, the probability that S_f contains nonsampled points for a uniformly chosen pair $(f, D_f) \in \mathcal{F}_1$ can be bounded by

$$\sum_{i=1}^{2n} \Pr\{S_f^{i+1} \text{ contains nonsampled points} \mid S_f^i \text{ does not contain nonsampled points}\}.$$

By Lemma 5.15, this probability can be bounded by $\sum_{i=1}^{2n} \frac{2}{m} = \frac{4n}{m}$. □

We now show that this event is unlikely also under $\mathcal{P}_2^A$.

LEMMA 5.17. *For every tester A and every possible sequence of random coins for A, the following holds:*

$$\Pr\{\textit{nonsampled points in } S_f^{i+1} \mid \textit{no nonsampled points in } S_f^i\} \leq \frac{i+2}{m},$$

where the probability is taken over the random choice of (f, D_f) from $\mathcal{F}_2$ consistent with S_f^i, and over the possible samplings of D_f for the chosen pair (f, D_f).

Proof. Clearly, the probability that, for a uniformly drawn pair $(f, D_f) \in \mathcal{F}_2$ consistent with S_f^i, the point q_{i+1} is in $U(x_1)$ for a point $x_1 \in X_1$ not known in S_f^i, can be bounded by the probability that, for such a pair (f, D_f), indeed $q_{i+1} \in U(x_1)$, given that x_2 (the couple of x_1), appears in S_f^i (the knowledge of x_2 can only raise the probability of the tester to choose a point $q_{i+1} \in U(x_1)$). However, this probability was already shown in Lemma 5.12 to be bounded by $\frac{i+2}{m}$. □

The following proposition is proved in a similar way to Proposition 5.13, with Lemma 5.17 playing the role of Lemma 5.12.

PROPOSITION 5.18. *For every tester A and every possible choice of random coins for A,*

$$\Pr\{S_f \textit{ contains nonsampled points}\} \leq \frac{3n^2}{m},$$

where the probability is taken over the random choice of the pair (f, D_f) from $\mathcal{F}_2$ and over the possible sampling of D_f for the chosen pair (f, D_f).

The next undesirable event is that, during the sampling stage, the tester receives the same sample more than once. We show that this event is unlikely both under $\mathcal{P}_1^A$ and under $\mathcal{P}_2^A$.

DEFINITION 5.19. *A knowledge sequence S_f is* repetitive *if there exists a point in $X_1 \cup X_2$ that was sampled at least twice in S_f.*

PROPOSITION 5.20. *For every tester A and every possible choice of random coins for A,*

$$\Pr\{S_f \textit{ is repetitive}\} \leq \frac{n^2}{m},$$

where the probability is taken over the random choice of the pair (f, D_f) from $\mathcal{F}_1$ or from $\mathcal{F}_2$ and over the possible sampling of D_f for the chosen pair (f, D_f).

Proof. To prove the lemma, we show that given a set L of size m, the probability that at least one of the points in L appears more than once in a random sample of size n is bounded by $\frac{n^2}{m}$. The probability that the ith sample from L is a point that was sampled before can be bounded by $\frac{i}{m}$. Hence, using the union bound, the probability that at least one of the n samples repeats a point that was sampled before is bounded by $\sum_{i=1}^{n} \frac{i}{m} = \frac{n \cdot (n+1)}{2m} \leq \frac{n^2}{m}$. □

We conclude that with very high probability, under both distributions, none of the undesirable events occur.

Definition 5.21. *We say that a knowledge sequence S_f is* good *if S_f does not contain a witness nor nonsampled points and is not repetitive; otherwise, we say that S_f is* bad.

Note that, if a knowledge sequence S_f is good, then the sequence is fully determined by the answers received from the sampling oracle in the first stage of the tester. That is, unless a query q is in $U(x)$ or $L(x)$ for a point x that was sampled in the first stage of the tester, the answer that the tester receives is 0 if $|q| \le \frac{d}{2}$, and 1 otherwise.

By Propositions 5.13, 5.16, 5.18, and 5.20, for every tester A, with probability at least $1-\frac{8n^2}{m}$, both with respect to $\mathcal{P}_1^A$ and with respect to $\mathcal{P}_2^A$, the knowledge sequence learnt by A is good. We show that for every good knowledge sequence S_f, the two probabilities $\mathcal{P}_1^A(S_f)$ and $\mathcal{P}_2^A(S_f)$ are very close. That is, we have the following.

Lemma 5.22. *For every good knowledge sequence S_f it holds that*

$$1 - \frac{O(n^2)}{m} \le \frac{\mathcal{P}_1^A(S_f)}{\mathcal{P}_2^A(S_f)} \le 1 + \frac{O(n^2)}{m}.$$

Proof. To prove the lemma, we show that, for every step that the tester performs (sample or query), and for every answer it receives, the ratio between the probability of this answer with respect to both a random function from $\mathcal{F}_1$ and a random function from $\mathcal{F}_2$ is bounded by $(1 \pm \frac{n}{2m-n})$. Recall that we assume the tester has two stages. The first stage is the sampling stage, while the second is dedicated to the membership queries of the tester. Since the sequences in question are all good, the answers that the tester receives in the second stage are fully determined by the answers it received in the sampling stage. Hence, the probability of these answers under both distributions is 1 and thus is identical. Therefore, for every knowledge sequence, the ratio between the probability of this sequence under the two distributions can be bounded from above by $(1+\frac{n}{2m-n})^n \cdot 1^n = (1+\frac{O(n^2)}{m})$ and from below by $(1-\frac{n}{2m-n})^n \cdot 1^n = (1-\frac{O(n^2)}{m})$ (where the equality follows the binomial expansion of $(1+\frac{n}{2m-n})^n$).[12]

It remains to show that indeed the ratio between the two probabilities holds with respect to the first stage of the tester, the sampling stage. Let x be the answer returned to the tester in the $(i+1)$st sample and assume, without loss of generality, that $|x| = (1/2-\alpha)d$. Let n_1 be the number of weight $(1/2-\alpha)d$ points sampled before x (that is, points from X_1). Denote by K the number of points in $B_{1/2}^{d/2}$. Hence, the probability under $\mathcal{P}_1^A$ that the tester receives a sampling of x is the probability that, in a function $f \in \mathcal{F}_1$ that is S_f^i consistent, the point x belongs to X_1 and that x was sampled according to D_f. It is easy to see that the distribution induced by $\mathcal{P}_1^A$ over the choice of prefix for the sampled point is the uniform distribution defined over the set of prefixes that have not yet appeared in the points sampled prior to x. In addition, once the prefix of x is determined, the choice of x is uniformly distributed over all K prefix p points. Hence, the probability under $\mathcal{P}_1^A$ that the tester receives a sampling of x is $\frac{m-n_1}{2m-i} \cdot \frac{1}{2m-i} \cdot \frac{1}{K}$, where the first term is the probability that the next point drawn according to D_f is in X_1, the second term is the probability that the prefix used for the next point is indeed the prefix p_x of x, and the third is the probability of choosing x out of all prefix p_x points. We now bound the same probability with respect to the distribution $\mathcal{P}_2^A$. In the case of functions from $\mathcal{F}_2$, the distribution

[12] $(1+\frac{n}{2m-n})^n \le \Sigma_{k=1}^n n^k (\frac{n}{2m-n})^k = \Sigma_{k=1}^n (\frac{n^2}{2m-n})^k \le \Sigma_{k=1}^\infty (\frac{n^2}{2m-n})^k$, where the last inequality follows the fact that $2m-n>0$. Since $m > n^2+n$, we have $\Sigma_{k=1}^\infty (\frac{n^2}{2m-n})^k = \frac{1}{1-\frac{n^2}{2m-n}} = \frac{2m-n}{2m-n-n^2}$ $= (1+\frac{n^2}{2m-n-n^2}) \le (1+\frac{n^2}{m})$. The expansion for $(1-\frac{n}{2m-n})^n$ is similar.

induced over the prefix of the drawn point is again uniform, as is the distribution induced over the possible choices of the point once the prefix has been determined. However, the probability that the drawn point belongs to X_1 is $\frac{1}{2}$, implying that the probability of choosing x in this case is $\frac{1}{2} \cdot \frac{1}{2m-i} \cdot \frac{1}{K}$. Hence, the ratio between the two probabilities is $\frac{2m-2n_1}{2m-i}$ and is bounded from above by $(1 + \frac{n}{2m-n})$ and from below by $(1 - \frac{n}{2m-n})$ as required.[13] $\square$

Therefore, the statistical difference between the two distributions $\mathcal{P}_1^A$ and $\mathcal{P}_2^A$ is bounded by

$$\frac{1}{2}\left(\sum_{S_f : S_f \text{ is bad}} \max\{\mathcal{P}_1^A(S_f), \mathcal{P}_2^A(S_f)\} + \sum_{S_f : S_f \text{ is good}} \mathcal{P}_2^A(S_f) \cdot \frac{O(n^2)}{m}\right).$$

Since under both distributions the probability that the knowledge sequence is bad is exponentially small, the statistical difference between the two distributions is bounded by $\frac{O(n^2)}{m}$. This implies the following theorem.

THEOREM 5.23. *There exists a constant c such that testing monotonicity of Boolean functions defined over the Boolean d-dimensional hypercube, in the distribution-free setting, requires 2^{cd} queries.*

6. Discussion and open problems. In this work we introduce the first *distribution-free* testers for some of the central problems studied in the property-testing literature: low-degree multivariate polynomials testing and monotonicity testing in the low-dimensional case. We show that a low-degree test can be obtained as a special case of a more general distribution-free scheme presented for a wider class of properties, thereby giving sufficient conditions for the existence of a distribution-free tester. By this, we answer a natural question that has already been raised explicitly by Fischer [19, subsection 9.3] and is implicit in [25]. In addition, by showing a lower bound on the query complexity required for distribution-free monotonicity testing in the high-dimensional case, we show that distribution-free testing, even if possible with nontrivial query complexity, cannot always be done using a query complexity similar to that used in the uniform setting.

However, there are many open questions with respect to distribution-free testing and we are still only beginning to explore them. The first problem that remains open is trying to narrow the gap between the query complexity of the known distribution-free monotonicity tester for the high-dimensional case (which is exponential in $\log n$) and our lower bound (which is only exponential in d). In addition, now that we already know it is possible to construct distribution-free testers for nontrivial problems, we wish to further study the existence of such testers for different problems that we know how to test with respect to the uniform distribution. Eventually, our goal will be to find characterizations (and not only sufficient conditions) for problems that can be efficiently tested in a distribution-free manner, given that they can be efficiently tested with respect to the uniform distribution.

Another interesting possible direction is to relax the requirements for distribution-free testing. There are certain problems for which testers exist for the uniform distribution case, and it has been proven that they cannot be efficiently tested in the distribution-free setting. Among these are the partition problems in the dense graph

[13] Note that $\frac{2m-2n_1}{2m-i} = 1 + \frac{i-2n_1}{2m-i}$. Since $0 \leq n_1 \leq i \leq n$, we have $2m - n \leq 2m - i$, and $-n \leq i - 2n_1 \leq n$.

model that were studied in [25]. In these cases, it is interesting to try relaxing the distribution-free testing requirement by allowing a stronger oracle to the input function f. One possibility is to enable the algorithm to ask queries in the following form: Is there a value x in a subdomain $\mathcal{X}' \subseteq \mathcal{X}$ of the function f, for which $f(x) = y$ (for a specific value of y)? This kind of oracle access seems relevant, for example, in monotonicity testing when, given a point $z \in \mathcal{X}$ such that $f(z) = 1$, we wish to find a point $x \in \{v \in \mathcal{X} : z \leq v\}$ such that $f(x) = 0$, and it was already introduced in the context of learning theory. Notice that the strength of the oracle is determined by the kind of specification we allow the algorithm to give on the subdomains. Another possibility is to allow the algorithm to ask for the actual probability of the points it sampled according to D (in the distribution-free model the tester has no knowledge of the probability of the sampled points) and to use it in its decision process. Finally, there is always the possibility of distribution-known testing, in which the distribution D is known to the tester in advance. This is very different from the distribution-free case, where we are only allowed to sample according to the distribution D, but have no actual knowledge of D.

Acknowledgments. The lower bound proof for monotonicity testing in the high-dimensional case was initiated by a discussion with Nader Bshouty. We wish to thank Nader for sharing his ideas with us; they played a significant role in the development of the results presented in section 5.

REFERENCES

[1] N. Alon, S. Dar, M. Parnas, and D. Ron, *Testing of clustering*, in Proceedings of the Symposium on Foundations of Computer Science, IEEE, Los Alamitos, CA, 2000, pp. 240–251.

[2] N. Alon, E. Fischer, M. Krivelevich, and M. Szegedy, *Efficient testing of large graphs*, Combinatorica, 20 (2000), pp. 451–476; also in Proceedings of the Symposium on Foundations of Computer Science, IEEE, Los Alamitos, CA, 1999, pp. 656–666.

[3] N. Alon, T. Kaufman, M. Krivelevich, S. Litsyn, and D. Ron, *Testing low-degree polynomials over GF*(2), in Approximation, Randomization and Combinatorial Optimization (RANDOM-APPROX), Springer, Berlin, 2003, pp. 188–199.

[4] N. Alon, M. Krivelevich, I. Newman, and M. Szegedy, *Regular languages are testable with a constant number of queries*, SIAM J. Comput., 30 (2001), pp. 1842–1862; preliminary version appeared in Proceedings of the Symposium on Foundations of Computer Science, IEEE, Los Alamitos, CA, 1999, pp. 645–655.

[5] N. Alon and A. Shapira, *A characterization of easily testable induced subgraphs*, in Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), ACM, New York, SIAM, Philadephia, 2004, pp. 935–944.

[6] N. Alon and A. Shapira, *A characterization of the (natural) graph properties testable with one-sided error*, in Proceedings of the 46th Annual Symposium on Foundations of Computer Science, IEEE, Los Alamitos, CA, 2005, pp. 429–438.

[7] N. Alon and A. Shapira, *Every monotone graph property is testable*, in Proceedings of the Symposium on Theory of Computing, ACM, New York, 2005, pp. 128–137.

[8] N. Alon and A. Shapira, *Testing satisfiability*, in Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), ACM, New York, SIAM, Philadephia, 2002, pp. 645–654.

[9] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, *Proof verification and the hardness of approximation problems*, J. ACM, 45 (1998), pp. 501–555; preliminary version appeared in Proceedings of the 33rd Annual Symposium on Foundations of Computer Science, IEEE, Los Alamitos, CA, 1992, pp. 14–23.

[10] S. Arora and S. Safra, *Probabilistic checkable proofs: A new characterization of NP*, J. ACM, 45 (1998), pp. 70–122; preliminary version appeared in Proceedings of the 33rd Annual Symposium on Foundations of Computer Science, IEEE, Los Alamitos, CA, 1992, pp. 2–13.

[11] S. Arora and M. Sudan, *Improved low-degree testing and its applications*, in Proceedings of the Symposium on Theory of Computing, ACM, New York, 1997, pp. 485–495.

[12] T. Batu, R. Rubinfeld, and P. White, *Fast approximation PCPs for multidimensional bin-packing problems*, in Randomization, Approximation, and Combinatorial Optimization (RANDOM-APPROX), Springer, Berlin, 1999, pp. 246–256.

[13] M. Blum, M. Luby, and R. Rubinfeld, *Self testing/correcting with applications to numerical problems*, J. Comput System Sci., 47 (1993), pp. 549–595.

[14] A. Bogdanov, K. Obata, and L. Trevisan, *A lower bound for testing* 3*-colorability in bounded-degree graphs*, in Proceedings of the Symposium on Foundations of Computer Science, IEEE, Los Alamitos, CA, 2002, pp. 93–102.

[15] A. Czumaj and C. Sohler, *Testing hypergraph coloring*, in Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP), Springer, Berlin, 2001, pp. 493–505.

[16] Y. Dodis, O. Goldreich, E. Lehman, S. Raskhodnikova, D. Ron, and A. Samorodnitsky, *Improved testing algorithms for monotonicity*, in Randomization, Approximation, and Combinatorial Optimization (RANDOM-APPROX), 1999, Springer, Berlin, pp. 97–108.

[17] E. Ergün, S. Kannan, R. Kumar, R. Rubinfeld, and M. Viswanathan, *Spot-checkers*, J. Comput. System Sci., 60 (2000), pp. 717–751; preliminary version appeared in Proceedings of the Symposium on Theory of Computing, ACM, New York, 1999, pp. 259–268.

[18] E. Fischer, *On the strength of comparisons in property testing*, Inform. and Comput., 189 (2004), pp. 107–116.

[19] E. Fischer, *The art of uninformed decisions: A primer to property testing*, Bull. Eur. Assoc. Theor. Comput. Sci. EATCS, 75 (2001), pp. 97–126.

[20] E. Fischer, G. Kindler, D. Ron, S. Safra, and A. Samorodnitsky, *Testing juntas*, in Proceedings of the Symposium on Foundations of Computer Science, IEEE, Los Alamitos, CA, 2002, pp. 103–112.

[21] E. Fischer, E. Lehman, I. Newman, S. Raskhodnikova, R. Rubinfeld, and A. Samorodnitsky, *Monotonicity testing over general poset domains*, in Proceedings of the Symposium on Theory of Computing, ACM, New York, 2002, pp. 474–483.

[22] E. Fischer and I. Newman, *Testing of matrix properties*, in Proceedings of the Symposium on Theory of Computing, ACM, New York, 2001, pp. 286–295.

[23] O. Goldreich, *Combinatorial property testing (a survey)*, in Randomization Methods in Algorithm Design (AMS-DIMACS), AMS, Providence, RI, 1999, pp. 45–59.

[24] O. Goldreich, S. Goldwasser, E. Lehman, D. Ron, and A. Samorodnitsky, *Testing monotonicity*, Combinatorica, 20 (2000), pp. 301–337; preliminary version appeared in Proceedings of the 39th Annual Symposium on Foundations of Computer Science, IEEE, Los Alamitos, CA, 1998, pp. 426–435.

[25] O. Goldreich, S. Goldwasser, and D. Ron, *Property testing and its connection to learning and approximation*, J. ACM, 45 (1998), pp. 653–750; preliminary version appeared in Proceedings of the 37th Annual Symposium on Foundations of Computer Science, IEEE, Los Alamitos, CA, 1996, pp. 339–348.

[26] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan, and A. Wigderson, *Self testing/correcting for polynomials and for approximate functions*, in Proceedings of the Symposium on Theory of Computing, ACM, New York, 1991, pp. 32–42.

[27] O. Goldreich and D. Ron, *On Testing Expansion in Bounded-Degree Graphs*, Report TR00-020, ECCC, 2000. Available online at http://eccc.hpi-web.de/eccc-reports/2000/TR00-020/index.html

[28] O. Goldreich and D. Ron, *Property testing in bounded degree graphs*, in Proceedings of the Symposium on Theory of Computing, ACM, New York, 1997, pp. 406–415.

[29] O. Goldreich and L. Trevisan, *Three theorems regarding testing graph properties*, in Proceedings of the Symposium on Foundations of Computer Science, IEEE, Los Alamitos, CA, 2001, pp. 302–317.

[30] S. Halevy and E. Kushilevitz, *Distribution-free property testing*, in Approximation, Randomization and Combinatorial Optimization (RANDOM-APPROX), Springer, Berlin, 2003, pp. 302–317.

[31] S. Halevy and E. Kushilevitz, *Testing monotonicity over graph products*, in Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP), Springer, Berlin, 2004, pp. 721–732.

[32] S. Halevy and E. Kushilevitz, *Distribution-free connectivity testing*, in Approximation, Randomization and Combinatorial Optimization (RANDOM-APPROX), Springer, Berlin, 2004, pp. 393–404.

[33] S. Halevy and E. Kushilevitz, *A lower bound for distribution-free monotonicity testing,* in Approximation, Randomization and Combinatorial Optimization (RANDOM-APPROX), Springer, Berlin, 2005, pp. 330–341.

[34] T. Kaufman, M. Krivelevich, and D. Ron, *Tight bounds for testing bipartiteness in general graphs*, in Approximation, Randomization and Combinatorial Optimization (RANDOM-APPROX), Springer, Berlin 2003, pp. 341–353.

[35] Y. Kohayakawa, B. Nagle, and V. Rodl, *Efficient testing of hypergraphs*, in Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP), 2002, pp. 1017–1028.

[36] T. Kaufman and D. Ron, *Testing polynomials over general fields*, in Proceedings of the Symposium on Foundations of Computer Science, IEEE, Los Alamitos, CA, 2004, pp. 413–422.

[37] M. J. Kearns and U. V. Vzirani, *An Introduction to Computational Learning Theory*, MIT Press, Cambridge, MA, 1994.

[38] R. J. Lipton, *New directions in testing*, in Distributed Computing and Cryptography, DIMACS Ser. Discrete Math. Theoret. Comput. Sci. 2, AMS, Providence, RI, 1991, pp. 191–202.

[39] I. Newman, *Testing of functions that have small width branching programs*, in Proceedings of the Symposium on Foundations of Computer Science, IEEE, Los Alamitos, CA, 2000, pp. 251–258.

[40] M. Parnas and D. Ron, *Testing metric properties*, in Proceedings of the Symposium on Theory of Computing, ACM, New York, 2001, pp. 276–285.

[41] M. Parnas and D. Ron, *Testing the diameter of graphs*, in Randomization, Approximation, and Combinatorial Optimization (RANDOM-APPROX), Springer, Berlin, 1999, pp. 85–96.

[42] M. Parnas, D. Ron, and R. Rubinfeld, *Testing parenthesis languages*, in Randomization, Approximation, and Combinatorial Optimization (RANDOM-APPROX), Springer, Berlin, 2001, pp. 261–272.

[43] M. Parnas, D. Ron, and A. Samorodnitsky, *Proclaiming dictators and juntas or testing Boolean formulae*, in Approximation, Randomization and Combinatorial Optimization (RANDOM-APPROX), Springer, Berlin, 2001, pp. 273–284.

[44] D. Ron, *Property testing*, in Handbook of Randomized Computing, S. Rajasekaran, P. M. Pardalos, J. H. Reif, and J. D. P. Rolin, eds., Kluwer, Dordrecht, 2001, pp. 597–649.

[45] R. Rubinfeld, *Robust functional equations and their applications to program testing*, SIAM J. Comput., 28 (1999), pp. 1972–1997; preliminary version appeared in Proceedings of the 35th Annual Symposium of Foundations of Computer Science, IEEE, Los Alamitos, CA, 1994, pp. 288–299.

[46] R. Rubinfeld and M. Sudan, *Robust characterization of polynomials with applications to program testing*, SIAM J. Comput., 25 (1996), pp. 252–271; first appeared as a technical report, Cornell University, Ithaca, NY, 1993.

SIAM J. Comput.
Vol. 37, No. 4, pp. 1139–1162

UNIVERSAL BUFFERLESS PACKET SWITCHING*

COSTAS BUSCH†, MALIK MAGDON-ISMAIL‡, AND MARIOS MAVRONICOLAS§

Abstract. A packet-switching algorithm specifies the actions of the nodes in order to deliver packets in the network. A packet-switching algorithm is *universal* if it applies to any network topology and for any batch communication problem on the network. A long-standing open problem has concerned the existence of a universal packet-switching algorithm with near-optimal performance guarantees for the class of *bufferless* networks where the buffer size for packets in transit is zero. We give a positive answer to this question. In particular, we give a universal bufferless algorithm which is within a polylogarithmic factor from optimal for arbitrary batch problems: $\mathcal{T} = O\left(\mathcal{T}^* \cdot \log^3(n+N)\right)$, where $\mathcal{T}$ is the packet delivery time of our algorithm, $\mathcal{T}^*$ is the optimal delivery time, n is the size of the network, and N is the number of packets. At the heart of our result is a new *deterministic* technique for constructing a universal bufferless algorithm by emulating a store-and-forward algorithm on a transformation of the network. The main idea is to replace packet buffering in the transformed network with packet circulation in regions of the original network. The cost of the emulation on the packet delivery time is proportional to the buffer sizes used by the store-and-forward algorithm. We obtain the advertised result by using a store-and-forward algorithm with logarithmic sized buffers. The resulting bufferless algorithm is constructive and can be implemented in a distributed way.

Key words. optimal scheduling, routing, graph decomposition, deterministic bufferless emulation

AMS subject classifications. 68W15, 68W20, 90B20, 90B35, 90B36

DOI. 10.1137/050642095

1. Introduction.

1.1. Motivation.

In a communication network, two or more packets collide when they wish to follow the same link at the same time. Typically, some of the colliding packets are stored in a buffer at the node where the collision occurs, until the collision is resolved (*store-and-forward* networks). Here, we examine the case where such buffers are unavailable (*bufferless* networks). At the same time, we do not allow packets to be dropped in collisions. Since buffers are unavailable and packets cannot be dropped, colliding packets must be deflected to neighboring nodes. This behavior of packets in a collision has led to communication algorithms on bufferless networks becoming known as *hot-potato* or *deflection* algorithms; here, we will simply call them *bufferless*. Bufferless algorithms are of practical interest since in optical networks the packets are propagated as lightwaves which are hard to buffer [45].

A packet-switching algorithm specifies the actions that nodes in the network fol-

*Received by the editors October 7, 2005; accepted for publication (in revised form) January 3, 2007; published electronically November 21, 2007. A preliminary version of this paper appears as "Universal Bufferless Routing," in *Proceedings of the 2nd Workshop on Approximation and Online Algorithms (WAOA)* (in conjunction with ALGO 2004), Lecture Notes in Comput. Sci. 3351, Springer-Verlag, Berlin, 2004, pp. 239–252.

http://www.siam.org/journals/sicomp/37-4/64209.html

†Department of Computer Science, Louisiana State University, 280 Coates Hall, Baton Rouge, LA 70803 (busch@csc.lsu.edu).

‡Department of Computer Science, Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180 (magdon@cs.rpi.edu).

§Department of Computer Science, University of Cyprus, P. O. Box 20537, Nicosia CY-1678, Cyprus (mavronic@ucy.ac.cy). This author's work was partially supported by the EU within the 6th Framework Programme under contract 001907 "Dynamically Evolving, Large Scale Information Systems" (DELIS).

low to deliver the packets. A packet-switching algorithm is *universal* if it applies to any network topology and can solve any batch problem on it, where an arbitrary set of packets has to be delivered in the network. Universal store-and-forward algorithms with optimal performance guarantees exist [24, 35, 33, 40, 43]. A long-standing and important open problem is to determine whether there exists a universal bufferless algorithm with performance close to that of store-and-forward algorithms. Here, we solve this problem in the affirmative by giving the *first* known universal bufferless algorithm with near-optimal performance. We analyze the performance of our algorithm for batch problems on a synchronous network model, which we now describe.

1.2. Network model. The communication network is a connected, unweighted, and undirected graph $G = (V, E)$, where $|V| = n$. In a *synchronous* network, time is divided into a sequence of discrete time steps. Edges are bidirectional and may be traversed by at most two packets at a time step, one packet in each direction.

At every time step, a node processes the incoming packets and then sends them to adjacent nodes. In store-and-forward networks, each node has three kinds of buffers: (i) an *injection buffer*, which stores the packets to be injected into the node (when the node is a packet source), (ii) *incoming edge-buffers*, of size one for every incident edge, which will store any packet received along the respective edge, (iii) *outgoing edge-buffers*, for every incident edge, which are the actual buffers for packets in transit. At every time step the node takes the packets from the incoming and injection buffers and either forwards them along incident links or places them in the outgoing buffers. If the outgoing buffer is full, then packets are dropped.

In bufferless networks, there are no outgoing edge-buffers; in other words, all outgoing edge-buffers have size zero. Further, packets may not be dropped. Thus, after the packet is injected into the network, it must traverse some edge at every time step (until it is absorbed). Preferably, the traversed edge brings the packet closer to the destination. However, due to collisions, this is not always possible, and the packet may be sent on an alternate edge taking it further from the destination; this event is called *deflection*.

In our model, bufferless networks still have the injection and incoming buffers. The incoming buffers help to process the incoming packets. The injection buffer is needed when a node has to inject packets and there are no available edges. So, the distinction between store-and-forward and bufferless networks is in the outgoing buffers which hold packets in transit.

1.3. Batch problems. We measure the efficiency of our bufferless algorithm on *batch problems* [32]. In a batch problem, we are given an arbitrary set of packets with the objective to deliver them to their destinations. Let $Q = (G, \Pi, \mathcal{S})$ denote a batch problem on graph G for a set of N packets $\Pi = \{\pi_1, \pi_2, \ldots, \pi_N\}$. Each packet π_i has source s_i and destination t_i. The set $\mathcal{S}$ contains all the pairs of sources and destinations for the respective packets; thus $\mathcal{S} = \{(s_1, t_1), (s_2, t_2), \ldots, (s_N, t_N)\}$.

We say that a set of paths $\mathcal{P} = \{p_1, p_2, \ldots, p_N\}$ *satisfies* batch problem Q if each path p_i is a path in G from the source s_i to the destination t_i of packet π_i. Typically, a packet-switching algorithm solves a batch problem Q by first selecting a set of paths $\mathcal{P}$ that satisfy Q (*routing*), and then sending the packets along the paths (*scheduling*). The *delivery time* of the packet switching algorithm is the number of time steps that elapse between the first packet injection and the last packet absorbtion at its destination.

It is useful to define some properties associated with a set of paths $\mathcal{P}$. A path $p \in \mathcal{P}$ could be specified either as a sequence of nodes or as a sequence of edges, and the length of the path, $|p|$, is the number of edges in the path. The *edge congestion* C is the maximum number of paths in $\mathcal{P}$ that use an edge in G; similarly, the *node congestion* $\overline{C}$ is the maximum number of paths that use a node in G; the *dilation* D is the maximum path length, $\max_{p_i \in \mathcal{P}} |p_i|$. Since at most one packet may traverse an edge in a particular direction during any time step, a lower bound on the delivery time is given by $\Omega(C + D)$. Any packet-switching algorithm, either bufferless or store-and-forward, obeys this lower bound.

Given a set of paths with edge congestion C and dilation D, there exist store-and-forward scheduling algorithms that deliver the packets in time $O(C + D)$ (plus logarithmic terms), which are optimal within constant factors for the given paths [24, 35, 33, 40, 43]. Let $\mathcal{P}^*$ denote the optimal set of paths which minimize $C + D$ for a given batch problem Q. Using the optimal paths, the packets can be delivered in optimal time (within constant factors) in store-and-forward networks. A long-standing open question is whether one can achieve near-optimal delivery time in bufferless networks as well. We answer this question in the affirmative.

1.4. Contribution. We show that for any batch problem Q on an arbitrary bufferless network G, a delivery time within logarithmic factors from optimal can be achieved. In particular, we give a randomized algorithm which does so, given the optimal paths.

THEOREM 1.1. *With probability* $1 - O((n + N)^{-\lambda})$*, for some constant* $\lambda > 0$*, any batch problem* Q *with* N *packets on an arbitrary bufferless network* G *with* n *nodes can be solved with delivery time* $O(\mathcal{T}^* \cdot \log^3(n + N))$*, where* $\mathcal{T}^*$ *is the optimal delivery time for* Q *(with or without buffers).*

In order to obtain this result, we actually prove that for any set of paths $\mathcal{P}$ with congestion C and dilation D, the packets can be delivered within time $O((C + D) \cdot \log^3(n + N))$. Thus, using the optimal set of paths $\mathcal{P}^*$, we obtain Theorem 1.1.

Our algorithm is universal, since it applies to arbitrary network topologies. It also applies to arbitrary batch problems. Given the set of paths $\mathcal{P}^*$, the algorithm is also constructive and can be implemented in a distributed manner. We do not address the issue of constructing the good (optimal) set of paths $\mathcal{P}^*$, which is an active area of research [5, 6, 44, 49]. Our focus is on the fundamental difference between buffered versus bufferless packet switching, which we show is small. We continue by describing the technique used in our algorithm.

1.5. Approach. Consider a batch problem $Q = (G, \Pi, \mathcal{S})$ in a bufferless network G. Let $\mathcal{P}$ be a set of paths that satisfy Q with edge congestion C and dilation D. Our goal is to deliver the packets in time $\mathcal{T} = O((C + D) \cdot \log^3(n + N))$. This is sufficient for proving Theorem 1.1.

If the packets are to be sent without any collisions, then there are no deflections and the only parameter that needs to be determined is the injection time of the packets. In [22], it is shown that it is an NP-hard problem to approximate efficiently the optimal injection times in a collision-free packet scheduling (by a reduction from the *vertex coloring* problem). Thus, packets need to be deflected in order to obtain a near-optimal solution in polynomial time. However, with deflections, it is hard to preserve packets along specific paths, since a packet may need to deviate from its path in order to give priority to packets that make progress.

Our approach to solving the problem is to control the packet deflections while the packets follow the paths in $\mathcal{P}$. We restrict the deflections in some particular areas of the network which are close to the original paths. By controlling the deflections in those areas, we effectively obtain a new set of paths $\widehat{\mathcal{P}}$ in G along which we can send the packets in a collision-free manner and with delivery time $\mathcal{T}$. We implicitly obtain the new set of paths and the collision-free schedule using a new technique that emulates a store-and-forward algorithm. There are three main steps in the emulation, which we describe next.

1. *Creation of store-and-forward network G'.* We transform the bufferless network G to a new store-and-forward network G'. Instead of separate outgoing edge-buffers, each node in G' has a unique *outgoing node-buffer* of size γ to store all the packets in transit. We divide the graph G into *regions* which are pairwise edge-disjoint connected components each consisting of about γ edges. Each node in G' corresponds to a region in G. (G' is also called the *region graph* of G; see Figure 1.)

The set of paths $\mathcal{P}$ in G is translated to set of paths $\mathcal{P}'$ in G'. If path $p \in \mathcal{P}$ uses an edge e in a region R of G, then in the respective path $p' \in \mathcal{P}'$, edge e is mapped to the node that represents R in G'. The final path p' is obtained by removing any cycles. We observe that the set of paths $\mathcal{P}'$ have node congestion $\overline{C}' = O(\gamma C)$, since at most γC paths of G are mapped to a single node in G'. Further, the dilation is $D' = O(D)$, since a path in G shrinks in G'.

2. *Store-and-forward scheduling in G'.* We execute a store-and-forward scheduling algorithm for the packets Π in G' using the paths in $\mathcal{P}'$. The scheduling algorithm has delivery time $\mathcal{T}' = O(C + D)$ and uses node-buffers of size $\gamma = O(\log(n + N))$. The algorithm is randomized and efficiently delivers the packets with high probability.

3. *Creation of set of paths $\widehat{\mathcal{P}}$ in G.* The store-and-forward schedule in G' is translated back to the original bufferless network G to give the set of paths $\widehat{\mathcal{P}}$ and a collision-free schedule for the packets. The translation is achieved implicitly with a deterministic bufferless emulation of the store-and-forward algorithm in G'. Each time step of the store-and-forward scheduling algorithm in G' is translated to a sequence of $O(\gamma^2 \cdot \log n)$ time steps in G. The main trick is to emulate the buffering. If in a time step a packet is buffered in a node in G', then the same packet in G circulates on an Euler tour of the edges in the corresponding region. Since a buffer in G' may hold multiple packets (but no more than γ), all those packets will circulate one after the other on the edges in the same region; recall that there are at least γ edges in each region. If in a time step a packet moves from one node to another node in G', then the packet moves from the first respective region to the next region in G. If a packet is injected (resp., absorbed) in G, the packet is injected (resp., absorbed) in the source (resp., destination) of the respective region.

During the emulation, the packets may appear to be deflected, while they are in fact circulating on the Euler tours in the region. By concatenating for each packet the respective Euler tours, we implicitly obtain the set of paths $\widehat{\mathcal{P}}$ on which the deflections in effect determine a collision-free schedule.

The cost of the emulation on the delivery time is a factor $O(\gamma^2 \cdot \log n)$ with respect to the store-and-forward schedule; that is, the resulting delivery time is $\mathcal{T} = O(\mathcal{T}' \cdot \gamma^2 \cdot \log n)$. This gives the desired delivery time of $\mathcal{T} = O((C+D) \cdot \log^3(n+N))$, which holds with high probability. We emphasize that the randomization is due to the store-and-forward algorithm, while the emulation is deterministic. Further, if the nodes know the graph G and the parameters C and N, then the resulting bufferless algorithm can be implemented in a distributed way.

1.6. Related work. There are no previously known results for *universal* bufferless packet-switching algorithms with near-optimal delivery time guarantees. However, there are efficient algorithms for specific bufferless models and architectures, which we summarize below.

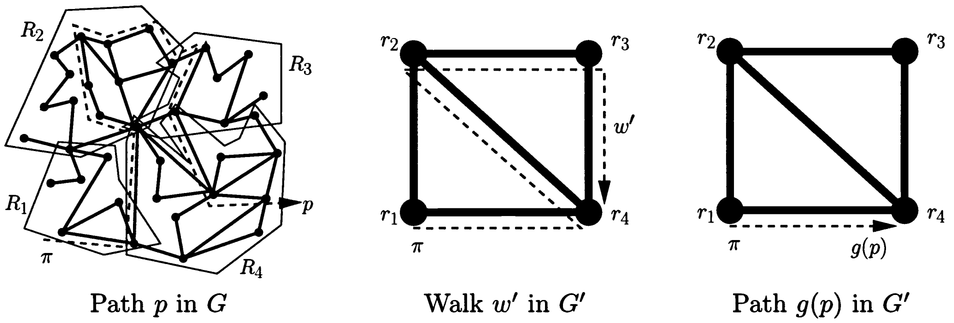


FIG. 1. *An example of the decomposition to the region graph.*

In *hot-potato* algorithms, packets are deflected to available links in a collision [8]. Our model of bufferless algorithms is based on the hot-potato model, with the significant exception that packets are deflected on particular available edges specified by the emulation, not on any available edge as is typical in hot-potato algorithms. This enables us to control the positions of the packets and implicitly obtain the aforementioned paths $\widehat{\mathcal{P}}$ and a collision-free schedule on them. Hot-potato algorithms have been extensively studied for a variety of architectures such as the mesh and torus [7, 9, 10, 14, 16, 18, 19, 20, 25, 26, 29, 30, 31, 39, 48], hypercubes [13, 15, 26, 28, 42], trees [23, 46], vertex-symmetric networks [36], and leveled networks [12, 17, 21]. Typically, by allowing packets to deviate slightly from their preselected paths, one obtains delivery times that are within polylogarithmic factors of optimal.

In *direct* (*collision-free*) packet scheduling, packets follow their paths without buffering and without any collisions, [4, 50, 22]. Busch et al. [22] give a comprehensive study of direct scheduling where they give a universal $O(C \cdot D)$ centralized algorithm, and near-optimal algorithms for the tree, mesh, butterfly, and hypercube.

Wormhole algorithms are similar to direct algorithms, although here, packets occupy more than one edge [24, 27]. Cypher et al. [24] give a randomized, universal distributed wormhole algorithm with delivery time $O(L \cdot C \cdot D)$, where L is the length of the packet; this bound can be improved if the edges have higher bandwidths. A dual to direct scheduling is *time-constrained* scheduling, where the task is to schedule as many packets as possible within a given time frame [1, 2]. In the related class of *matching* routing algorithms, packets are swapped at adjacent nodes. Under this model, permutation problems on trees have been studied in [3, 41, 51].

There are two variants of store-and-forward algorithms: those that use outgoing buffers on every edge (*edge-buffers*) and those that use a single outgoing buffer on every node (*node-buffers*). For nonbounded degree networks, these variants may not be equivalent, since one model does not necessarily translate to the other. The existence of universal store-and-forward scheduling algorithms with optimal delivery time $O(C + D)$ (plus additive logarithmic terms) and constant size edge-buffers was first established in the seminal work of Leighton, Maggs, and Rao [35]. Scheideler [47] showed that edge-buffers of size two are sufficient. These results are nonconstructive. Thereafter, the main focus has been on constructive algorithms with optimal delivery time $O(C+D)$ [11, 33, 37, 40, 43]. These algorithms use large buffers (proportional to the congestion C). Leighton, Maggs, and Rao [35] give a universal distributed algorithm that uses edge-buffers of size $O(\log ND)$ and has delivery time $O(C+D \log ND)$. Cypher et al. [24] give an algorithm with edge-buffers of size $O(\log CD)$ and slightly better delivery time. There are no better constructive results known for arbitrary networks that achieve smaller buffer sizes.

Our bufferless algorithm is based on emulating the universal distributed store-and-forward algorithm in Leighton, Maggs, and Rao [35]. Here, we analyze the delivery time of this store-and-forward algorithm in terms of the node congestion $\overline{C}$ and bound the node-buffer requirements, which is necessary for determining the performance of the bufferless emulation.

1.7. Paper outline.

In section 2, we give a graph decomposition into regions which is fundamental to our bufferless algorithm. Next, in section 3, we discuss store-and-forward algorithms and introduce a simple randomized algorithm using node-buffers. In section 4, we show how to emulate store-and-forward algorithms in a bufferless manner using the regions obtained by the graph decomposition in section 2. We apply the emulation on the randomized store-and-forward algorithm to obtain near-optimal universal bufferless algorithm in section 5. We finish with a discussion and future directions in section 6.

2. Regions.

The bufferless emulation uses regions in the graph to simulate buffering. To construct these regions, we will need to decompose the connected graph G into connected components of approximately a specified size.

2.1. Graph decomposition.

Let $G = (V, E)$ be a connected, unweighted, and undirected graph. Let F be a subset of the edges in E. The subgraph induced by F is the graph $H = (U, F)$, where U is the union of all vertices in V that are incident with edges in F. The edge set F is *connected* if the induced subgraph H is connected.

Definition 2.1. *A connected decomposition of G is a* partition *of the edges in E into a collection of disjoint sets $\{E_1, E_2, \ldots, E_k\}$ such that $\cup_{i=1}^{k} E_i = E$ and every E_i is connected.*

We refer to the E_i's as the *connected edge sets* or *regions* in the decomposition, and call the number of edges in E_i, $|E_i|$, the *size* of E_i. Notice that the subgraphs $H_1 = (V_1, E_1), \ldots, H_k = (V_k, E_k)$ induced by the edge sets may have overlapping vertex sets. We say that E_i is *connected* to E_j if and only if $V_i \cap V_j \neq \emptyset$. Notice that if E_i is connected to E_j, then $E_i \cup E_j$ is a connected edge set.

An $[\alpha, \beta]$*-partition* of G (if it exists) is a connected decomposition $\{E_1, \ldots, E_k\}$ of G, such that $\alpha \leq |E_i| \leq \beta$ for $i = 1, \ldots, k$. Notice that if $\alpha \approx \beta$, then an $[\alpha, \beta]$-partition decomposes G into connected edge sets of size approximately equal to α. Our first task is to show that such approximate decompositions exist for any connected graph. Our proof is constructive; hence, it can be converted to an algorithm.

The following lemma will be instrumental in the proof. Essentially, it states that a connected graph can be decomposed into two large connected edge sets.

Lemma 2.2. *Let $k \geq 2$. Any connected graph $G = (V, E)$ with $|E| \geq 3k - 2$ can be decomposed into two disjoint connected edge sets each of size at least k.*

Proof. Using a depth first search, determine a connected edge set F with $|F| = 2k - 2 \geq k$. Note that $|E \setminus F| \geq k$. $E \setminus F$ is composed of a number of connected edge sets (called *satellites*) $\alpha_1, \alpha_2, \alpha_{,3} \ldots$, each of which is not connected to any other, but all of which are connected to F. The situation is illustrated in Figure 2. Let α_1 be the largest such satellite edge set of F. If $|\alpha_1| \geq k$, then α_1 and $F \cup \alpha_2 \cup \alpha_3 \cup \cdots$ are both connected edge sets that have size $\geq k$; thus we are done. We need only consider the case where $|\alpha_1| \leq k - 1$. We will show how to replace $|F|$ with another edge set F' of the same size, and whose largest satellite α_1' will have size at least $|\alpha_1| + 1$. We can thus repeat this argument until the size of α_1 is at least k, concluding the proof.

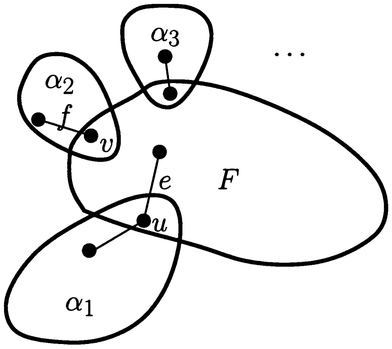


FIG. 2. *F and its satellites α_i.*

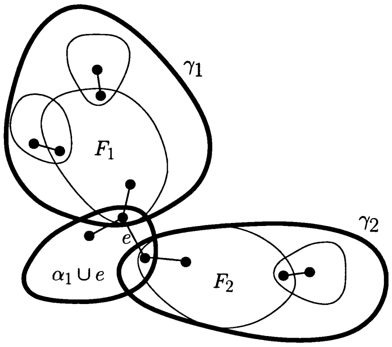


FIG. 3. *e is a bridge in F.*

We now show how to construct F'.

Suppose that $|\alpha_1| \leq k-1$, in which case there is at least one other satellite α_2. Let u and v be common nodes of α_1 and F and of α_2 and F, respectively. (Note that these nodes must exist, and they are different since F is connected to both these satellites.) Let e be an edge in F incident with u, and let f be an edge in α_2 incident with v (as shown in Figure 2). Increase the size of α_1 to $|\alpha_1|+1$ by adding e to it (and removing e from F). Note that α_1 remains connected. If $F \setminus e$ is a connected edge set, then add f to $F \setminus e$ to get F'. Note that $|F'| = |F|$. The edge set $\alpha_1 \cup e$ is now a connected edge set of size $|\alpha_1|+1$ which is part of the largest satellite of F' (note that while adding e to α_1, we may have connected α_1 to some other satellite). Thus the largest satellite α_1' of F' has size at least $|\alpha_1|+1$.

The only remaining case to consider is that including e into α_1 disconnects F; thus e is a bridge in F connecting two connected edge sets $F_1, F_2 \subset F$. The situation is illustrated in Figure 3, where we have merged F_1 and F_2 with their respective satellites to get edge sets γ_1 and γ_2 as illustrated. Note that since $|\alpha_1 \cup e| \leq k$, $|\gamma_1|+|\gamma_2| \geq 2k-2$. If neither $|\gamma_1| \geq k$ nor $|\gamma_2| \geq k$, this implies that $|\gamma_1| = |\gamma_2| = k-1$. In this case, merge γ_2 with e to form a connected edge set of size k. The remaining edges form a connected edge set of size at least $2k-2 \geq k$; hence we are done. Thus, suppose that one of γ_1 or γ_2 has size $\geq k$; without loss of generality, suppose that it is γ_1. Now consider $\alpha_1' = \gamma_2 \cup \alpha_1 \cup e$. There are two cases: $|\alpha_1'| \geq k$, and we are done; or $|\alpha_1| < |\alpha_1'| < k$, in which case $|\gamma_1| \geq 2k-1$, so that F_1 together with its satellites contains more than $2k-2$ edges. We construct F' from F_1 by adding edges from its satellites (while keeping it connected) until $|F'| = 2k-2$. To conclude, note that the largest satellite of F' must have size at least $|\alpha'_1| \geq |\alpha_1| + 1$. □

Using an induction argument and Lemma 2.2, we will show that there always exists an $[\alpha, \beta]$-partition with $\beta = \Theta(\alpha)$.

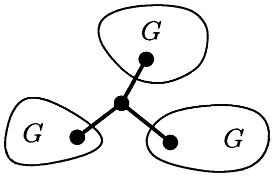
THEOREM 2.3 (existence of a $[k, 3k-3]$-partition). *Let $G = (V, E)$ be a connected graph. For any k, where $1 < k \leq |E|$, there exists a $[k, 3k-3]$-partition of G.*

Proof. If $2 \leq k \leq |E| \leq 3k-3$, then E itself is a $[k, 3k-3]$-partition and thus there is nothing to prove. We will now prove the claim by strong induction on $|E|$. The induction hypothesis is

$\mathbf{P}(N)$: There exists a $[k, 3k-3]$-partition for any $G = (V, E)$ whenever $|E| \in [k, N]$.

We claim that $\mathbf{P}(N)$ is true for all N. We know that $\mathbf{P}(3k-3)$ is true, so suppose that $\mathbf{P}(N)$ is true for some $N \geq 3k-3$ and consider $\mathbf{P}(N+1)$. Let $G = (V, E)$ be any graph with $|E| = N+1$. Since $|E| \geq 3k-2$, Lemma 2.2 implies that E can be decomposed into two disjoint connected edge sets E_1, E_2 with $k \leq |E_1| \leq |E_2| \leq N$. By the induction hypothesis, there exist $[k, 3k-3]$-partitions of E_1 and E_2. The union of these two partitions is a $[k, 3k-3]$-partition of E, concluding the proof. □

The following example proves that the result of Theorem 2.3 is tight. For a given k, let G be any connected graph with $k-2$ edges, and connect three such graphs in a wheel configuration as shown below. It is easy to see that the only decomposition in which every edge set has at least k edges is the entire graph itself, which has $3k-3$ edges.



The proof in Theorem 2.3 is constructive, based upon the construction in Lemma 2.2. In order to analyze the run time more easily, we convert the construction into the algorithmic format in Algorithm 1. We use the same notation that was used in the proof of Theorem 2.3. The depth first search and the computation of the satellites take time $O(E)$. The while loop executes at most k times, since $|\alpha_1|$ strictly increases in each execution (else the function calls itself and returns). In each execution, at most $O(E)$ work is done, and get_components can possibly be called on two smaller instances, both corresponding to graphs of size $\geq k$. Thus, letting $T(|E|, k)$ denote the worst-case run time to obtain a $[k, 3k-3]$-partition for a graph with size $|E|$, we have that for some constant c,

$$T(|E|, k) \leq \max_{3k-3 \leq b \leq |E|-3k+3} \{T(b, k) + T(|E|-b, k)\} + ck|E|,$$

with $T(|E|, k) = 1$ for $|E| \leq 3k-3$. One can show by induction that $T(k, |E|) \leq \frac{3}{2}c|E|^2 = O(|E|^2)$, and hence the algorithm to compute the decomposition is polynomial in $|E|$.

2.2. The region graph. Consider a connected graph $G = (V, E)$, with n nodes. Take an $[\alpha, \beta]$-partition of G, which gives regions $R_1, R_2, \ldots, R_k$. Let the subgraphs induced by these regions have vertex sets $U_1, U_2, \ldots, U_k$. The *region graph* $G' =$

Algorithm 1 get_components(E, k)

```
// Returns a $[k, 3k-3]$-partition for the edge set $E$; Assume $|E| \geq k$;
if $|E| \leq 3k-3$ then
  return $E$;
Using DFS, compute $F \subset E$ and all its satellites; let $\alpha_1$ be its largest satellite;
while $|\alpha_1| < k$ do
  Choose an edge $e \in F$ that is incident with a vertex in the subgraph induced by $\alpha_1$;
  Choose an edge $f$ in a satellite $\alpha_2 \neq \alpha_1$ which is incident to a node induced by $F$;
  if $F \setminus e$ is a connected edge set then
    $\alpha_1 \leftarrow \alpha_1 \cup e$; $F \leftarrow (F \setminus e) \cup f$;
  else if $F \setminus e$ is disconnected and $|\gamma_1| = |\gamma_2| = k-1$ then
    Label $\gamma_1, \gamma_2$ so that $\alpha_1$ is connected to $\gamma_1$;
    return $\{\gamma_2 \cup e\} \cup$ get_components$(\alpha_1 \cup \gamma_1, k)$;
  else ($F \setminus e$ is disconnected into $\gamma_1, \gamma_2$ which are labeled so that $|\gamma_1| \geq k$)
    if $|\alpha_1 \cup \gamma_2| \geq k$ then
      return get_components$(\gamma_1, k) \cup$ get_components$(\alpha_1 \cup \gamma_2, k)$;
    else
      $\alpha_1 \leftarrow \alpha_1 \cup \gamma_2$;
      $F \leftarrow$ connected subset of $\gamma_1$ of size $2k-2$ that is adjacent to $\alpha_1$;
```

(V', E') has a vertex set $V' = \{r_1, r_2, \ldots, r_k\}$, where each vertex r_i corresponds to the region R_i of G. Two vertices r_i, r_j are adjacent in G' (that is, the edge (r_i, r_j) is in E') if and only if $U_i \cap U_j \neq \emptyset$ (that is, the corresponding regions are connected). An example of a region graph is given in Figure 1. Since each region consists of at least α and at most β edges, we immediately have that $|E|/\beta \leq |V'| \leq |E|/\alpha$. We proceed to show that G' is connected.

LEMMA 2.4. *Graph G' is connected.*

Proof. Let r_i, r_j be two nodes in V' corresponding to regions R_i, R_j in G. We show that there is a path in G' from r_i to r_j. If R_i and R_j share a node, then r_i and r_j are adjacent. Otherwise, let $e_i = (u_i, v_i) \in R_i$ and $e_j = (u_j, v_j) \in R_j$ be two edges in E. Since G is connected, there is a path in G from v_i to u_j. This path consists of edges $e_1, e_2, \ldots, e_k$ in regions $R_1, R_2, \ldots, R_k$, respectively. (Note that consecutive regions are not necessarily distinct.) Now consider the path $e_i, e_1, e_2, \ldots, e_k, e_j$ and the regions $R_i, R_1, R_2, \ldots, R_k, R_j$; since every two consecutive edges share a node, every two consecutive regions in this list are connected, which implies the existence of a walk from r_i to r_j in G'. Since r_i and r_j are arbitrary nodes, it follows that G' is connected. □

2.3. Paths on region graph.

Let $\mathcal{P}$ denote a set of paths on G with edge congestion C, node congestion $\overline{C}$, and dilation D. Let $\{R_1, \ldots, R_k\}$ be an $[\alpha, \beta]$-partition of G into regions. Every edge in G belongs to exactly one region. Let $G' = (V', E')$ be the corresponding region graph. We define a mapping $f : E \to V'$ from the edges of G to the nodes of G' as follows:

For every $e \in E$, $f(e) = r_i$ if and only if $e \in R_i$.

Consider a path $p \in \mathcal{P}$, with $p = (e_1, e_2, \ldots, e_l)$. We define a mapping g which maps a path in G to a path in G' as follows:

For any path $p = (e_1, e_2, \ldots, e_l)$ in G, consider the walk in G' given by $w' = (f(e_1), f(e_2), \ldots, f(e_l))$. The path $g(p)$ denotes the walk w' that we obtain after removing all the cycles and repeated nodes in w', $g(p) = (f(e_{i_1}), f(e_{i_2}), \ldots, f(e_{i_k}))$ (see Figure 1).

We now transform the set of paths $\mathcal{P}$ of the original graph G into a set of paths $\mathcal{P}'$ on the region graph G' as follows:

$\mathcal{P}' = \{p'_1, p'_2, \ldots, p'_N\}$, where $p'_i = g(p_i)$ for every path $p_i \in \mathcal{P}$.

Let C', $\overline{C'}$, and D' denote the edge congestion, the node congestion, and the dilation of the paths in $\mathcal{P}'$, respectively. For any set of paths, the edge congestion is trivially bounded by the node congestion; hence $C' \leq \overline{C}'$. A path uses node r_i only if it contains edges in R_i. By construction, $|R_i| \leq \beta$, so the number of edges in $\mathcal{P}$ that use R_i is at most βC; thus, $\overline{C'} \leq \beta C$. Since $|g(p)| \leq |p|$ for any path p in G, we immediately have the following lemma.

LEMMA 2.5 (congestion and dilation in the region graph). *$C' \leq \overline{C'} \leq \beta C$; $D' \leq D$.*

2.4. Euler cycles in regions. Given an undirected graph $G = (V, E)$, we define the *directed representation* of G to be the graph $G^{dir} = (V, E^{dir})$, where each (undirected) edge $(u, v) \in E$ is replaced by two directed edges $(u, v), (v, u) \in E^{dir}$. Consider a graph decomposition of G into regions $R_1, R_2, \ldots, R_k$. We associate with each R_i a region R_i^{dir} in G^{dir}, where each edge in R_i is replaced by the two respective directed edges in R_i^{dir}. Take any node v induced by region R_i. Since every edge in R_i is replaced by two edges in opposite directions in R_i^{dir}, the in-degree of v is equal to its out-degree in R_i^{dir}.

An *Euler cycle* in a region R_i^{dir} is an edge-simple cycle that contains all the edges of R_i^{dir}. Since in R_i^{dir} the in-degree equals the out-degree of every node, R_i^{dir} has an Euler cycle. Let ψ_i denote an Euler cycle in R_i^{dir}. Let $\psi_i = (v_1, v_2, \ldots, v_1)$ be the sequence of nodes that ψ_i visits in R_i^{dir}. Since G and G^{dir} have the same set of nodes, ψ_i is mapped to a walk in R_i, when we follow the same sequence of nodes as in R_i^{dir}. We will refer to ψ_i as the *Euler cycle* of R_i as well.[1] In R_i, ψ_i will traverse the same edge twice, since the edge is traversed in two opposite directions in R_i^{dir}. Thus, in an $[\alpha, \beta]$-partition of G, every Euler cycle ψ_i satisfies $2\alpha \leq |\psi_i| \leq 2\beta$ (since $\alpha \leq |R_i| \leq \beta$).

3. Store-and-forward scheduling in G'. In graph G, the bufferless algorithm will emulate a store-and-forward algorithm which is applied in region graph G'. Here, we discuss the specifications for the store-and-forward algorithm. Let A denote such a store-and-forward algorithm. We will define the specifications of A. We will then give an instantiation of such an algorithm below (Algorithm A_1).

3.1. Specification of Algorithm A. Consider the batch problem $Q = (G, \Pi, \mathcal{S})$ and a set of paths $\mathcal{P}$ that satisfy Q. Let $\mathcal{P}'$ be the respective set of paths in G' (recall section 2.3). The objective of the store-and-forward Algorithm A is to solve a *packet scheduling problem* $Q' = (G', \Pi, \mathcal{P}')$ in graph G', where the task is to send the packets Π along their respective set of paths $\mathcal{P}'$ in G'. Let γ denote the size of the *node-buffers* that Algorithm A uses. Specifically, each node has a node-buffer of size γ. A packet scheduling is *valid* whenever packets are not dropped (there are no buffer overflows). During any single time step in Algorithm A, a packet may perform one of four actions:

[1] This is clearly an abuse of notation, since ψ_i is an Euler cycle of R_i^{dir}, not of R_i.

Algorithm 2 Store-and-Forward Algorithm A_1

```
Divide time into phases of length $\gamma = 6\log(n' + 2N)$ time steps.
for each packet $\pi$ do
    $\pi$ will be injected at a phase $\phi_\pi$, where $\phi_\pi$ is chosen uniformly and at random between 1 and $12\overline{C'}/\gamma$;
    Packet $\pi$ is injected at the first time step of phase $\phi_\pi$;
    Packet $\pi$ follows its path traversing one edge per phase;
```

(i) be injected into the network at its source node; [**Injection**]
(ii) move from its current node to a neighboring node; [**Transfer**]
(iii) move to and be absorbed in its destination node; [**Absorbtion**]
(iv) remain in the buffer of its current node. [**Buffering**]

If a packet is received into a node's buffer at time t or was already buffered there during time step t, then at time $t+1$, it must either be transmitted along the next edge in its path or remain stored in the buffer. It is possible to divide any valid scheduling into a sequence of *phases*, so that each phase has the following three properties:

(i) Each phase is a time interval consisting of at least one time step.
(ii) During a phase, each packet traverses at most one edge in G'.
(iii) During a phase, a node receives at most γ packets (by transfer or injection).

Suppose that Algorithm A produces a valid schedule. A trivial division of the execution of Algorithm A into phases that satisfies these three properties is to take each phase to be a single time step, since at every time step, any valid execution of Algorithm A must satisfy these three properties. (Property (iii) is satisfied for any valid scheduling because the size of the buffer is γ.) As we will show later, any candidate store-and-forward algorithm which produces valid schedules and satisfies these three properties may be used in our bufferless emulation algorithm to create a universal bufferless algorithm. In this case, we will say that the store-and-forward algorithm is *emulatable.* We now give a simple, randomized emulatable store-and-forward algorithm, which with high probability gives a valid schedule that satisfies the aforementioned three properties.

3.2. Store-and-forward Algorithm A_1.

We now give an instantiation of a universal, emulatable store-and-forward algorithm, which is actually the universal distributed algorithm of Leighton, Maggs, and Rao [35]. Here, we analyze the node-buffer requirements of the algorithm and bound its delivery time with respect to the node congestion, while originally in [35] the algorithm is analyzed with respect to edge-buffers and edge-congestion. The algorithm is randomized, uses node-buffers whose size γ is logarithmic with respect to the parameters of the scheduling problem, and has near-optimal delivery time. We refer to this algorithm as Algorithm A_1.

Let $Q' = (G', \Pi, \mathcal{P}')$ be a scheduling problem with path set $\mathcal{P}'$ on an arbitrary graph $G' = (V', E')$. Let $\overline{C'}$ be the node congestion and D' the dilation for the paths in $\mathcal{P}'$. Let N be the number of packets and $n' = |V'|$.

Algorithm 2 contains the details of the store-and-forward Algorithm A_1. The intuition behind this algorithm is that at most γ packets will be stored during a phase in any node. The packets will leave the node by the end of the phase. This is feasible, since the phase consists of γ time steps. However, we need to be a little careful to ensure that all γ packets that are leaving a node will find buffer space to enter their destination node. The detailed analysis of the algorithm follows below.

We will show that with high probability, Algorithm A_1 successfully delivers the packets and at the same time is emulatable. For Algorithm A_1 and phase ϕ, we define the following properties:

$P1(\phi)$: In phase ϕ every packet in the network successfully traverses one edge in its path.

$P2(\phi)$: No more than γ packets are buffered at any node during phase ϕ.

$P3(\phi)$: No more than $\gamma/2$ packets arrive at any node during phase ϕ.

$P4(\phi)$: No more than $\gamma/2$ packets remain at any node at the end of phase ϕ.

Note that $P3$ is stronger than we need. We introduce property $P4$ for technical convenience. Since the maximum injection phase is $12\overline{C'}/\gamma$ and the maximum path length is D', we have the following lemma.

Lemma 3.1. *If $P1$–$P4$ hold for $12\overline{C'}/\gamma+D'$ phases, then $\Phi_{A_1}(Q') \le 12\overline{C'}/\gamma+D'$, and Algorithm A_1 is a valid algorithm for bufferless emulation (emulatable).*

Let $\Pr[\phi_0]$ be the probability that properties $P1$–$P4$ hold for all phases $\phi \le \phi_0$. $\Pr[0] = 1$ by default. We now give a lower bound for $\Pr[\phi_0 + 1]$ in terms of $\Pr[\phi_0]$.

Lemma 3.2. *If $P1$–$P4(\phi_0)$ are true and $P3(\phi_0+1)$ is true, then $P1$–$P4(\phi_0+1)$ are true.*

Proof. If no more than $\gamma/2$ packets arrive at a node during phase $\phi_0 + 1$, then since $P4(\phi_0)$ is true, there are at most γ packets in the node during any time step of phase $\phi_0 + 1$; therefore $P2(\phi_0 + 1)$ is true. In the worst case all of the at most $\gamma/2$ packets in the node at the end of phase ϕ_0 may leave sequentially on a single edge, requiring at most $\gamma/2$ time steps, which is less than the duration of the phase; thus $P1(\phi_0+1)$ is true. The packets remaining in the node at the end of phase ϕ_0+1 are only those that entered, which is at most $\gamma/2$ packets; thus $P4(\phi_0 + 1)$ is true. □

By induction, we obtain the following corollary.

Corollary 3.3. *$P1$–$P4(\phi)$ are true for all $\phi \le \phi_0$ if and only if* P3(ϕ) *is true for all $\phi \le \phi_0$.*

Thus, $\Pr[\phi_0 + 1] = \Pr[\{P1\text{–}P4(\phi) \text{ are true for } \phi \le \phi_0\} \wedge \{P3(\phi_0 + 1) \text{ is true}\}]$. Noting that $\Pr[A \wedge B] = 1 - \Pr[\sim A \vee \sim B] \ge 1 - \Pr[\sim A] - \Pr[\sim B]$,

$$\Pr[\phi_0 + 1] \ge \Pr[\phi_0] - \Pr[\{P3(\phi_0 + 1) \text{ is false}\}]. \tag{3.1}$$

Consider a node v and phase $\phi_0 + 1$. Let q_π be the probability that packet π arrives at node v during phase $\phi_0 + 1$, which can happen only if it is injected at a particular phase. Since the probability that it is injected at that particular phase is $\gamma/12\overline{C'}$, we conclude that $q_\pi \le \gamma/12\overline{C'}$ if π uses node v (at most $\overline{C'}$ such packets), and 0 otherwise. Let $X_i(v) = 1$ if packet π_i appears at node v at phase $\phi_0 + 1$. $X_i(v)$ are independent random variables, whose sum is the number of packets that appear at node v at phase $\phi_0 + 1$. Let $X(v) = \sum_i X_i(v)$. $\mathbf{E}[X(v)] = \sum_i q_{\pi_i} \le \overline{C'} \cdot \gamma/12\overline{C'} = \gamma/12$. By applying a version of the Chernoff bound [38, Exercise 4.1], we obtain

$$\Pr[X(v) > \gamma/2] < 2^{-\gamma/2}.$$

Applying the union bound now gives that $\Pr[\max_v X(v) > \gamma/2] < n'2^{-\gamma/2}$, giving the following lemma.

Lemma 3.4. $\Pr[\{P3(\phi_0 + 1)$ *is false*$\}] < n'2^{-\gamma/2}$.

Using (3.1), Lemma 3.4, and the fact that $\Pr[0] = 1$, we get the following result by induction.

Lemma 3.5. $\Pr[\phi_0] \ge 1 - \phi_0 n' 2^{-\gamma/2}$.

Since $n' < n' + 2N$, $12\overline{C'}/\gamma + D' < n' + 2N$ (because $\overline{C'} \le N$ and $D' \le n'$), and $2^{-\gamma/2} = (n' + 2N)^{-3}$, by setting $\phi_0 = \Phi_{A_1}(Q')$ in Lemma 3.5 and using Lemma 3.1, we obtain the main result of this section.

THEOREM 3.6 (delivery time of Algorithm A_1). *With probability at least* $1 - 1/(n' + 2N)$*, Algorithm* A_1 *solves scheduling problem* Q' *in at most* $12\overline{C'}/\gamma + D'$ *phases, satisfying* $P1$*–*$P4$ *in each phase (Algorithm* A_1 *is emulatable). The node-buffer size required is* $\gamma = 6\log(n' + 2N)$*.*

4. Bufferless emulation in G.

Let $G = (V, E)$ be a connected graph with n nodes and let $\{R_1, \ldots, R_k\}$ be an $[\alpha, \beta]$-partition of G with corresponding region graph $G' = (V', E')$. Consider batch problem $Q = (G, \Pi, \mathcal{S})$ in G. Let $\mathcal{P}$ be a set of paths that satisfy Q. Let the corresponding path scheduling problem in G' be $Q' = (G', \Pi, \mathcal{P}')$.

Our goal is to design a bufferless algorithm to solve the batch problem Q, with delivery time $\widetilde{O}(C + D)$. In other words, the goal is to construct a set of paths $\widehat{\mathcal{P}}$ for which a bufferless and collision-free schedule exists with delivery time $\widetilde{O}(C + D)$. We implicitly obtain this set of paths through emulation of a store-and-forward algorithm. The store-and-forward algorithm solves the scheduling problem Q' in G' with set of paths $\mathcal{P}'$; the set of paths $\mathcal{P}'$ was derived from $\mathcal{P}$. In solving Q in G, the bufferless algorithm will *emulate* Algorithm A in G' (in a step-by-step fashion). The set of paths $\widehat{\mathcal{P}}$ derived by the bufferless algorithm will depend on the set of paths in $\mathcal{P}'$, and hence also on the set of paths in $\mathcal{P}$. Thus, the preselected paths in $\mathcal{P}$ are crucial to the functioning of the bufferless algorithm, even though the paths used may deviate significantly from the preselected paths.

4.1. The emulation.

Assume that we have already constructed an emulatable, store-and-forward *Algorithm A* which solves the region graph scheduling problem $Q' = (G', \Pi, \mathcal{P}')$ using a buffer of size γ. Assume that $2\gamma \leq |E|$ (we will deal with this assumption later in section 5). We now discuss how to obtain a bufferless *Algorithm B* which will emulate the phases of Algorithm A, which may possibly be faster than emulating the individual time steps of Algorithm A. During a single phase of Algorithm A, a packet π performs one of four actions (in G'): injection, transfer, absorbtion, or buffering. Algorithm B emulates Algorithm A phase for phase by emulating each of these actions that a packet can make. We continue with an informal description of Algorithm B.

Algorithm B emulates the buffering of packets and their transfer from node to node using the $[\alpha, \beta]$-partition of G, where we set $\alpha = 2\gamma$ (by assumption, $\alpha = 2\gamma \leq |E|$). By Theorem 2.3, we guarantee the existence of such an $[\alpha, \beta]$-partition by choosing $\beta = 6\gamma - 3$. Recall that we refer to nodes in the region graph by r_i and their corresponding region in the original graph by R_i.

- When in Algorithm A a packet is buffered in a node r_i of G', Algorithm B emulates this by letting the packet circulate in the edges of region R_i in G.
- When in Algorithm A a packet is transferred from node r_i to node r_j of G', in Algorithm B the packet is transferred from region R_i to region R_j in G. If the packet is absorbed by r_j, it will be absorbed by the appropriate node in R_j.
- If a packet is injected into the buffer of a node r_i by Algorithm A, then in Algorithm B, it will be injected into its injection node in R_i. From then on, it will continue to circulate in the region until it is transferred to the next region.

We now describe the details of the emulation.

4.1.1. Phases and rounds.

Let Φ denote the number of phases Algorithm A uses to deliver the packets in the region graph. In Algorithm B, time is divided into

Φ phases. Each phase of Algorithm B emulates a phase of Algorithm A. In order to perform the emulation of a phase, Algorithm B further divides each phase into Ξ *rounds*, where Ξ will be specified later. The duration of each round is $Z = 4\beta^2 + 4\beta$ time steps. (Recall that $\beta = 6\gamma - 3$.) Thus, the bufferless algorithm runs for $\Phi \cdot \Xi \cdot Z$ time steps in total.

For the duration of an entire round, a region is either in the *sending* or the *receiving* mode—we say that the region is sending or receiving. In the emulation, when a packet has to be transferred from region R_i to the region R_j, R_i must be sending and R_j receiving. We will show how to guarantee that for any pair of adjacent nodes $r_i, r_j \in V'$, there is a round in every phase in which region R_i is sending and R_j is receiving (and vice versa).

In order to determine if a region is sending or receiving, we first obtain a vertex coloring of G'. Let $\boldsymbol{\chi} : V' \mapsto [0, n']$ be a *valid* vertex coloring of G', where $\boldsymbol{\chi}(r)$ is the color assigned to node $r \in V'$ and no two nodes have the same color. Let $\chi = \max_i \boldsymbol{\chi}(r_i)$ denote the maximum color used in the vertex coloring. A valid coloring can be obtained by a simple greedy algorithm where the maximum color is bounded by the maximum node degree. Since the maximum node degree is bounded by n', where $n' = |V'|$, we have that $\chi \leq n'$.

We define the color of a region R_i as the color assigned to the corresponding node r_i. Let $\boldsymbol{\delta}_i$ be the binary representation of $\boldsymbol{\chi}(r_i)$. Let σ denote the number of bits in χ, $\sigma = \lceil \log \chi \rceil \leq \lceil \log n' \rceil$. By prepadding with zeros, we assume that every color $\boldsymbol{\delta}_i$ has σ bits. We define the *mode parameter* $\mathbf{x}_i$ for region R_i to be the 2σ long binary vector $\bar{\boldsymbol{\delta}}_i \boldsymbol{\delta}_i$, where $\bar{\boldsymbol{\delta}}_i$ is the binary complement of $\boldsymbol{\delta}_i$. For $1 \leq k \leq 2\sigma$, we denote the kth bit of $\mathbf{x}_i$ by $\mathbf{x}_i(k)$.

We set the number of rounds in a phase to be $\Xi = 2\sigma \leq 2\lceil \log n' \rceil$; thus, each phase in Algorithm B consists of the 2σ rounds, $\omega_1, \omega_2, \ldots, \omega_{2\sigma}$. During round ω_k, if $\mathbf{x}_i(k) = 0$, then region R_i is sending; otherwise, if $\mathbf{x}_i(k) = 1$, then region R_i is receiving. Our assignment of colors ensures that during every phase, any region R_i may send a packet to any neighboring region R_j (i.e., R_i will be sending and R_j receiving), and similarly it may receive a packet from any neighboring region R_j (i.e., R_j will be sending and R_i receiving).

LEMMA 4.1. *If R_i and R_j are adjacent, then during every phase, there is at least one round ω_s (ω_r) in which R_i is sending (receiving)* and *R_j is receiving (sending).*

Proof. Since $\boldsymbol{\chi}$ is a valid coloring, and R_i and R_j are adjacent, $\boldsymbol{\delta}_i$ and $\boldsymbol{\delta}_j$ must differ at some bit. Suppose they differ in the kth bit, $1 \leq k \leq \sigma$. Thus, rounds k and $k + \sigma$ satisfy the requirements, since $\overline{\mathbf{x}_i(k+\sigma)} = \mathbf{x}_i(k) = \overline{\mathbf{x}_j(k)} = \mathbf{x}_j(k+\sigma)$. □

The fundamental operation that is needed for the emulation by Algorithm B is packet circulation within a region.

4.1.2. Packet circulation. Packet circulation is a basic function for the emulation. During packet circulation, a packet π repeatedly follows the Euler cycle ψ_i of the region R_i that it is in: At each time step, packet π follows the next edge in ψ_i; when π reaches the end of the Euler cycle it continues from the beginning of the cycle, and so on. At the time step in which packet π traverses an edge $e \in \psi_i$, we say that e is the *current* edge of π.

At each round of a phase, a region is either sending or receiving. The speed at which a packet circulates in its region depends on whether the region is sending or receiving:

- If the region is receiving, then the packet follows the Euler cycle in the normal fashion (one link per time step).

- If the region is sending, then the packet moves at an effectively slower speed as follows. At time step 0 (the beginning of the round), suppose that π is at node u with current edge $e = (u, v) \in \psi_i$. At time step 0, packet π follows its current edge (u, v) and at time step 1, π appears in node v. At time step 1, suppose that its new current edge in ψ_i is (v, w); the packet *does not* follow its new current edge in ψ_i, but instead it follows edge (v, u) from v back to u, and thus at time step 2, it appears back in node u. Thus after two time steps, the packet has effectively not moved. We call such an operation an *oscillation*, and we say that packet π oscillates on its current edge in the Euler cycle. The time period of the oscillation is two time steps, The packet continues in this fashion for subsequent time steps, so at even time steps $t = 2i$, it appears in node u, and at odd time steps $t = 2i + 1$ it appears in node v, for $i \geq 0$. The packet performs β such oscillations on its current edge e; thus, after 2β time steps, the packet appears at u and follows edge e for the last time. At time step $T_s = 2\beta + 1$, the packet is now at v, and at this point it stops oscillating on edge e and begins oscillating on its new current edge $(v, w) \in \psi_i$. Thus, after T_s time steps, the packet advances by one edge in the Euler cycle of ψ_i. Consequently, since $|\psi_i| \leq 2\beta$, after $2\beta T_s = 4\beta^2 + 2\beta$ time steps, a packet circulating in region R_i has oscillated at least once on every edge of ψ_i.

From the above description of the packet movement in a sending region, we obtain the following lemma.

LEMMA 4.2. *After $4\beta^2 + 2\beta < Z$ time steps, a packet circulating in a sending region R_i has oscillated at least once on every edge in ψ_i.*

Suppose that the directed edge $e = (u, v) \in \psi_i$ is an edge in the Euler cycle of a receiving region R_i. If at time step t, no packet has edge e as its current edge, then we say that e is *empty*. At each time step, we say that an empty edge is associated with an *empty slot*. Empty slots are similar to packets in that they too circulate—as the packets in a receiving region circulate (forwards) in ψ_i, the empty slots circulate (backwards) in ψ_i at the same rate. They continue to circulate until some packet occupies the empty edge.

Once a packet enters the network, its default status is to be circulating in the region it is in. Packets enter a region either through injection or packet transfer. We discuss how these steps are emulated by bufferless Algorithm B. In particular, whenever a packet enters a region, it must not interfere with packets that are already circulating in the region.

4.1.3. Emulation of injection. Suppose that in phase ϕ, Algorithm A injects packet π into node r_i. Let p be the path of π in G, e the first edge in this path, and u the injection node. Since Algorithm A injects the packet into node r_i of G', we know that e and u are in R_i. Algorithm B will inject π into R_i in phase ϕ during the last round of the phase in which R_i is receiving. Algorithm B will inject π into an empty edge in the Euler cycle ψ_i, i.e., one which is not a current edge of any packet circulating in R_i. In this way, we guarantee that the injection of π will not interfere with any currently circulating packets. After injection, π will continue to circulate in R_i until the end of phase ϕ. All that remains is to show that it is possible to inject π so that it does not interfere with any circulating packets. We will say that a region is *ready for phase ϕ* of bufferless Algorithm B if at the beginning of phase ϕ (end of phase $\phi - 1$) there are at most γ packets circulating in the region. By default, all regions are ready for the first phase (since there are no packets in the network).

Remember that $2\alpha \leq |\psi_i| \leq 2\beta$, so $4\gamma \leq |\psi_i|$. Suppose that region R_i is ready for phase ϕ of Algorithm B. Then, by definition of a phase in Algorithm A, at most γ packets will need to enter region R_i during phase ϕ; i.e., at most $\gamma - 1$ packets other than π need to enter. This means that only at most $2\gamma - 1$ of the edges in ψ are current edges of packets; the remaining $|\psi_i| - (2\gamma - 1) \geq 2\alpha - (2\gamma - 1) = 2\gamma + 1$ edges are empty slots that continually circulate backwards one edge at a time during the round. An empty time slot is *available* for π if it will not be occupied by any packet other than π during its backward circulation. Thus, there are at least $2\gamma + 1$ available empty slots for π in region R_i. For any given edge e, each of these available empty slots will be at e once every $|\psi_i| \leq 2\beta < Z$ time steps.

Let $e = (u, v) \in \psi_i$ be the first edge of packet π in R_i. Packet π is injected into the network, and e becomes its current edge at the first time step when e becomes empty. Since π is injected into an empty edge, it does not interfere with any packets already circulating in R_i. There are at least $2\gamma + 1$ available empty slots, so we know that e becomes empty at least $2\gamma + 1$ times during the round. Thus, even if all (at most γ) packets entering R_i during this phase are through injection at node u with the *same* first edge $e = (u, v)$ in their path, all of these packets can be injected into circulation in R_i in just this one round.

Lemma 4.3. *Suppose that packet π with first edge $e \in R_i$ is injected into node r_i during phase ϕ of Algorithm A. If R_i is ready for phase ϕ of Algorithm B, then packet π can be injected into R_i without interfering with any already circulating packets.*

4.1.4. Emulation of packet transfer. Suppose that in phase ϕ of Algorithm A, packet π moves from node r_i to node r_j. Assume that at the beginning of phase ϕ in Algorithm B, packet π is circulating in region R_i, and that region R_j is ready for phase ϕ of Algorithm B. During phase ϕ in Algorithm B, π will move from R_i to R_j as follows. Packet π will circulate in R_i until the first round ω of phase ϕ in which R_i is sending and R_j is receiving. (The existence of such a round is guaranteed by Lemma 4.1.)

Since r_i and r_j are adjacent in G', there exists a node u which is common to R_i and R_j. Since node u is in R_i, there exists an edge $e_i = (u_i, u) \in \psi_i$ on the Euler cycle of R_i. Similarly, there exists an edge $e_j = (u, u_j) \in \psi_j$ on the Euler cycle of R_j. During round ω, packet π circulates (in slow mode) in region R_i along the Euler cycle ψ_i. At some particular slow time step τ of the round, the current edge of π will be e_i. During the next $T_s = 2\beta + 1$ time steps, π oscillates on edge e_i and will appear at the common node u at the $\beta + 1$ times $\tau + 1, \tau + 3, \ldots, \tau + 2\beta + 1$. If at any of these times, the edge $e_j \in \psi_j$ is an empty slot, i.e., not the current edge of any packet circulating (in normal mode) in R_j, then π switches from oscillation on edge e_i, making e_j its new current edge. Packet π now continues to circulate in R_j at normal speed. Since π enters R_j on an empty edge, it will not interfere with any packets already circulating in R_j. Note that π will have completed a full circuit on its Euler path ψ_i in at most $4\beta^2 + 2\beta$ time steps; thus, it will have completed its oscillations on edge e_i within the first $4\beta^2 + 2\beta$ time steps of the round. Thus, π will enter R_j within the first $4\beta^2 + 2\beta$ time steps of round ω, provided that it found an empty edge on which to enter.

We now show that during round ω, π will indeed find an empty edge on which to move into R_j. Specifically, for at least one of the time steps $\tau + 1, \tau + 3, \ldots, \tau + 2\beta + 1$, the edge $e_j \in \psi_j$ will be an empty slot. Remember that empty slots circulate backwards in R_j at the rate of one edge per time step. Thus, *every* available empty slot will pass e_j at least once during *any* consecutive 2β time steps. By the arguments in section 4.1.3, we know that there are at least $2\gamma + 1$ such available empty slots. Therefore, edge e_j will become an empty slot at least once in the $2\beta + 1$ consecutive time steps $\tau+1, \tau+2, \tau+3, \dots, \tau+2\beta+1$. However, due to the nature of packet π's oscillation on edge e_i, packet π will only be able to use an empty slot if the slot passes e_j at time step $\tau+k$ for some *odd* $k \in [1, 2\beta+1]$. To show that such a situation is guaranteed to occur, we need to show that there is at least one pair of *consecutive* available empty slots.

LEMMA 4.4. *Suppose R_j is ready for phase ϕ of Algorithm B. There is at least one pair of consecutive empty slots that is available for π.*

Proof. Say that a slot is *booked* if it is not available for π. Let $\Gamma \leq 2\gamma - 1$ be the number of booked slots. The number of available empty slots is $|\psi_j| - \Gamma$. Since $|\psi_j| \geq 4\gamma$, we have that $\Gamma < |\psi_j|/2$. Suppose there is no pair of consecutive available slots for π. For every available slot, the next slot must therefore be booked; hence the number of available slots is at most $|\psi_j|/2$. Thus, $|\psi_j|$, the total number of slots (booked plus available), is at most $\Gamma + |\psi_j|/2 < |\psi_j|$, a contradiction. □

Let the two consecutive available slots implied by Lemma 4.4 be c_1 and c_2. Suppose that c_1 passes e_j first at time step $\tau + k_1$ for some $k_1 \in [0, 2\beta - 1]$. If k_1 is odd, then this empty slot can be used by π. If not, then c_2 passes e_j at time step $\tau + k_1 + 1$, where $k_1 + 1 \in [1, 2\beta]$ is odd, and thus can be used by π. We have therefore shown that the following lemma holds.

LEMMA 4.5. *Suppose that packet π is transferred from r_i to r_j in phase ϕ of Algorithm A. In Algorithm B, suppose that R_j is ready for phase ϕ, and that at the beginning of phase ϕ, π is circulating in region R_i. Then, π can be transferred from region R_i to R_j during the first $4\beta^2+2\beta$ time steps of a round in phase ϕ of Algorithm B.*

Note that since the packet transfers over into an empty edge, it does not interfere with any packets that were already circulating.

4.1.5. Emulation of absorbtion. Suppose that packet π moves from node r_i to its destination node r_j in phase ϕ in store-and-forward Algorithm A (and is absorbed). Assume that R_j is ready for phase ϕ of Algorithm B and that π is circulating in R_i at the beginning of phase ϕ. We use the packet transfer emulation (section 4.1.4) to first move the packet from region R_i to R_j in phase ϕ. By Lemma 4.5, this can be done in the first $4\beta^2 + 2\beta$ time steps of a round of phase ϕ in which R_i is sending and R_j receiving. The packet then circulates in R_j at normal speed until it reaches its destination node, at which point it is absorbed. Since the packet completes the Euler cycle for R_j in at most 2β time steps, the number of time steps to be transferred, circulate, and be absorbed is at most $4\beta^2 + 4\beta \leq Z$, and this implies the following lemma.

LEMMA 4.6. *Suppose that packet π is transferred from r_i to r_j where it is absorbed in phase ϕ of Algorithm A. In Algorithm B, suppose that R_j is ready for phase ϕ, and that at the beginning of phase ϕ, π is circulating in region R_i. Then, π can be transferred from region R_i to R_j and absorbed during a single round of phase ϕ in Algorithm B.*

4.1.6. Emulation of buffering. Suppose that packet π is buffered at node r_i during phase ϕ of Algorithm A. Assume that in Algorithm B, packet π is already circulating in region R_i. Packet π will then continue to circulate in R_i uninterrupted through the entire phase ϕ. This is certainly possible unless some new packets entered the region (by transfer or injection) into the current edge of π. As we have already shown, injected or transferred packets do not interfere with already circulating packets, since they always enter on empty edges. We have the following lemma.

LEMMA 4.7. *If packet π is circulating in R_i at the end of phase $\phi - 1$ of Algorithm B, and in phase ϕ of Algorithm A, π is buffered at r_i, then in phase ϕ of Algorithm B, it can be buffered in R_i using circulation.*

4.2. Analysis of emulation.

First, we prove that Algorithm B correctly emulates Algorithm A. We then analyze the delivery time of Algorithm B in G in terms of the delivery time of Algorithm A in G'.

4.2.1. Correctness.

Assume that $\alpha = 2\gamma \leq |E|$ in order to guarantee the existence of the $[\alpha, \beta]$-partition. Algorithm B correctly emulates Algorithm A phase by phase if, at the end of every phase ϕ, the following two statements hold:

(i) in Algorithm A, packet π is in node r_i if and only if in Algorithm B it is circulating in region R_i;

(ii) in Algorithm A, packet π is injected (absorbed) at node r_i if and only if in Algorithm B, packet π is injected (absorbed) into region R_i.

We now prove by induction on the phase number ϕ that Algorithm B correctly emulates Algorithm A. Observe that when $\phi = 1$, Algorithm A can only inject packets into nodes. The conditions of Lemma 4.3 are satisfied, and since at most γ packets are injected into a node in G', Algorithm B can successfully inject these packets into the corresponding regions. Suppose that Algorithm B correctly emulates Algorithm A up to phase $\phi_0 \geq 1$. At the end of phase ϕ_0, there are at most γ packets circulating in any region R_i since every packet π in node r_i in the execution of Algorithm A is in region R_i in the execution of Algorithm B, and during the last time step of phase ϕ, Algorithm A can only be buffering at most γ packets (all other packets have gone to adjacent nodes). Thus, the conditions of Lemmas 4.3, 4.5, 4.6, and 4.7 are satisfied for every packet π. Every action that π could take in phase $\phi_0 + 1$ of Algorithm A can now be emulated in phase $\phi_0 + 1$ of Algorithm B. By induction, we now have the following theorem.

THEOREM 4.8 (correctness of emulation). *Algorithm B correctly emulates in G every phase in the execution of Algorithm A in G'. Hence, Algorithm B solves the batch problem Q without outgoing edge-buffers and implicitly constructs the paths $\widehat{\mathcal{P}}$ and a collision-free schedule in them.*

4.2.2. Bufferless delivery time.

Let $\mathcal{T}_B(Q)$ be the delivery time for bufferless Algorithm B to solve the batch problem Q (using initial paths $\mathcal{P}$), and let $\Phi_A(Q')$ be the number of phases required by Algorithm A to solve scheduling problem Q' (corresponding to the batch problem Q) on the region graph G'. Since Algorithm B emulates Algorithm A phase for phase, Algorithm B uses the same number of phases as Algorithm B; i.e., $\Phi_B(Q) = \Phi_A(Q')$. The delivery time is given by

$$\begin{aligned}\mathcal{T}_B(Q) &\leq \Xi \cdot Z \cdot \Phi_B(Q)\\ &\leq 576\gamma^2 \lceil \log \chi \rceil \cdot \Phi_A(Q'),\end{aligned}$$

where we have used $Z = 4\beta^2 + 4\beta \leq 8\beta^2$, $\beta = 6\gamma - 3 \leq 6\gamma$, and $\Xi = 2\sigma = 2\lceil \log \chi \rceil$ (χ is the chromatic number of the region graph G').

THEOREM 4.9 (bufferless delivery time). *$\mathcal{T}_B(Q) \leq c \cdot \Phi_A(Q') \cdot \gamma^2 \cdot \log \chi$ for some constant c.*

Since $\chi \leq n' \leq |E|/\alpha = O(n^2)$, we have that $\mathcal{T}_B(Q) = O(\Phi_A(Q') \cdot \gamma^2 \cdot \log n)$.

Recap. The bufferless algorithm solves the batch problem by deterministically emulating a store-and-forward algorithm on a corresponding scheduling problem in the region graph. Buffering is replaced by packet circulation, and the cost of the emulation is $O(\gamma^2 \cdot \log n)$, where γ is the node-buffer size required by the store-and-forward algorithm. Any store-and-forward algorithm that satisfies the two required properties for emulation (section 3) will give a valid bufferless algorithm. The more efficient the store-and-forward algorithm is (in terms of phases) and the smaller the buffer size used, the more efficient the bufferless algorithm will be.

5. A universal bufferless algorithm.

By combining the results in sections 3 and 4 (specifically Theorems 3.6 and 4.9), we will obtain a specific randomized universal bufferless algorithm which we denote *Algorithm* B_1. Algorithm B_1 emulates the store-and-forward Algorithm A_1. The buffer size required by Algorithm A_1 is $\gamma \geq 6\log(n' + 2N)$. Since $n' \leq |E|/\alpha \leq |E|$, we can set $\gamma = 6\log(|E| + 2N)$. In order to apply the bufferless emulation algorithm, we need an $[\alpha, \beta]$-partition, where $\alpha = 2\gamma$ and $\beta = 6\gamma - 3$. Since $\alpha \leq |E|$, we must have that $2\gamma \leq |E|$. Substituting the expression for γ, we find that $2N \leq 2^{|E|/12} - |E|$. Thus, for the case where $2N > 2^{|E|/12} - |E|$, we need to apply a different approach. We examine these two cases separately.

5.1. The case $2N \leq 2^{|E|/12} - |E|$.

In this case, we can apply the bufferless emulation with $\gamma = 6\log(|E| + 2N)$. Note that γ is independent of the size of G'. Combining Theorems 3.6 and 4.9, we obtain

$$\begin{aligned}\mathcal{T}_{B_1}(Q) &\leq c \cdot \Phi_{A_1}(Q') \cdot \gamma^2 \log \chi \\ &\leq c \cdot \left(12\frac{\overline{C'}}{\gamma} + D'\right) \cdot \log^2(|E| + 2N) \cdot \log \chi \\ &\leq c \cdot (C + D) \cdot \log^2(|E| + 2N) \cdot \log |E|,\end{aligned}$$

where c represents a generic constant, not necessarily the same from line to line. The last inequality follows by using Lemma 2.5 and the facts that $\beta \leq 6\gamma$ and $\chi \leq n' \leq |E|/\alpha \leq |E|$. Since the randomized store-and-forward Algorithm A_1 succeeds with probability at least $1 - 1/(n' + 2N)$, the bufferless Algorithm B_1 has the same probability of success (i.e., it correctly sends the packets with the advertised delivery time with the same probability). Since $n' \geq |E|/\beta$, $\beta = 6\gamma - 3$, $\gamma = 6\log(|E| + 2N)$, and $|E| \leq n^2$, we have that the probability of success is at least

$$1 - \frac{1}{n' + 2N} \geq 1 - \frac{1}{\frac{|E|}{36 \cdot \log(|E|+2N) - 3} + 2N} = 1 - O((n + N)^{-\lambda})$$

for some constant $\lambda > 0$.

We now consider how the bufferless emulation of the store-and-forward algorithm can be performed in a distributed manner by the nodes of network G; that is, packet forwarding decisions can be made locally at each node. In order to do so, we need to assume that every node in G knows the following before the algorithm starts: the network topology G, the value of congestion C, and the number of packets N (such assumptions are commonly made in distributed bufferless routing algorithms [17, 34]). Based on these parameters, each node in G can compute the parameters which are necessary for the emulation, such as the structure of G', the buffer size γ, and the duration of phases. In this way, the nodes in G know what kinds of actions to perform at each time step in order to emulate the actions of the store-and-forward Algorithm A_1. Note that a node does need to have a priori information about the packets with origin at other nodes.

Alternatively, instead of knowing G, each node could have been supplied a priori information about G'. In particular, each node needs to have information about the regions of G' in which it participates. However, G' depends on N, since each region has $O(\log(|E|+N))$ edges. Different batch problems may have different values of N and use different graphs G'. If we write $2^y \leq N < 2^{y+1}$, where $0 \leq y \leq |E|/12$, we observe that there are in total $\Theta(|E|)$ different graphs G' that we could use for the emulation, one graph for each range $[2^y, 2^{y+1})$ of the value N. Each node in G could have been informed about all these possible graphs of G', and could choose an appropriate one for the current value of N.

5.2. The case $2N > 2^{|E|/12} - |E|$. In this case, we send the N packets of batch problem Q to their destinations one after the other along their prespecified paths in G. Each packet takes time at most D to be delivered to its destination; thus, the total delivery time to send all the packets is at most DN. By the pigeonhole principle, $C \geq N/|E|$, and thus $C > (2^{|E|/12} - |E|)/2|E|$. Since $|E| = O(\log N)$ and $D \leq |E|$, the delivery time is $ND \leq CD|E| = O(C \log^2 N)$.

This simple algorithm can be converted to a distributed algorithm, where nodes make local decisions about packets, using packet priorities. There are two packet priorities, 0 and 1; in collisions, packets with priority 1 win, while packets with priority 0 are deflected. The details are given below. The algorithm proceeds in phases of duration $|E| = O(\log N)$.

i. At the beginning of the first phase, for each outgoing edge e, a node injects at most one packet (if it has one) with e being the first edge in the path of the packet. All packets start with priority 1.
ii. For the duration of the phase, priority 1 packets always try to move to their destinations along their path, unless there is a collision. In a collision, priority 1 packets have precedence over priority 0 packets. If multiple priority 1 packets collide, then one of them (arbitrarily) wins the collision, and all the other priority 1 packets involved in the collision drop to priority 0.
iii. Once a packet becomes priority 0, it randomly follows any available edge from its current node toward its destination in hot-potato style. If a packet of priority 0 happens to arrive at its destination during a phase, then it is absorbed.
iv. At the end of the phase, each remaining packet resets its path to be the shortest path from its current node to its destination.
v. For each subsequent phase, all nodes inject at most one packet per outgoing edge as at the beginning of the first phase, with one exception: They do not inject a packet with initial edge e if some other packet is already at the node from the previous phase and has e as the next edge in its new (shortest) path to its destination. All packets (newly injected and from previous phase) begin the phase with priority 1.
vi. The entire process repeats for N phases.

If at the beginning of the phase there are packets in the network (newly injected or from previous phase), then all of them have priority 1. During the phase, at least one packet will retain priority 1, since in collisions involving packets with priority 1, at least one priority 1 packet survives. Further, in every time step, a priority 1 packet (unless it drops to priority 0) moves one edge closer to its destination. Since any path is no longer than the number of edges in the network, after $D \leq |E|$ time steps, all the priority 1 packets that did not drop to priority 0 (at least 1 of them exists) have been absorbed at their respective destinations. Thus, at least one packet is absorbed in each phase that starts with at least one packet. Note that if nodes have packets to inject, then at the beginning of the phase there is at least one packet in the network (newly injected or from the previous phase). Therefore, at most N phases are needed. Since each phase has $|E|$ time steps, the total delivery time is at most $N|E| \leq C|E|^2 = O(C \log^2 N)$.

5.3. Main result. To wrap up, when $2N \leq 2^{|E|/12} - |E|$, we use bufferless emulation which with high probability obtains a delivery time of $O((C + D) \log^2(|E| + 2N) \log |E|)$. Otherwise, we use the simple brute-force algorithm of sending the packets one by one, which has delivery time $O(C \log^2 N)$. Combining these two results and using the fact that $|E| = O(n^2)$, we have a universal bufferless Algorithm B_1 which with high probability has near-optimal delivery time.

THEOREM 5.1 (delivery time of Algorithm B_1). *Given paths $\mathcal{P}$ that satisfy batch problem Q, bufferless Algorithm B_1 delivers the packets in time $\mathcal{T}_{B_1}(Q) = O((C + D) \cdot \log^3(n + N))$, with probability $1 - O((n + N)^{-\lambda})$, for some constant $\lambda > 0$.*

When choosing optimal initial paths, Theorem 5.1 establishes our main result, Theorem 1.1. Furthermore, using the distributed version for each case $2N \leq 2^{|E|/12} - |E|$ and $2N > 2^{|E|/12} - |E|$, we obtain the distributed version of B_1.

6. Discussion. Our main goal has been to establish the existence of universal, near optimal (to within polylog factors) bufferless communication algorithms. Our proof of this fact has been *constructive*, using a bufferless emulation technique to emulate a store-and-forward algorithm on a batch scheduling problem related to the original batch packet problem. The heart of the emulation is to replace buffering with packet circulation. The algorithm we have given is distributed modulo the need for nodes to know C and N. Note that in Theorem 5.1, it is crucial to allow the paths to deviate from the preselected paths; otherwise, it is possible to construct problems for which the optimal bufferless routing time is at least a $\sqrt{N}$ factor from optimal [22].

We briefly discuss some interesting directions for future work. The most natural question is whether some of the polylog factors can be removed. Two of the polylog factors arise purely from the bufferless emulation, and our guess is that these polylog factors may not be necessary. Specifically, for a given batch problem Q with N packets on a network G, we can define the *Q-bufferless efficiency* $\rho_B(Q; N, G)$ as the ratio between the smallest possible delivery time of a bufferless algorithm for Q and the smallest possible delivery time of a store-and-forward algorithm for Q. The *bufferless efficiency* $\rho_B(N, G)$ is the maximum possible value of the Q-bufferless efficiency over all batch problems, $\rho_B(N, G) = \sup_Q \rho_B(Q; N, G)$. Our result shows that the bufferless efficiency is polylogarithmicaly bounded, $\rho_B(N, G) = O(\log^3(n + N))$, where n is the size of G. An interesting problem is to determine tighter asymptotic upper bounds as well as lower bounds for the bufferless efficiency for specific as well as arbitrary networks (for example, it is shown in [21] that the bufferless efficiency for leveled networks is only $O(\log(n + N))$).

A related question is whether special purpose store-and-forward scheduling algorithms can be used with our emulation technique to obtain *optimal* bufferless delivery time on specific classes of networks. If the region graph can be efficiently colored (for example, fixed degree region graphs) and if the node-buffering requirement on such region graphs is some constant, then the resulting bufferless emulation only adds an additional constant factor to the delivery time. A good candidate for such a result is the mesh network, since a natural decomposition into regions would yield another mesh-like network, with good properties.

A slightly different line of enquiry is to determine whether the algorithm we have given can be made dynamic (i.e., not requiring a priori knowledge of C and N), in addition to being distributed. Such a result would show the existence of near-optimal bufferless algorithms for dynamic packet problems.

Acknowledgment. We are indebted to the referees of this journal for their valuable and insightful comments.

REFERENCES

[1] M. Adler, S. Khanna, R. Rajaraman, and A. Rosen, *Time-constrained scheduling of weighted packets on trees and meshes*, Algorithmica, 36 (2003), pp. 123–152.

[2] M. Adler, A. L. Rosenberg, R. K. Sitaraman, and W. Unger, *Scheduling time-constrained communication in linear networks*, Theory Comput. Syst., 35 (2002), pp. 599–623.

[3] N. Alon, F. R. K. Chung, and R. L. Graham, *Routing permutations on graphs via matchings*, SIAM J. Discrete Math., 7 (1994), pp. 513–530.

[4] S. Alstrup, J. Holm, K. de Lichtenberg, and M. Thorup, *Direct routing on trees*, in Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, 1998, pp. 342–349.

[5] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts, *On-line routing of virtual circuits with applications to load balancing and machine scheduling*, J. ACM, 44 (1997), pp. 486–504.

[6] B. Awerbuch and Y. Azar, *Local optimization of global objectives: Competitive distributed deadlock resolution and resource allocation*, in Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science, Santa Fe, NM, 1994, pp. 240–249.

[7] A. Bar-Noy, P. Raghavan, B. Schieber, and H. Tamaki, *Fast deflection routing for packets and worms*, in Proceedings of the Twelfth Annual ACM Symposium on Principles of Distributed Computing, Ithaca, NY, 1993, pp. 75–86.

[8] P. Baran, *On distributed communications networks*, IEEE Trans. Comm. Systems, 12 (1964), pp. 1–9.

[9] I. Ben-Aroya, T. Eilam, and A. Schuster, *Greedy hot-potato routing on the two-dimensional mesh*, Distrib. Comput., 9 (1995), pp. 3–19.

[10] A. Ben-Dor, S. Halevi, and A. Schuster, *Potential function analysis of greedy hot-potato routing*, Theory Comput. Syst., 31 (1998), pp. 41–61.

[11] P. Berenbrink and C. Scheideler, *Locally efficient on-line strategies for routing packets along fixed paths*, in Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, Baltimore, MD, 1999, pp. 112–121.

[12] S. N. Bhatt, G. Bilardi, G. Pucci, A. G. Ranade, A. L. Rosenberg, and E. J. Schwabe, *On bufferless routing of variable-length message in leveled networks*, IEEE Trans. Comput., 45 (1996), pp. 714–729.

[13] A. Borodin and J. E. Hopcroft, *Routing, merging, and sorting on parallel models of computation*, J. Comput. System Sci., 30 (1985), pp. 130–145.

[14] A. Borodin, Y. Rabani, and B. Schieber, *Deterministic many-to-many hot potato routing*, IEEE Trans. Parallel Distrib. Systems, 8 (1997), pp. 587–596.

[15] J. T. Brassil and R. L. Cruz, *Bounds on maximum delay in networks with deflection routing*, IEEE Trans. Parallel Distrib. Systems, 6 (1995), pp. 724–732.

[16] A. Broder and E. Upfal, *Dynamic deflection routing on arrays*, in Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, 1996, pp. 348–358.

[17] C. Busch, *$\widetilde{O}$(congestion + dilation) hot-potato routing on leveled networks*, Theory Comput. Syst., 37 (2004), pp. 371–396.

[18] C. Busch, M. Herlihy, and R. Wattenhofer, *Hard-potato routing*, in Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, 2000, pp. 278–285.

[19] C. Busch, M. Herlihy, and R. Wattenhofer, *Randomized greedy hot-potato routing*, in Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, 2000, pp. 458–466.

[20] C. Busch, M. Herlihy, and R. Wattenhofer, *Routing without flow control*, in Proceedings of the Thirteenth ACM Symposium on Parallel Algorithms and Architectures, 2001, pp. 11–20.

[21] C. Busch, S. Kelkar, and M. Magdon-Ismail, *Efficient bufferless routing on leveled networks*, in Proceedings of the Eleventh International Conference on Parallel and Distributed Computing (Euro-Par 2005), Lisbon, Portugal, Lecture Notes in Comput. Sci. 3648, Springer-Verlag, Berlin, 2005, pp. 931–940.

[22] C. Busch, M. Magdon-Ismail, M. Mavronicolas, and P. Spirakis, *Direct routing: Algorithms and complexity*, Algorithmica, 45 (2006), pp. 45–68.

[23] C. Busch, M. Magdon-Ismail, M. Mavronicolas, and R. Wattenhofer, *Near-optimal hot potato routing on trees*, in Proceedings of Euro-Par 2004, Pisa, Italy, Lecture Notes in Comput. Sci. 3149, Springer-Verlag, Berlin, 2004, pp. 820–827.

[24] R. Cypher, F. Meyer auf der Heide, C. Scheideler, and B. Vöcking, *Universal algorithms for store-and-forward and wormhole routing*, in Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing, 1996, pp. 356–365.

[25] U. Feige, *Nonmonotonic phenomena in packet routing*, in Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, Atlanta, GA, 1999, pp. 583–591.

[26] U. Feige and P. Raghavan, *Exact analysis of hot-potato routing*, in Proceedings of the Thirty-Third Annual IEEE Symposium on Foundations of Computer Science, Pittsburgh, PA, 1992, pp. 553–562.

[27] R. I. Greenberg and H.-C. Oh, *Universal wormhole routing*, IEEE Trans. Parallel Distrib. Systems, 8 (1997), pp. 254–262.

[28] B. Hajek, *Bounds on evacuation time for deflection routing*, Distrib. Comput., 5 (1991), pp. 1–6.

[29] C. Kaklamanis, D. Krizanc, and S. Rao, *Hot-potato routing on processor arrays*, in Proceedings of the Fifth Annual ACM Symposium on Parallel Algorithms and Architectures, Velen, Germany, 1993, pp. 273–282.

[30] M. Kaufmann, H. Lauer, and H. Schroder, *Fast deterministic hot-potato routing on meshes*, in Proceedings of the Fifth International Symposium on Algorithms and Computation (ISAAC), Lecture Notes in Comput. Sci 834, Springer-Verlag, Berlin, 1994, pp. 333–341.

[31] M. Kunde, *A new bound for pure greedy hot potato routing*, in Proceedings of the Twenty-Fourth Annual Symposium on Theoretical Aspects of Computer Science (Aachen, Germany), Lecture Notes in Comput. Sci. 4393, Springer-Verlag, Berlin, 2007, pp. 49–60.

[32] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*, Morgan Kaufmann, San Mateo, CA, 1992.

[33] F. T. Leighton, B. M. Maggs, and A. W. Richa, *Fast algorithms for finding O(congestion + dilation) packet routing schedules*, Combinatorica, 19 (1999), pp. 375–401.

[34] F. T. Leighton, B. M. Maggs, A. G. Ranade, and S. B. Rao, *Randomized routing and sorting on fixed-connection networks*, J. Algorithms, 17 (1994), pp. 157–205.

[35] F. T. Leighton, B. M. Maggs, and S. B. Rao, *Packet routing and job-scheduling in* $O(\textit{congestion} + \textit{dilation})$ *steps*, Combinatorica, 14 (1994), pp. 167–186.

[36] F. Meyer auf der Heide and C. Scheideler, *Routing with bounded buffers and hot-potato routing in vertex-symmetric networks*, in Proceedings of the Third Annual European Symposium on Algorithms (Corfu, Greece), Lecture Notes in Comput. Sci. 979, Springer-Verlag, Berlin, Paul G. Spirakis, ed., 1995, pp. 341–354.

[37] F. Meyer auf der Heide and B. Vöcking, *Shortest-path routing in arbitrary networks*, J. Algorithms, 31 (1999), pp. 105–131.

[38] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, Cambridge, UK, 1995.

[39] I. Newman and A. Schuster, *Hot-potato algorithms for permutation routing*, IEEE Trans. Parallel Distrib. Systems, 6 (1995), pp. 1168–1176.

[40] R. Ostrovsky and Y. Rabani, *Universal* $O(\textit{congestion} + \textit{dilation} + \log^{1+\varepsilon} N)$ *local control packet switching algorithms*, in Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, New York, 1997, pp. 644–653.

[41] G. E. Pantziou, A. Roberts, and A. Symvonis, *Many-to-many routing on trees via matchings*, Theoret. Comput. Sci., 185 (1997), pp. 347–377.

[42] R. Prager, *An Algorithm for Routing in Hypercube Networks*, Master's thesis, Computer Science Department, University of Toronto, Toronto, 1986.

[43] Y. Rabani and É. Tardos, *Distributed packet switching in arbitrary networks*, in Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, Philadelphia, 1996, pp. 366–375.

[44] P. Raghavan and C. D. Thompson, *Randomized rounding: A technique for provably good algorithms and algorithmic proofs*, Combinatorica, 7 (1987), pp. 365–374.

[45] R. Ramaswami and K. N. Sivarajan, *Optical Networks: A Practical Perspective*, Morgan Kaufmann, San Mateo, CA, 1998.

[46] A. Roberts, A. Symvonis, and D. R. Wood, *Lower bounds for hot-potato permutation routing on trees*, in Proceedings of the Seventh International Colloquium on Structural Information and Communication Complexity (SIROCCO), M. Flammini, E. Nardelli, G. Proietti, and P. Spirakis, eds., Carleton Scientific, Waterloo, ON, 2000, pp. 281–295.

[47] C. Scheideler, *Universal Routing Strategies for Interconnection Networks*, Lecture Notes in Comput. Sci. 1390, Springer-Verlag, Berlin, Germany, 1998.

[48] P. Spirakis and V. Triantafillou, *Pure greedy hot-potato routing in the 2-D mesh with random destinations*, Parallel Process. Lett., 7 (1997), pp. 249–258.

[49] A. Srinivasan and C-P. Teo, *A constant-factor approximation algorithm for packet routing and balancing local vs. global criteria*, SIAM J. Comput., 30 (2001), pp. 2051–2068.

[50] A. Symvonis, *Routing on trees*, Inform. Process. Lett., 57 (1996), pp. 215–223.

[51] L. Zhang, *Optimal bounds for matching routing on trees*, in Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, 1997, pp. 445–453.

SIAM J. COMPUT.
Vol. 37, No. 4, pp. 1163–1181

DISTRIBUTED SELFISH LOAD BALANCING[*]

PETRA BERENBRINK[†], TOM FRIEDETZKY[‡], LESLIE ANN GOLDBERG[§], PAUL W. GOLDBERG[§], ZENGJIAN HU[†], AND RUSSELL MARTIN[§]

Abstract. Suppose that a set of m tasks are to be shared as equally as possible among a set of n resources. A game-theoretic mechanism to find a suitable allocation is to associate each task with a "selfish agent" and require each agent to select a resource, with the cost of a resource being the number of agents that select it. Agents would then be expected to migrate from overloaded to underloaded resources, until the allocation becomes balanced. Recent work has studied the question of how this can take place within a distributed setting in which agents migrate selfishly without any centralized control. In this paper we discuss a natural protocol for the agents which combines the following desirable features: It can be implemented in a strongly distributed setting, uses no central control, and has good convergence properties. For $m \gg n$, the system becomes approximately balanced (an ϵ-Nash equilibrium) in expected time $O(\log \log m)$. We show using a martingale technique that the process converges to a perfectly balanced allocation in expected time $O(\log \log m + n^4)$. We also give a lower bound of $\Omega(\max\{\log \log m, n\})$ for the convergence time.

Key words. load balancing, reallocation, equilibrium, convergence

AMS subject classifications. 68Q25, 68W20, 68W15, 91A80

DOI. 10.1137/060660345

1. Introduction.

Suppose a consumer learns the price she would be charged by some domestic power supplier other than the one she is currently using. It is plausible that if the alternative price is lower than the price she is currently paying, then there is some possibility that she will switch to the new power supplier. Furthermore, she is more likely to switch if the ratio of the current price to the new price is large. If there is only a small savings, then it becomes unattractive to make the switch, since an influx of new business (hers and that of other consumers) may drive up the price of the new power supplier and make it no longer competitive.

We study a simple mathematical model of the above natural rule, in the context of a load balancing (or task allocation) scenario that has received a lot of recent attention. We assume the presence of many individual users who may assign their tasks to chosen resources. The users are *selfish* in the sense that they attempt to optimize their own situation, i.e., try to assign their tasks to minimally loaded resources, without trying to optimize the global situation. In general, a *Nash equilibrium* (NE) among a set of selfish users is a state in which no user has the incentive to change her current decision. In our setting, this corresponds to no user having an incentive to reallocate

[*] Received by the editors May 19, 2006; accepted for publication (in revised form) April 23, 2007; published electronically November 21, 2007. A preliminary version appeared in *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2006, pp. 354–363. This work was partially supported by the EPSRC grants "Discontinuous Behaviour in the Complexity of Randomized Algorithms" and "Algorithmics of Network-sharing Games," and by the Natural Sciences and Engineering Research Council of Canada (NSERC) discovery grant 250284-2002.

http://www.siam.org/journals/sicomp/37-4/66034.html

[†] School of Computing Science, Simon Fraser University, 8888 University Drive, Burnaby V5A 1S6, BC, Canada (petra@cs.sfu.ca, zhu@cs.sfu.ca).

[‡] Department of Computer Science, Durham University, Science Labs, South Road, Durham, DH1 3LE, UK (tom.friedetzky@dur.ac.uk).

[§] Department of Computer Science, University of Liverpool, Ashton Building, Liverpool, L69 3BX, UK (l.a.goldberg@liverpool.ac.uk, pwg@csc.liv.ac.uk, r.martin@csc.liv.ac.uk).

her task to some other resource. An ϵ-Nash equilibrium (ϵ-NE) is a standard notion of an approximate NE, and is a state in which no user can reduce her cost by a multiplicative factor of less than $1-\epsilon$ by changing action. Here we do not focus on the *quality* of equilibria but rather on the (perhaps more algorithmic) question of convergence time to such a state.

We assume a strongly distributed and concurrent setting; i.e., there is no centralized control mechanism whatsoever, and all users may choose to reallocate their tasks at the same time. Thus, we do not (and cannot) use the *elementary step system* [25] (discussed in more detail in the next section), where the assumption is that at most one user may reallocate her task at any given stage.

Throughout we let m denote the number of tasks (in the above discussion, customers) and n denote the number of resources (power suppliers). As hinted at in the above discussion, we assume that typically $m \gg n$. In a single time step (or round) each task does the following: Let i be the resource currently being used by the task. Select j uniformly at random from $\{1,\ldots,n\}$ and find the load of resource j. Let X_i and X_j be the loads of resources i and j, respectively. If $X_j < X_i$, migrate from i to j with a probability of $1 - X_j/X_i$; the transition from round t to round $t+1$ is given in Figure 1.1. Notice that if we had unconditional migrations, i.e., without an additional coin flip (move only with probability $1 - X_j(t)/X_i(t)$), then this might lead to an unstable system; consider, for example, the case $m = 2$ with initially most tasks assigned to one of the resources. The overload would oscillate between the two resources, with a load ratio tending towards 2:1. (This observation about the risk of oscillation has also been made in similar contexts in [12, 11], and we will not elaborate on it further.)

```
For each task $b$ do in parallel
  Let $i_b$ be the current resource of task $b$
  Choose resource $j_b$ uniformly at random
  Let $X_{i_b}(t)$ be the current load of resource $i$
  Let $X_{j_b}(t)$ be the current load of resource $j$
  If $X_{i_b}(t) > X_{j_b}(t)$ then
    Move task $b$ from resource $i_b$ to $j_b$ with probability $1 - X_{j_b}(t)/X_{i_b}(t)$
```

FIG. 1.1. *The protocol with "neutral moves" allowed.*

It can easily be seen that if all tasks use the above policy, then the expected load of every resource at the next step is m/n.

OBSERVATION 1.1. *Regardless of the load distribution at time step t, the expected load of every resource at the next step is m/n.*

Proof. To see this, assume that the loads $X_i(t)$ are arranged in descending order so that $X_j(t) \geq X_{j+1}(t)$ and note that

$$\begin{aligned}\mathbb{E}[X_i(t+1)] &= X_i(t) + \sum_{\ell=1}^{i-1} \frac{1}{n} X_\ell(t)\left(1 - \frac{X_i(t)}{X_\ell(t)}\right) - \sum_{\ell=i+1}^{n} \frac{1}{n} X_i(t)\left(1 - \frac{X_\ell(t)}{X_i(t)}\right)\\ &= X_i(t) + \frac{1}{n}\sum_{\ell=1}^{i-1}(X_\ell(t) - X_i(t)) - \frac{1}{n}\sum_{\ell=i+1}^{n}(X_i(t) - X_\ell(t))\\ &= X_i(t) + \frac{1}{n}\sum_{\ell=1}^{n}(X_\ell(t) - X_i(t)) = \frac{1}{n}\sum_{\ell=1}^{n} X_\ell(t) = \frac{m}{n}. \qquad \square\end{aligned}$$

This provides a compelling motivation for the policy, which is that as a result, no task has an incentive to deviate unilaterally from this policy. This implies that in the terminology of [8] it is a *Nash rerouting policy.* It is also a simple regret-minimizing policy in the sense of [2] since the average cost of resources used by an agent is no higher than the best choice of a single resource to be used repeatedly. Although the above rule is very natural and has the nice properties described above, we show that it may take a long time to converge to a perfectly balanced allocation of tasks to resources. We address this problem as follows. Define a *neutral move* to be a task migration from a resource with load ℓ at time t to a resource with load $\ell - 1$ at time t (so, if no other task migrates, then the cost to the migrating task is unchanged). We consider a modification in which neutral moves are specifically disallowed (see Figure 2.1). That seemingly minor change ensures fast convergence from an almost balanced state to a perfectly balanced state. To summarize, here are the most important features of the modified protocol:

- We do not need any global information whatsoever (apart from the number of available resources); in particular, a task does not need to know the total number of tasks in the system. Also, it is strongly distributed and concurrent. If additional tasks were to enter the system, it would rapidly converge once again, with no outside intervention.
- A migrating task needs to query the load of only one other resource (thus, doing a constant amount of work in each round).
- When a task finds a resource with a significantly smaller load (that is, a load that is smaller by at least two), the migration policy is *exactly* the same as that used by the Nash rerouting policy of Figure 1.1, so the incentive is to use that probability.
- When a task finds a resource with a load that is smaller by exactly one unit, the migration policy is sufficiently close to the Nash rerouting policy that the difference in expected load is at most one, and there is little incentive to deviate.
- The protocol is simple (as well as provably efficient) enough to convince users to actually stick to it.

1.1. Related work. We are studying a simple kind of *congestion game.* In their general form, congestion games specify a set of agents, a set of resources, and, for each agent, a set of allowed strategies, where a strategy is the selection of a subset of the resources (in this paper, any singleton subset is allowed). The cost of a resource is a nondecreasing function of the number of agents using it, and the cost for an agent is the sum of the costs of resources it uses. A classical result due to Rosenthal [26] is that pure Nash equilibria (NEs) always exist for congestion games, and this is shown by exhibiting a potential function; they are a type of *potential game* [24]. The potential function also establishes that pure NEs can be found via sequences of "better-response" moves, in which agents repeatedly switch to lower-cost strategies. The potential function we use later in this paper is that of [26], modulo a linear rescaling.

These results do not show how to find NEs efficiently, the problem being that in the worst case, sequences of these self-improving moves may be exponentially long. The following questions arise: When can NEs be found by any efficient algorithm, and if so, can they be found via an algorithm that purports to be a realistic model of agents' behavior? Regarding the first of these questions, the answer is no in the general setting (the problem is PLS-complete for general congestion games [9]; see also [1, 3]). PLS-completeness (introduced in [17]) is a generally accepted criterion for intractability of computational problems in which we seek a local optimum of a given objective function.

However, due to the basic fact of [26, 24] that pure NEs are sure to result from a sufficiently long better-response sequence, many algorithms for finding them are based on such sequences. An important subclass is the *elementary step system* (ESS), proposed in Orda, Rom, and Shimkin [25], which consists of best-response moves (where a migrating agent switches not to any improved choice but to one that is optimal at the time of migration). For matroid games (a class of congestion games that includes the ones we consider here), Ackermann, Röglin, and Vöcking [1] show that best-response sequences must have length polynomial in the number of players, resources, and maximal rank of the matroids. In this paper we consider the special case of singleton congestion games (where players' strategies are always single resources, and thus the ranks of the matroids is 1). For these games, Ieong et al. [16] give polynomial bounds for best-response and better-response sequences. Chien and Sinclair [3] study a version of the ESS in the context of approximate NEs, and show that in some cases the ϵ-*Nash dynamics* may find an ϵ-NE where finding an exact NE is PLS-complete. Mirrokni and Vetta [22] study the convergence rate of the ESS to solutions, and the quality of the approximation after limited iterations.

While best- and better-response dynamics are a plausible model of selfish behavior, the associated algorithms typically require that migrations be done one-by-one, and another common assumption is that best- (not better-) responses are always selected. This means that to some extent, agents are being assumed to be governed by a centralized algorithm that finds an NE, raising the question of what sort of *distributed* algorithms can do so, especially if agents have limited information about the state of the system (and so may not be able to find best responses). That issue is of central importance to us in this paper. Goldberg [14] studied situations where simple better-response approaches can be realized as weakly distributed algorithms (where each agent looks for moves independently of the others, but it is assumed that moves take place consecutively, not simultaneously). In a strongly distributed setting (as we study here), where moves may occur simultaneously, we need to address the possibility that a change of strategy may increase an agent's cost. It may happen that after a best response has been identified, it is not optimal at the time it is executed. Even-Dar and Mansour [8] consider concurrent, independent rerouting decisions where tasks are allowed to migrate from overloaded to underloaded resources. Their rerouting process terminates in expected $O(\log \log m + \log n)$ rounds when the system reaches an NE. Note that their convergence rate as a function of the number n of resources is faster than the one we obtain in this paper. The reason is that it requires agents to have a certain amount of global knowledge. A task is required to know whether its resource is overloaded (having above-average load), and tasks on underloaded resources do not migrate at all. Our rerouting policy does not require that agents know anything other than their current resource load and the load of a randomly chosen alternative. Even-Dar and Mansour also present a general framework that can be used to show a logarithmic convergence rate for a wide class of rerouting strategies. Our protocol does not fall into that class, since we do not require migrations to occur only from overloaded resources. Note that our lower bound is linear in n (thus, more than logarithmic).

Distributed algorithms have been studied in the Wardrop setting (the limit of infinitely many agents), for which recent work has also extensively studied the coordination ratio [28, 27]. Fischer, Räcke, and Vöcking [11] investigate convergence to

Wardrop equilibria for games where agents select paths through a shared network to route their traffic. (Singleton games correspond to a network of parallel links.) Their rerouting strategies are slightly different to ours—they assume that in each round, an agent queries a path with probability proportional to the traffic on that path. Here we assume that paths (individual elements of a set of parallel links) are queried uniformly at random, so that agents can be assumed to have minimal knowledge. As in this paper, the probability of switching to a better path depends on the latency difference, and care has to be taken to avoid oscillation. Also in the Wardrop setting, Blum, Even-Dar, and Ligett [2] show that approximate NE is the outcome of *regret-minimizing* rerouting strategies, in which an agent's cost, averaged over time, should approximate the cost of the best individual link available to that agent.

Certain generalizations of singleton games have also been considered. These generalizations are not strictly congestion games according to the standard definition we gave above, but many ideas carry over. One version introduced by Koutsoupias and Papadimitriou [18] has been studied extensively in different contexts (for example, [20, 6, 13, 4, 28]). In this generalization, each task may have a numerical *weight* (sometimes called traffic, or demand), and each resource has a *speed* (or capacity). The cost of using a resource is the total weight of tasks using it, divided by its speed. Even-Dar, Kesselman, and Mansour [7] give a generalized version of the potential function of [26] that applies to these games and which was subsequently used in [14]. For these games, however, it seems harder to find polynomial-length best-response sequences. Feldman et al. [10] show how a sequence of steps may lead to NEs, under the weaker condition that the maximal cost experienced by agents must not increase, but individual steps need not necessarily be "selfish." They also note that poorly chosen better-response moves may lead to an exponential convergence rate. Another generalization of singleton games is *player-specific* cost functions [21], which allow different agents to have different cost functions for the same resource. In this setting there is no potential function, and better-response dynamics may cycle, although it remains the case that pure NEs always exist.

Our rerouting strategy is also related to reallocation processes for balls-into-bins games. The goal of a balls-into-bins game is to allocate m balls as evenly as possible into n bins. It is well known that a fairly even distribution can be achieved if every ball is allowed to randomly choose d bins and then the ball is allocated to the least loaded among the chosen bins (see [23] for an overview). Czumaj, Riley, and Scheideler [5] consider such an allocation where each ball initially chooses two bins. They show that, in a polynomial number of steps, the reallocation process ends up in a state with maximum load at most $\lceil m/n \rceil + 1$. Sanders, Egner, and Korst [29] show that a maximum load of $\lceil m/n \rceil + 1$ is optimal if every ball is restricted to two random choices.

In conclusion, this paper sits at one end of a spectrum in which we study a very simple load-balancing game, but we seek solutions in a very adverse setting in which agents have, at any point in time, a minimal amount of information about the state of their environment and carry out actions simultaneously in a strongly distributed sense.

1.2. Overview of our results. Section 3 deals with upper bounds on convergence time. The main result, Theorem 3.1, is that the protocol of Figure 2.1 converges to an NE within expected time $O(\log\log m + n^4)$.

The proof of Theorem 3.1 shows that the system becomes *approximately* balanced very rapidly. Specifically, Corollary 3.11 shows that if $n \le m^{1/3}$, then for all ϵ, either version of the distributed protocol (with or without neutral moves allowed) attains an ϵ-NE (where all load ratios are within $[1-\epsilon, 1+\epsilon]$; we use ϵ to denote a multiplicative factor as in [3]) in expected $O(\log\log m)$ rounds. The rest of section 3 analyzes the protocol of Figure 2.1. It is shown that within an additional $O(n^4)$ rounds the system becomes optimally balanced.

In section 4, we provide two lower bound results. The first one, Theorem 4.1, shows that the first protocol (of Figure 1.1, including moves that do not necessarily yield a strict improvement for an individual task but allow for simply "neutral" moves as well) results in exponential (in n) expected convergence time. Finally, in Theorem 4.2 we provide a general lower bound (regardless of which of the two protocols is being used) on the expected convergence time of $\Omega(\log\log m)$. This lower bound matches the upper bound as a function of m.

2. Notation. There are m tasks and n resources. An assignment of tasks to resources is represented as a vector $(x_1, \ldots, x_n)$ in which x_i denotes the number of tasks that are assigned to resource i. In the remainder of this paper, $[n]$ denotes $\{1, \ldots, n\}$. The assignment is an NE if for all $i \in [n]$ and $j \in [n]$, $|x_i - x_j| \le 1$. We study a distributed process for constructing an NE. The states of the process, $X(0), X(1), \ldots,$ are assignments. The transition from state $X(t) = (X_1(t), \ldots, X_n(t))$ to state $X(t+1)$ is given by the greedy distributed protocol in Figure 2.1.

```
For each task $b$ do in parallel
  Let $i_b$ be the current resource of task $b$
  Choose resource $j_b$ uniformly at random
  Let $X_{i_b}(t)$ be the current load of resource $i$
  Let $X_{j_b}(t)$ be the current load of resource $j$
  If $X_{i_b}(t) > X_{j_b}(t) + 1$ then
    Move task $b$ from resource $i_b$ to $j_b$ with probability $1 - X_{j_b}(t)/X_{i_b}(t)$
```

FIG. 2.1. *The modified protocol, with "neutral moves" disallowed.*

Note that if $X(t)$ is an NE, then $X(t+1) = X(t)$ so the assignment stops changing. Here is a formal description of the transition from a state $X(t) = x$. Independently, for every $i \in [n]$, let $(Y_{i,1}(x), \ldots, Y_{i,n}(x))$ be a random variable drawn from a multinomial distribution with the constraint $\sum_{j=1}^{n} Y_{i,j}(x) = x_i$. ($Y_{ij}$ represents the number of migrations from i to j in a round.) The corresponding probabilities $(p_{i,1}(x), \ldots, p_{i,n}(x))$ are given by

$$p_{i,j}(x) = \begin{cases} \frac{1}{n}\left(1 - \frac{x_j}{x_i}\right) & \text{if } x_i > x_j + 1, \\ 0 & \text{if } i \neq j \text{ but } x_i \le x_j + 1, \\ 1 - \sum_{j \neq i} p_{i,j}(x) & \text{if } i = j. \end{cases}$$

Then $X_i(t+1) = \sum_{\ell=1}^{n} Y_{\ell,i}(x)$.

For any assignment $x = (x_1, \ldots, x_n)$, let $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$. We define the potential function $\Phi(x) = \sum_{i=1}^{n} (x_i - \overline{x})^2$. Note that $\Phi(x) = \sum_{i=1}^{n} x_i^2 - n\overline{x}^2$ and that a single selfish move reduces the potential.

3. Upper bound on convergence time. Our main result is the following.

THEOREM 3.1. *Let T be the number of rounds taken by the protocol of Figure* 2.1 *to reach an NE for the first time. Then* $\mathbb{E}[T] = O(\log\log m + n^4)$.

The proof of this theorem proceeds as follows. First (Lemma 3.6) we give an upper bound on $\mathbb{E}[\Phi(X(t))]$ which implies (Corollary 3.10) that there is a $\tau = O(\log\log m)$ such that, with high probability, $\Phi(X(\tau)) = O(n)$. We also show (Observation 3.5 and Corollary 3.14) that $\Phi(X(t))$ is a supermartingale and (Lemma 3.15) that it has enough variance. Using these facts, we obtain the upper bound on the convergence time.

DEFINITION. *Let* $S_i(x) = \{j \mid x_j < x_i - 1\}$. $S_i(x)$ *is the set of resources that are significantly smaller than resource* i *in state* x *(in the sense that their loads are at least two tasks smaller than the load of resource* i*). Similarly, let* $L_i(x) = \{j \mid x_j > x_i + 1\}$ *and let* $d_i(x) = \frac{1}{n}\sum_{j:|x_i - x_j|\le 1}(x_i - x_j)$.

OBSERVATION 3.2. $\mathbb{E}[X_i(t+1) \mid X(t) = x] = \overline{x} + d_i(x)$.

Proof.

$$\begin{aligned}
\mathbb{E}[X_i(t+1) \mid X(t) = x] &= \sum_{\ell=1}^{n} \mathbb{E}[Y_{\ell,i}(x)] = \sum_{\ell=1}^{n} x_\ell p_{\ell,i}(x)\\
&= \sum_{\ell\in L_i(x)} x_\ell \frac{1}{n}\left(1 - \frac{x_i}{x_\ell}\right) + x_i\left(1 - \sum_{j\in S_i(x)} \frac{1}{n}\left(1 - \frac{x_j}{x_i}\right)\right)\\
&= x_i + \frac{1}{n}\left(\sum_{\ell\in L_i(x)} (x_\ell - x_i) - \sum_{j\in S_i(x)} (x_i - x_j)\right)\\
&= x_i + \frac{1}{n}\sum_{\ell\in L_i(x)\cup S_i(x)} (x_\ell - x_i)\\
&= x_i + \frac{1}{n}\sum_{\ell=1}^{n}(x_\ell - x_i) - \frac{1}{n}\sum_{\ell\notin L_i(x)\cup S_i(x)} (x_\ell - x_i)\\
&= \overline{x} - \frac{1}{n}\sum_{\ell\notin L_i(x)\cup S_i(x)} (x_\ell - x_i)\\
&= \overline{x} + \frac{1}{n}\sum_{\ell\notin L_i(x)\cup S_i(x)} (x_i - x_\ell). \qquad \square
\end{aligned}$$

OBSERVATION 3.3. $\sum_{i=1}^{n}(\mathbb{E}[X_i(t+1) \mid X(t) = x])^2 = n\overline{x}^2 + \sum_{i=1}^{n} d_i(x)^2$.

Proof. Using Observation 3.2,

$$\sum_{i=1}^{n}(\mathbb{E}[X_i(t+1) \mid X(t) = x])^2 = \sum_{i=1}^{n}(\overline{x} + d_i(x))^2 = n\overline{x}^2 + 2\overline{x}\sum_{i=1}^{n} d_i(x) + \sum_{i=1}^{n} d_i(x)^2,$$

and the second term is zero since $d_i(x) = \mathbb{E}[X_i(t+1) \mid X(t) = x] - \overline{x}$. $\square$

OBSERVATION 3.4. $\mathrm{var}[X_i(t+1) \mid X(t) = x] \le \frac{1}{n}\sum_{\ell\in L_i(x)}(x_\ell - x_i) + \frac{1}{n}\sum_{j\in S_i(x)}(x_i - x_j)$.

Proof.

$$\begin{aligned}
\mathrm{var}(X_i(t+1) \mid X(t) = x) &= \sum_{\ell=1}^{n}\mathrm{var}(Y_{\ell,i}(x)) = \sum_{\ell=1}^{n} x_\ell p_{\ell,i}(x)(1 - p_{\ell,i}(x))\\
&= \sum_{\ell\in L_i(x)} x_\ell \frac{1}{n}\left(1 - \frac{x_i}{x_\ell}\right)(1 - p_{\ell,i}(x))
\end{aligned}$$

$$+ x_i p_{i,i}(x) \left(\sum_{j \in S_i(x)} \frac{1}{n} \left(1 - \frac{x_j}{x_i}\right) \right)$$

$$= \frac{1}{n} \sum_{\ell \in L_i(x)} (x_\ell - x_i)(1 - p_{\ell,i}(x)) + p_{i,i}(x) \frac{1}{n} \sum_{j \in S_i(x)} (x_i - x_j)$$

$$\leq \frac{1}{n} \sum_{\ell \in L_i(x)} (x_\ell - x_i) + \frac{1}{n} \sum_{j \in S_i(x)} (x_i - x_j). \qquad \square$$

Definition. *For any assignment x, let $s_i(x) = |\{j \mid x_j = x_i - 1\}|$ and $l_i(x) = |\{j \mid x_j = x_i + 1\}|$. Let $u_1(x) = \sum_{i=1}^n \sum_{j \in [n]: |x_i - x_j| > 1} |x_i - x_j|$ and $u_2(x) = \sum_{i=1}^n (s_i(x) - l_i(x))^2$. Let $u(x) = u_1(x)/n + u_2(x)/n^2$. We will show that $u(x)$ is an upper bound on the expected potential after one step, starting from state x. The quantity $u_1(x)$ corresponds to the contribution arising from the sum of the variances of the individual loads, and $u_2(x)$ corresponds to the rest.*

Observation 3.5. $\mathbb{E}[\Phi(X(t+1)) \mid X(t) = x] \leq u(x)$.

Proof.

$$\begin{aligned} \mathbb{E}[\Phi(X(t+1)) \mid X(t) = x] + n\overline{x}^2 &= \sum_{i=1}^n \mathbb{E}[X_i(t+1)^2 \mid X(t) = x] \\ &= \sum_{i=1}^n (\mathbb{E}[X_i(t+1) \mid X(t) = x])^2 \\ &\quad + \sum_{i=1}^n \operatorname{var}(X_i(t+1) \mid X(t) = x). \end{aligned}$$

Using Observations 3.3 and 3.4, this is at most $n\overline{x}^2 + \sum_{i=1}^n d_i(x)^2 + u_1(x)/n$. But

$$d_i(x) = \frac{1}{n} \sum_{j: |x_i - x_j| \leq 1} (x_i - x_j) = \frac{1}{n}(s_i(x) - \ell_i(x)),$$

so the result follows. $\square$

Lemma 3.6. $\mathbb{E}[\Phi(X(t+1)) \mid X(t) = x] \leq n + 2n^{1/2}\Phi(x)^{1/2}$.

Proof. In the proof of Observation 3.5, we established that $\mathbb{E}[\Phi(X(t+1)) \mid X(t) = x] \leq \sum_{i=1}^n d_i(x)^2 + u_1(x)/n$. Upper-bounding $u_1(x)$ and using $d_i(x) \leq 1$, we have

$$\mathbb{E}[\Phi(X(t+1)) \mid X(t) = x] \leq n + \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^n |x_i - x_j|,$$

and since $|x_i - x_j| \leq |x_i - \overline{x}| + |x_j - \overline{x}|$, this is at most $n + 2\sum_{i=1}^n |x_i - \overline{x}|$. By Cauchy–Schwarz, $(\sum_i |x_i - \overline{x}| \cdot 1)^2 \leq \sum_i |x_i - \overline{x}|^2 \sum_i 1$; thus

$$\mathbb{E}[\Phi(X(t+1)) \mid X(t) = x] \leq n + 2\left(n \sum_{i=1}^n |x_i - \overline{x}|^2\right)^{1/2}. \qquad \square$$

Corollary 3.7. $\mathbb{E}[\Phi(X(t+1))] \leq n + 2n^{1/2}(\mathbb{E}[\Phi(X(t))])^{1/2}$.

Proof. Using Lemma 3.6, $\mathbb{E}[\Phi(X(t+1))] \leq n + 2n^{1/2}\mathbb{E}[f^{1/2}]$, where f denotes the random variable $\Phi(X(t))$. By Jensen's inequality $\mathbb{E}[f^{1/2}] \leq (\mathbb{E}[f])^{1/2}$ since the square-root function is concave, so we get $\mathbb{E}[\Phi(X(t+1))] \leq n + 2n^{1/2}(\mathbb{E}[f])^{1/2}$. $\square$

LEMMA 3.8. *Either there is a $t' < t$ such that $\mathbb{E}[\Phi(X(t'))] \le 18n$ or $\mathbb{E}[\Phi(X(t))] \le 9^{1-2^{-t}} n^{1-2^{-t}} \Phi(X(0))^{2^{-t}}$.*

Proof. The proof is by induction on t. The base case is $t = 0$. For the inductive step, note that $1 - 2^{-t} = \sum_{k=1}^{t} 2^{-k}$. Suppose that for all $t' < t$, $\mathbb{E}[\Phi(X(t'))] > 18n$ (otherwise we are finished). Then by Corollary 3.7,

$$\mathbb{E}[\Phi(X(t))] = n + 2n^{1/2}(\mathbb{E}[\Phi(X(t-1))])^{1/2} \le 3n^{1/2}(\mathbb{E}[\Phi(X(t-1))])^{1/2}.$$

Applying the inductive hypothesis,

$$\mathbb{E}[\Phi(X(t))] \le 3n^{1/2}\left(3^{2(1-2^{-(t-1)})} n^{1-2^{-(t-1)}} \Phi(X(0))^{2^{-(t-1)}}\right)^{1/2}. \qquad \square$$

COROLLARY 3.9. *There is a $\tau \le \lceil \lg \lg \Phi(X(0)) \rceil$ such that $\mathbb{E}[\Phi(X(\tau))] \le 18n$.*

Proof. Take $t = \lceil \lg \lg \Phi(X(0)) \rceil$. Either there is a $\tau < t$ with $\mathbb{E}[\Phi(X(\tau))] \le 18n$ or, by the lemma,

$$\mathbb{E}[\Phi(X(t))] \le 9n\Phi(X(0))^{2^{-t}} \le 18n. \qquad \square$$

COROLLARY 3.10. *There is a $\tau \le \lceil \lg \lg \Phi(X(0)) \rceil$ such that $\Pr(\Phi(X(\tau)) > 720n) \le 1/40$.*

Proof. Consider the (nonnegative) random variable $Y = \Phi(X(\tau))$, where τ is the quantity from Corollary 3.9. Markov's inequality says that for any $a > 0$, $\Pr(Y \ge a) \le \mathbb{E}[Y]/a$. Now use Corollary 3.9 with $a = 720n$. □

COROLLARY 3.11. *For all $\epsilon > 0$, provided that $n < m^{1/3}$, the expected time to reach an ϵ-NE is $O(\log \log m)$.*

Proof. Since the bound is asymptotic as a function of m for fixed ϵ, we can assume without loss of generality that $m > (60/\epsilon)^2$ and that $\epsilon m/(2n)$ is an integer. We show that for any starting assignment $X(0)$, there exists $\tau \le \log \log(m^2)$ such that $\Pr(X(\tau)$ is ϵ-Nash$) > \frac{39}{40}$. This implies the statement of the result since the number of blocks of τ steps needed to reach an ϵ-NE is at most

$$1 + \left(\frac{1}{40}\right) + \left(\frac{1}{40}\right)^2 + \cdots = \frac{40}{39} < 2.$$

Suppose assignment x is not ϵ-Nash. If $X(t) = x$, there exist resources i, j with $X_i(t) - X_j(t) > \epsilon m/n$. We use the following notation. Let $\Delta = \epsilon m/(2n)$. Let $\beta = X_i(t) - X_j(t) - 2\Delta$. Note $\beta > 0$. If $X(t+1)$ is obtained from $X(t)$ by transferring Δ tasks from i to j, then

$$\begin{aligned}
&\Phi(X(t)) - \Phi(X(t+1)) \\
&= X_i(t)^2 + X_j(t)^2 - X_i(t+1)^2 - X_j(t+1)^2 \\
&= (2\Delta + \beta + X_j(t))^2 + X_j(t)^2 - (\Delta + \beta + X_j(t))^2 - (\Delta + X_j(t))^2 \\
&= 2\Delta(\Delta + \beta + X_j(t)) + \Delta^2 - \left(2\Delta X_j(t) + \Delta^2\right) \\
&= 2\Delta(\Delta + \beta) \ge \Delta^2 = (\epsilon m/2n)^2.
\end{aligned}$$

It follows that $\Phi(X(t)) \ge (\epsilon m/2n)^2$. From Corollary 3.10, $\Pr(\Phi(X(\tau)) < 720n) > \frac{39}{40}$, for $\tau = \log \log(\Phi(0)) = O(\log \log m)$.

An assignment $X(\tau)$ with $\Phi(X(\tau)) \le 720n$ must be ϵ-Nash if $(\epsilon m/2n)^2 > 720n$. Note that $m > n^3$ and $m > (60/\epsilon)^2$. Hence, from $\epsilon^2(60/\epsilon)^2 n^3 > 4.720.n^3$, we can deduce $\epsilon^2 m^2 > 4.720.n^3$; hence $(\epsilon m/2n)^2 > 720n$. □

Corollary 3.10 tells us that $\Phi(X(\tau))$ is likely to be $O(n)$. We want to show that $\Phi(X(t))$ quickly gets even smaller (all the way to an NE) and to this end, we show that $\Phi(X(t))$ is a supermartingale. By Observation 3.5, it suffices to show $u(x) \le \Phi(x)$, and we proceed with this. In the following, we shall consider the cases $|x_i - \overline{x}| < 2.5$ for all $i \in [n]$ (Lemma 3.12) and $\exists i \in [n] : |x_i - \overline{x}| \ge 2.5$ (Lemma 3.13) separately.

Lemma 3.12. *Suppose that assignment $x = (x_1, \ldots, x_n)$ satisfies $|x_i - \overline{x}| < 2.5$ for all $i \in [n]$. Then $u(x) \le \Phi(x)$.*

Proof. For all $i \in [n]$ and $j \in [n]$ we have $|x_i - x_j| \le |x_i - \overline{x}| + |x_j - \overline{x}| < 5$. Let $z = \min_i x_i$ so that every $x_i \in \{z, \ldots, z+4\}$. Let $n_i = |\{j \mid x_j = z + i\}|$. Then

$$n^2\Phi(x) = n^2 \sum_{i=1}^{n} x_i^2 - n\left(\sum_{i=1}^{n} x_i\right)^2 = n^2 \left(\sum_{j=0}^{4} n_j (z+j)^2\right) - \left(\sum_{j=0}^{4} n_j(z+j)\right)^2.$$

Also, $n^2 u(x) = n u_1(x) + u_2(x)$, where

$$\begin{aligned} u_1(x) &= n_0(2n_2 + 3n_3 + 4n_4) + n_1(2n_3 + 3n_4) \\ &\quad + n_2(2n_0 + 2n_4) + n_3(3n_0 + 2n_1) + n_4(4n_0 + 3n_1 + 2n_2) \end{aligned}$$

and

$$u_2(x) = n_0 n_1^2 + n_1(n_0 - n_2)^2 + n_2(n_1 - n_3)^2 + n_3(n_2 - n_4)^2 + n_4 n_3^2.$$

Plugging in these expressions and simplifying, we get

$$\begin{aligned} &n^2\Phi(x) - n^2 u(x) \\ &= 4n_0n_1n_2 + 3n_0^2n_3 + 4n_0n_1n_3 + 4n_0n_2n_3 + 4n_1n_2n_3 + 3n_0n_3^2 \\ &\quad + 8n_0^2n_4 + 12n_0n_1n_4 + 3n_1^2n_4 + 8n_0n_2n_4 + 4n_1n_2n_4 + 12n_0n_3n_4 \\ &\quad + 4n_1n_3n_4 + 4n_2n_3n_4 + 8n_0n_4^2 + 3n_1n_4^2, \end{aligned}$$

which is clearly nonnegative since all coefficients are positive. □

Lemma 3.13. *Suppose that assignment $x = (x_1, \ldots, x_n)$ satisfies $|x_n - \overline{x}| \ge 2.5$ and, for all $i \in [n]$, $|x_i - \overline{x}| \le |x_n - \overline{x}|$. Let $w = (w_1, \ldots, w_{n-1})$ be the assignment with $w_i = x_i$ for $i \in [n-1]$. Then $\Phi(x) - u(x) \ge \Phi(w) - u(w)$; that is, the lower bound on the potential drop for x is at least as big as that for w.*

Proof. Let $k = |x_n - \overline{x}|$. We will show that

(1) $\Phi(x) - \Phi(w) \ge k^2$ and

(2) $u(x) - u(w) \le 2k + 1$.

Then

$$\Phi(x) - u(x) - (\Phi(w) - u(w)) \ge k^2 - (2k+1),$$

which is nonnegative since $k \ge 2.5 \ge 1 + \sqrt{2}$.

First, we prove (1). Let $f(z) = \sum_{i=1}^{n-1}(x_i - z)^2$. Note that the derivative of $f(z)$ is

$$f'(z) = 2(n-1)z - 2\sum_{i=1}^{n-1} x_i = 2(n-1)z - 2(n-1)\overline{w}.$$

Furthermore, the second derivative is $f''(z) = 2(n-1) \ge 0$. Thus, $f(z)$ is minimized at $z = \overline{w}$. Now note that

$$\Phi(x) - \Phi(w) = k^2 + \sum_{i=1}^{n-1}(x_i - \overline{x})^2 - \sum_{i=1}^{n-1}(x_i - \overline{w})^2 \ge k^2.$$

Now we finish the proof by proving (2). Assume first that $x_n = \overline{x} + k$. Then

$$u_1(x) - u_1(w) = 2 \sum_{i \in [n]: |x_i - x_n| > 1} |x_i - x_n| \leq 2 \sum_{i=1}^{n} |x_i - x_n| = 2 \sum_{i=1}^{n} (x_n - x_i) = 2nk.$$

Let $z_j = |\{\ell \mid x_\ell = j\}|$. Clearly $z_j = 0$ for $j > x_n$. Let $\xi = \lceil x_n - 2k \rceil$. For $\ell \in [n]$ we have $x_\ell \geq \overline{x} - k = x_n - 2k$, so $z_j = 0$ for $j < \xi$. Now $u_2(x) = \sum_{j=\xi}^{x_n} z_j(z_{j-1} - z_{j+1})^2$. The representation of w in terms of z_js is the same as the representation of x except that z_{x_n} is reduced by one. Therefore,

$$\begin{aligned} u_2(x) - u_2(w) &= z_{x_n - 1}\left((z_{x_n - 2} - z_{x_n})^2 - (z_{x_n - 2} - z_{x_n} + 1)^2\right) + (z_{x_n - 1} - z_{x_n + 1})^2 \\ &= z_{x_n - 1}(-2z_{x_n - 2} + 2z_{x_n} + z_{x_n - 1} - 1) \leq z_{x_n - 1}(2z_{x_n} + z_{x_n - 1}). \end{aligned}$$

But since $z_{x_n} \leq n - z_{x_n - 1}$, the upper bound on the right-hand side is at most

$$z_{x_n - 1}(2n - 2z_{x_n - 1} + z_{x_n - 1}) = 2z_{x_n - 1}(n - z_{x_n - 1}/2),$$

which is at most n^2 since the right-hand side is maximized at $z_{x_n - 1} = n$. To finish the proof of (2), use the definition of u to deduce that

$$u(x) - u(w) \leq \frac{u_1(x) - u_1(w)}{n} + \frac{u_2(x) - u_2(w)}{n^2}.$$

The proof of (2) when $x_n = \overline{x} - k$ is similar. □

COROLLARY 3.14. *For any assignment $x = (x_1, \ldots, x_n)$, $\Phi(x) - u(x) \geq 0$.*

Proof. The proof is by induction on n. The base case, $n = 1$, follows from Lemma 3.12. Suppose $n > 1$. Neither $\Phi(x)$ nor $u(x)$ depends upon the order of the components in x, so assume without loss of generality that $|x_i - \overline{x}| \leq |x_n - \overline{x}|$ for all i. If $|x_n - \overline{x}| < 2.5$, then apply Lemma 3.12. Otherwise, use Lemma 3.13 to find an assignment $w = (w_1, \ldots, w_{n-1})$ such that $\Phi(x) - u(x) \geq \Phi(w) - u(w)$. By the inductive hypothesis, $\Phi(w) - u(w) \geq 0$. □

Together, Observation 3.5 and Corollary 3.14 tell us that $\mathbb{E}[\Phi(X(t+1)) \mid X(t) = x] \leq \Phi(x)$. The next lemma will be used to give a lower bound on the variance of the process. Let $V = 0.4n^{-2}$.

LEMMA 3.15. *Suppose that $X(t) = x$ and that x is not an NE. Then*

$$\Pr(\Phi(X(t+1)) \neq \Phi(x) \mid X(t) = x) \geq V.$$

Proof. Choose s and ℓ such that for all $i \in [n]$, $x_s \leq x_i \leq x_\ell$. Since x is not an NE, $x_\ell > x_s + 1$. Assuming $X(t) = x$, consider the following experiment for choosing $X(t+1)$.

The intuition behind the experiment is as follows. We wish to show that the transition from $X(t)$ to $X(t+1)$ has some variance in the sense that $\Phi(X(t+1))$ is sufficiently likely to differ from $\Phi(X(t))$. To do this, we single out a "least loaded" resource s and a "most loaded" resource ℓ as above. In the transition from $X(t)$ to $X(t+1)$ we make transitions from resources other than resource ℓ in the usual way. We pay special attention to transitions from resource ℓ (and particular attention to transitions from resource ℓ which could either go to resource s or stay at resource ℓ). It helps to be very precise about how the random decisions involving tasks that start at resource ℓ are made. In particular, for each task b that starts at resource ℓ, we first make a decision about whether b would *accept* the transition from resource ℓ to resource s *if* b *happened to choose resource* s. Then we make the decision about which resource task b should choose. Of course, we cannot cheat and we have to sample from the original required distribution. Here are the details.

Independently, for every $i \neq \ell$, choose $(Y_{i,1}(x), \ldots, Y_{i,n}(x))$ from the multinomial distribution described in section 2. (In the informal description above, this corresponds to making transitions from resources other than resource ℓ in the usual way.) Now, for every task $b \in x_\ell$, let $z_b = 1$ with probability $1 - x_s/x_\ell$ and $z_b = 0$ otherwise. (In the informal description above, this corresponds to deciding whether b would *accept* the transition to s if resource s were (later) chosen.) Let x_ℓ^+ be the number of tasks b with $z_b = 1$ and let x_ℓ^- be the number of tasks b with $z_b = 0$. Choose $(Y_{\ell,1}^+(x), \ldots, Y_{\ell,n}^+(x))$ from a multinomial distribution with the constraint $\sum_{j=1}^n Y_{\ell,j}^+(x) = x_\ell^+$ and probabilities given by

$$p_{\ell,j}^+(x) = \begin{cases} \frac{1}{n} & \text{if } j = s, \\ \frac{1}{n}\left(1 - \frac{x_j}{x_\ell}\right) & \text{if } j \neq s \text{ and } x_\ell > x_j + 1, \\ 0 & \text{if } \ell \neq j \text{ but } x_\ell \leq x_j + 1, \\ 1 - \sum_{j \neq \ell} p_{\ell,j}(x) & \text{if } \ell = j. \end{cases}$$

Similarly, choose $(Y_{\ell,1}^-(x), \ldots, Y_{\ell,n}^-(x))$ from a multinomial distribution with the constraint $\sum_{j=1}^n Y_{\ell,j}^-(x) = x_\ell^-$ and probabilities given by

$$p_{\ell,j}^-(x) = \begin{cases} 0 & \text{if } j = s, \\ \frac{1}{n}\left(1 - \frac{x_j}{x_\ell}\right) & \text{if } j \neq s \text{ and } x_\ell > x_j + 1, \\ 0 & \text{if } \ell \neq j \text{ but } x_\ell \leq x_j + 1, \\ 1 - \sum_{j \neq \ell} p_{\ell,j}(x) & \text{if } \ell = j. \end{cases}$$

For all j, let $Y_{\ell,j}(x) = Y_{\ell,j}^+(x) + Y_{\ell,j}^-(x)$. Informally, the $p_{\ell,j}^+$ transition probabilities are set up so that packets which decided that they would accept a transition to s behave appropriately, and the $p_{\ell,j}^-$ transition probabilities are set up so that packets which decided that they would *not* accept a transition to s behave appropriately. By combining the probabilities, we see that $X(t{+}1)$ is chosen from the correct distribution in this way.

Now, consider the transition from x to $X(t+1)$. Condition on the choice for $(Y_{i,1}(x), \ldots, Y_{i,n}(x))$ for all $i \neq \ell$. Suppose $x_\ell^+ > 2$. Condition on the choice for $(Y_{\ell,1}^-(x), \ldots, Y_{\ell,n}^-(x))$. Flip a coin for each of the first $x_b^+ - 2$ tasks with $z_b = 1$ to determine which of $Y_{\ell,1}^+(x), \ldots, Y_{\ell,n}^+(x)$ the task contributes to. Condition on these choices. Consider the following options:

(1) Let x_1 be the resulting value of $X(t+1)$ when we add both of the last two tasks to $Y_{\ell,\ell}^+(x)$.

(2) Let x_2 be the resulting value of $X(t+1)$ when we add one of the last two tasks to $Y_{\ell,\ell}^+(x)$ and the other to $Y_{\ell,s}^+(x)$.

(3) Let x_3 be the resulting value of $X(t+1)$ when we add both of the last two tasks to $Y_{s,s}^+(x)$.

Note that, given the conditioning, each of these choices occurs with probability at least n^{-2}. Also, $\Phi(x_1)$, $\Phi(x_2)$, and $\Phi(x_3)$ are not all the same. Thus, $\Pr(\Phi(X(t{+}1) \neq \Phi(x) \mid X(t) = x, x_\ell^+ > 2) \geq n^{-2}$. Also,

$$\Pr(x_\ell^+ > 2) = 1 - \left(\frac{x_s}{x_\ell}\right)^{x_\ell} - x_\ell \left(1 - \frac{x_s}{x_\ell}\right)\left(\frac{x_s}{x_\ell}\right)^{x_\ell - 1}.$$

Since the derivative with respect to x_s is negative, this is minimized by taking x_s as large as possible, namely $x_\ell - 2$; thus $\Pr(x_\ell^+ > 2) \geq 1 - 7e^{-2} \geq 0.4$, and the result follows. □

In order to finish our proof of convergence, we need the following observation about $\Phi(x)$.

OBSERVATION 3.16. *For any assignment x, $\Phi(x) \leq m^2$. Let $r = m \bmod n$. Then $\Phi(x) \geq r(1 - r/n)$, with equality if and only if x is an NE.*

Proof. Suppose that in assignment x there are resources i and j such that $x_i - x_j \geq 2$. Let x' be the assignment constructed from x by transferring a task from resource i to resource j. Then

$$\begin{aligned}\Phi(x) - \Phi(x') &= x_i^2 - {x_i'}^2 + x_j^2 - {x_j'}^2 = x_i^2 - (x_i^2 - 2x_i + 1) + x_j^2 - (x_j^2 + 2x_j + 1)\\ &= 2x_i - 2x_j - 2 = 2(x_i - x_j) - 2 > 0.\end{aligned}$$

Now suppose that, in some assignment x', resources i and j satisfy $x_i' \geq x_j' > 0$. Let x be the assignment constructed from x' by transferring a task from resource j to resource i. Since $(x_i' + 1) - (x_j' - 1) \geq 2$, the above argument gives $\Phi(x) > \Phi(x')$. We conclude that an assignment x with maximum $\Phi(x)$ must have all of the tasks in the same resource, with $\Phi(x) = m^2$.

Furthermore, an assignment x with minimum $\Phi(x)$ must have $|x_i - x_j| \leq 1$ for all i, j. In this case there must be r resources with loads of $q + 1$ and $n - r$ resources with loads of q, where $m = qn + r$. So

$$\Phi(x) = r(q + 1 - \bar{x})^2 + (n - r)(q - \bar{x})^2 = r\left(1 - \frac{r}{n}\right)^2 + (n - r)\left(\frac{r}{n}\right)^2 = r\left(1 - \frac{r}{n}\right).$$

Note that x is a Nash assignment if and only if $|x_i - x_j| \leq 1$ for all i and j. □

Combining Observation 3.16 and Corollary 3.10, we find that there is a $\tau \leq \lceil \lg \lg m^2 \rceil$ such that $\Pr(\Phi(X(\tau)) > 720n) \leq 1/40$. Let $B = 7200n + \left\lceil \frac{m^2}{n} \right\rceil - \frac{m^2}{n}$. Let $t' = \tau + \lceil 10B^2/V \rceil$.

LEMMA 3.17. *Given any starting state $X(0) = x$, the probability that $X(t')$ is an NE is at least 3/4.*

Proof. The proof is based on a standard martingale argument; see [19]. Suppose that $\Phi(X(\tau)) \leq 720n$. Let $W_t = \Phi(X(t + \tau)) - r(1 - r/n)$ and let $D_t = \min(W_t, B)$. Note that $D_0 \leq 720n$. Together, Observation 3.5 and Corollary 3.14 tell us that W_t is a supermartingale. This implies that D_t is also a supermartingale since

$$\mathbb{E}[D_{t+1} \mid D_t = x < B] \leq \mathbb{E}[W_{t+1} \mid W_t = x < B] \leq W_t = D_t,$$

and

$$\mathbb{E}[D_{t+1} \mid D_t = B] \leq B = D_t.$$

Together, Lemma 3.15 and Observation 3.16 tell us that if $x > 0$, $\Pr(W_{t+1} \neq W_t \mid W_t = x) \geq V$. Thus, if $0 < x < B$,

$$\begin{aligned}\Pr(D_{t+1} \neq D_t \mid D_t = x) &= \Pr(\min(W_{t+1}, B) \neq W_t \mid W_t = x)\\ &\geq \Pr(W_{t+1} \neq W_t \wedge B \neq W_t \mid W_t = x)\\ &= \Pr(W_{t+1} \neq W_t \mid W_t = x) \geq V.\end{aligned}$$

Since $D_{t+1} - D_t$ is an integer, $\mathbb{E}[(D_{t+1} - D_t)^2 \mid 0 < D_t < B] \geq V$. Let T be the first time at which either (a) $D_t = 0$ (i.e., $X(t + \tau)$ is an NE), or (b) $D_t = B$.

Note that T is a stopping time. Define $Z_t = (B - D_t)^2 - Vt$, and observe that $Z_{t \wedge T}$ is a submartingale, where $t \wedge T$ denotes the minimum of t and T. Let p be the probability that (a) occurs. By the optional stopping theorem $\mathbb{E}[D_T] \leq D_0$; thus $(1-p)B = \mathbb{E}[D_T] \leq D_0$ and $p \geq 1 - D_0/B \geq \frac{9}{10}$. Also, by the optional stopping theorem

$$pB^2 - V\mathbb{E}[T] = \mathbb{E}[(B - D_T)^2] - V\mathbb{E}[T] = \mathbb{E}[Z_T] \geq Z_0 = (B - D_0)^2 > 0,$$

and thus $\mathbb{E}[T] \leq pB^2/V$. Conditioning on the occurrence of (a), it follows that $\mathbb{E}[T \mid D_T = 0] \leq B^2/V$. Hence $\Pr(T > 10B^2/V \mid D_T = 0) \leq \frac{1}{10}$. So, if we now run for $10B^2/V$ steps, then the probability that we do not reach an NE is at most $\frac{1}{40} + 2 \cdot \frac{1}{10} < 1/4$. □

Now we can give the proof of Theorem 3.1.

Proof. Subdivide time into intervals of t' steps. The probability that the process has not reached an NE before the $(j+1)$st interval is at most $(1/4)^{-j}$. □

4. Lower bounds. In this section we prove the lower bound results stated in the introduction. We will use the following Chernoff bound which can be found, for example, in [15]. Let $N \geq 1$ and let $p_i \in [0,1]$ for $i = 1, \dots, N$. Let $X_1, X_2, \dots, X_N$ be independent Bernoulli random variables with $\Pr(X_i = 1) = p_i$ for $i = 1, \dots, N$ and let $X = X_1 + \cdots + X_N$. Then we have $\mathbb{E}[X] = \sum_{i=1}^N p_i$ and for $0 \leq \epsilon \leq 1$,

$$\Pr(X \leq (1-\epsilon) \cdot \mathbb{E}[X]) \leq \exp\left(-\frac{\epsilon^2 \cdot \mathbb{E}[X]}{3}\right). \tag{4.1}$$

The following theorem gives an exponential lower bound for the expected convergence time of the process in Figure 1.1.

THEOREM 4.1. *Let $X(t)$ be the process in Figure 1.1 with $m = n$. Let $X(0)$ be the assignment given by $X(0) = (n, 0, \dots, 0)$. Let T be the first time at which $X(t)$ is an NE. Then $\mathbb{E}[T] = \exp(\Theta(\sqrt{n}))$.*

Proof. For an assignment x, let $n_0(x)$ denote the number of resources i with $x_i = 0$. Thus, $n_0(X(0)) = n - 1$. The (unique) NE x assigns one task to each resource; thus $n_0(x) = 0$. Let $k = \lfloor \sqrt{n} \rfloor$. We will show that for any assignment x with $n_0(x) \geq k$,

$$\Pr(n_0(X(t)) < k \mid X(t-1) = x) \leq \exp(-\Theta(\sqrt{n})).$$

This implies the result.

Suppose $X(t-1) = x$ with $n_0(x) \geq k$. For convenience, let n_0 denote $n_0(x)$. Let x' denote $X(t)$, and let n_0' denote $n_0(x')$. We will show that, with probability at least $1 - \exp(-\Theta(\sqrt{n}))$, $n_0' \geq k$. During the course of the proof, we will assume, where necessary, that n is sufficiently large. This is without loss of generality given the Θ notation in the statement of the result.

Case 1. $n_0 > 8k$.

Consider the protocol in Figure 1.1. Let $U = \{b \mid x_{j_b} = 0\}$. $\mathbb{E}[|U|] = n_0$, so by the Chernoff bound (4.1), $\Pr(|U| \leq \lceil \frac{n_0}{2} \rceil + \lceil \frac{3n_0}{8} \rceil) \leq \Pr(|U| \leq \frac{8}{9} n_0) = \exp\left(-\Theta(\sqrt{n})\right)$. Thus, $|U| \geq \lceil n_0/2 \rceil + \lceil 3n_0/8 \rceil$ with probability at least $1 - \exp(-\Theta(\sqrt{n}))$. Suppose this is the case. Partition U into U_1 and U_2 with $|U_1| = \lceil n_0/2 \rceil$. Let $W = \cup_{b \in U_1} \{j_b\}$. First, suppose $|W| \leq \frac{3}{8} n_0$. In that case

$$|\{j \mid x_j' > 0\}| \leq n - |U_1| + \frac{3}{8} n_0 = n - \lceil n_0/2 \rceil + \frac{3}{8} n_0 \leq n - k,$$

so $n'_0 \geq k$. Otherwise, let $U' = \{b \in U_2 \mid j_b \in W\}$. Since

$$\mathbb{E}[|U'|] = |U_2|\frac{|W|}{n_0} \geq \frac{9}{64}n_0 > \frac{9}{8}k,$$

by the Chernoff bound (4.1), $\Pr(|U'| \leq k) = \Pr(|U'| \leq (1-\frac{1}{9})\mathbb{E}[|U'|]) = \exp\left(-\Theta(\sqrt{n})\right)$, recalling that $k = \lfloor\sqrt{n}\rfloor$. Thus $|U'| \geq k$ with probability at least $1 - \exp(-\Theta(\sqrt{n}))$, which implies $n'_0 \geq k$.

Case 2. $k \leq n_0 \leq 8k$.

Consider the protocol in Figure 1.1. Let L be the set of "loners" defined by $L = \{i \mid x_i = 1\}$ and let $\ell = |L|$. The number of resources i with $x_i > 1$ is $n - n_0 - \ell$, and this is at most half as many as the number of tasks assigned to such resources (which is $n - \ell$), so $\ell \geq n - 2n_0$. Let $U = \{b \mid i_b \in L \text{ and } x_{j_b} = 0\}$. $\mathbb{E}[|U|] = \ell\frac{n_0}{n} \geq \frac{(n-2n_0)n_0}{n} = \Theta(\sqrt{n})$, so by the Chernoff bound (4.1), $\Pr(|U| \leq 2\lceil\frac{1}{4}\ell\frac{n_0}{n}\rceil) \leq \Pr(|U| \leq \frac{2}{3}\mathbb{E}[|U|]) \leq \exp\left(-\Theta(\sqrt{n})\right)$. Thus, $|U| \geq 2\lceil\frac{1}{4}\ell\frac{n_0}{n}\rceil$ with probability at least $1 - \exp(-\Theta(\sqrt{n}))$. Suppose this is the case. Let U_1 and U_2 be disjoint subsets of U of size $\lceil\frac{1}{4}\ell\frac{n_0}{n}\rceil$. Order tasks in U arbitrarily and let $S = \{b \in U \mid$ for some $b' \in U$ with $b' < b$, $j_{b'} = j_b.\}$. (Note that $|S|$ does not depend on the ordering.) Let $W = \cup_{b \in U_1}\{j_b\}$.

Note that if $|W| \leq \frac{1}{5}\ell\frac{n_0}{n}$, then $|S| \geq \frac{1}{20}\ell\frac{n_0}{n} > \frac{n_0}{40}\left(\frac{\ell}{n}\right)^2$. Otherwise, let $U' = \{b \in U_2 \mid j_b \in W\}$. Since

$$\mathbb{E}[|U'|] = |U_2|\frac{|W|}{n_0} \geq \frac{n_0}{20}\left(\frac{\ell}{n}\right)^2,$$

by the Chernoff bound (4.1), $\Pr(|U'| \leq \frac{1}{2}\frac{n_0}{20}\left(\frac{\ell}{n}\right)^2) \leq \exp\left(-\Theta(\sqrt{n})\right)$ (recall that $n_0\left(\frac{\ell}{n}\right)^2 \geq n_0\left(\frac{n-2n_0}{n}\right)^2 \geq k\left(\frac{n-16k}{n}\right)^2 = \Theta(\sqrt{n})$), and thus $|U'| \geq \frac{n_0}{40}\left(\frac{\ell}{n}\right)^2$ with probability at least $1 - \exp(-\Theta(\sqrt{n}))$; hence $|S| \geq \frac{n_0}{40}\left(\frac{\ell}{n}\right)^2$.

Suppose then that $|S| \geq \frac{n_0}{40}\left(\frac{\ell}{n}\right)^2$. Assuming that n is sufficiently large, $|S| \geq k/41$. Let $B_0 = \cup_{b \in U}\{j_b\}$ and $B_1 = \cup_{b \in L-U}\{i_b\}$. Note that every resource in $B_0 \cup B_1$ is used in x' for some task $b \in L$. Thus, $|B_0 \cup B_1| \leq \ell - |S|$. Let $R = \{i \mid x_i = 0\} \cup L - B_0 - B_1$. Then $|R| \geq n_0 + \ell - (\ell - |S|) \geq n_0 + |S| \geq (1 + \frac{1}{41})k$.

Let $T = \{b \mid i_b \notin L,\ j_b \in R\}$. $\mathbb{E}[T] = (n-\ell)\frac{|R|}{n}$, and

$$\Pr\left(T \geq \frac{|R|}{100}\right) \leq \binom{n-\ell}{\frac{|R|}{100}}\left(\frac{|R|}{n}\right)^{|R|/100} \leq \left(\frac{2n_0e100}{n}\right)^{|R|/100};$$

thus with probability at least $1 - \exp(-\Theta(\sqrt{n}))$, $T < |R|/100$. In that case, $n'_0 \geq |R|(1 - \frac{1}{100}) \geq k$. $\square$

The following theorem provides a lower bound on the expected convergence time regardless of which of the two protocols is being used.

THEOREM 4.2. *Suppose that m is even. Let $X(t)$ be the process in Figure* 2.1 *with $n = 2$. Let $X(0)$ be the assignment given by $X(0) = (m, 0)$. Let T be the first time at which $X(t)$ is an NE. Then $\mathbb{E}[T] = \Omega(\log\log m)$. The same result holds for the process in Figure* 1.1.

Proof. Note that both protocols have the same behavior since m is even, and, therefore, the situation $x_1 = x_2 + 1$ cannot arise. For concreteness, focus on the protocol in Figure 2.1.

Let $y(x) = \max_i x_i - m/2$ and let $y_t = y(X(t))$; thus $y_0 = m/2$ and, for an NE x, $y(x) = 0$. We will show that for any assignment x, $\Pr(y_{t+1} > y(x)^{1/10} \mid X(t) = x) \geq 1 - y_t^{-1/4}$. (There is nothing very special about the exact value "1/10"—this value is being used as part of an explicit "lack of concentration" inequality in the proof, noting that for a lower bound we essentially want to lower-bound the variances of the load distributions. This seems to require a somewhat ad hoc approach, in contrast with the usage of concentration inequalities.)

Suppose $X(t) = x$ is an assignment with $x_1 \geq x_2$. As we have seen in section 2, $Y_{1,2}(x)$ (the number of migrations from resource 1 to resource 2 in the round) is a binomial random variable

$$B\left(x_1, \frac{1}{2}\left(1 - \frac{x_2}{x_1}\right)\right) = B\left(\frac{m}{2} + y_t, \frac{2y_t}{m + 2y_t}\right).$$

In general, let T_t be the number of migrations from the most-loaded resource in $X(t)$ to the least-loaded resource and note that the distribution of T_t is $B\big(\frac{m}{2} + y_t, \frac{2y_t}{m+2y_t}\big)$ with mean y_t. If $T_t = y_t + \ell$ or $T_t = y_t - \ell$, then $y_{t+1} = \ell$. Thus $\Pr(y_{t+1} > y_t^{1/10}) = \Pr(|T_t - \mathbb{E}[T_t]| > y_t^{1/10})$. We continue by showing that this binomial distribution is sufficiently "spread out" in the region of its mode that we can find an upper bound on $\Pr(y_{t+1} \leq y_t^{1/10})$. This will lead to our lower bound on the expected time for $(y_t)_t$ to decrease below some constant (we use the constant 16):

$$\Pr(T_t = y_t) = \binom{\frac{1}{2}m + y_t}{y_t}\left(\frac{2y_t}{m + 2y_t}\right)^{y_t}\left(\frac{m}{m + 2y_t}\right)^{\frac{1}{2}m},$$

$$\Pr(T_t = y_t + j) = \binom{\frac{1}{2}m + y_t}{y_t + j}\Big(\frac{2y_t}{m + 2y_t}\Big)^{y_t + j}\Big(\frac{m}{m + 2y_t}\Big)^{\frac{1}{2}m - j}.$$

Suppose $j > 0$. Then

$$\begin{aligned}\frac{\Pr(T_t = y_t + j)}{\Pr(T_t = y_t)} &= \Big(\frac{2y_t}{m + 2y_t}\Big)^{j}\Big(\frac{m}{m + 2y_t}\Big)^{-j}\left(\frac{y_t!(\frac{1}{2}m)!}{(y_t + j)!(\frac{1}{2}m + y_t - (y_t + j))!}\right)\\ &= \Big(\frac{2y_t}{m}\Big)^{j}\left(\prod_{\ell=1}^{j}\frac{\frac{1}{2}m + 1 - \ell}{y_t + \ell}\right) = \Big(\frac{2y_t}{m}\Big)^{j}\left(\prod_{\ell=1}^{j}\frac{m + 2 - 2\ell}{2y_t + 2\ell}\right)\\ &> \Big(\frac{2y_t}{m}\Big)^{j}\left(\prod_{\ell=1}^{j}\frac{m - 2j}{2y_t + 2j}\right) = \Big[\Big(\frac{2y_t}{m}\Big)\Big(\frac{m - 2j}{2y_t + 2j}\Big)\Big]^{j}.\end{aligned}$$

Similarly, for $j < 0$,

$$\begin{aligned}\frac{\Pr(T_t = y_t + j)}{\Pr(T_t = y_t)} &= \Big(\frac{2y_t}{m}\Big)^{j}\left(\prod_{\ell=1}^{|j|}\frac{y_t + 1 - \ell}{\frac{1}{2}m + \ell}\right) = \Big(\frac{m}{2y_t}\Big)^{|j|}\left(\prod_{\ell=1}^{|j|}\frac{2y_t + 2 - 2\ell}{m + 2\ell}\right)\\ &> \Big(\frac{m}{2y_t}\Big)^{|j|}\Big(\frac{2y_t - 2|j|}{m + 2|j|}\Big)^{|j|} = \Big[\Big(\frac{m}{2y_t}\Big)\Big(\frac{2y_t - 2|j|}{m + 2|j|}\Big)\Big]^{|j|}\\ &= \Big[\Big(\frac{2y_t}{m}\Big)\Big(\frac{m - 2j}{2y_t + 2j}\Big)\Big]^{j}.\end{aligned}$$

Thus for all j,

$$\frac{\Pr(T_t = y_t + j)}{\Pr(T_t = y_t)} > \Big[\Big(\frac{2y_t}{m}\Big)\Big(\frac{m - 2j}{2y_t + 2j}\Big)\Big]^{j} = \Big[\Big(\frac{y_t}{y_t + j}\Big)\Big(\frac{m - 2j}{m}\Big)\Big]^{j}.$$

Thus for all j with $|j| \leq y_t^{1/4}$, where $y_t^{1/4}$ is the positive fourth root of y_t, this is at least

$$\begin{aligned}
&\left(\frac{y_t}{y_t+y_t^{1/4}}\right)^{y_t^{1/4}} \Big(\frac{m-2y_t^{1/4}}{m}\Big)^{y_t^{1/4}} \\
&\geq \left(\frac{y_t}{y_t+y_t^{1/4}}\right)^{y_t^{1/4}} \Big(\frac{2y_t-2y_t^{1/4}}{2y_t}\Big)^{y_t^{1/4}} = \left(\frac{y_t-y_t^{1/4}}{y_t+y_t^{1/4}}\right)^{y_t^{1/4}} = \left(\frac{y_t+y_t^{1/4}-2y_t^{1/4}}{y_t+y_t^{1/4}}\right)^{y_t^{1/4}} \\
&= \left(1-\frac{2y_t^{1/4}}{y_t+y_t^{1/4}}\right)^{y_t^{1/4}} \geq \left(1-\frac{2y_t^{1/4}}{y_t}\right)^{y_t^{1/4}} = \left(1-2y_t^{-3/4}\right)^{y_t^{1/4}} \\
&\geq 1-2y_t^{-3/4}y_t^{1/4} = 1-2y_t^{-1/2} \geq \frac{1}{2},
\end{aligned}$$

where the last inequality just requires $y_t \geq 16$.

Note that the mode of a binomial distribution is one or both of the integers closest to the expectation, and the distribution is monotonically decreasing as one moves away from the mode. But, for $|j| \leq y_t^{1/4}$, $\Pr(T_t = y_t + j) \geq \frac{1}{2}\Pr(T_t = y_t)$; hence $\Pr(T_t = y_t) \leq 2/(1+2y_t^{1/4})$. Since $\Pr(T_t = y_t + j) \leq \Pr(T_t = y_t)$, it follows that

$$\Pr(T_t \in [y_t - y_t^{1/10}, y_t + y_t^{1/10}]) \leq (2y_t^{1/10}+1)\Pr(T_t = y_t) < 3y_t^{-3/20}.$$

We say that the transition from y_t to y_{t+1} is a "fast round" if $y_{t+1} \leq y_t^{1/10}$ (equivalently, it is a fast round if $T_t \in [y_t - y_t^{1/10}, y_t + y_t^{1/10}]$). Otherwise it is a slow round. Recall that $y_0 = m/2$. Let

$$r = \left\lfloor \log_{10}\left(\frac{\log(y_0)}{\log(12^{20/3})}\right)\right\rfloor.$$

If the first j rounds are slow, then $y_j \geq y_0^{10^{-j}}$. If $j \leq r$, then $y_0^{10^{-j}} \geq 12^{20/3}$; thus the probability that the transition from y_j to y_{j+1} is the first fast round is at most $3\big(y_0^{10^{-j}}\big)^{-3/20} \leq 1/4$.

Also, if $j < r$, then these probabilities increase geometrically so that the ratio of the probability that the transition to y_{j+1} is the first fast round and the probability that the transition to y_j is the first fast round is

$$\frac{3\left(y_0^{10^{-(j+1)}}\right)^{-3/20}}{3\big(y_0^{10^{-j}}\big)^{-3/20}} = \left(y_0^{10^{-j}-10^{-(j+1)}}\right)^{3/20} \geq \left(y_0^{10^{-(j+1)}}\right)^{3/20} \geq 12 \geq 2;$$

thus $\sum_{j=0}^{r-1}\Pr(\text{transition from } y_j \text{ to } y_{j+1} \text{ is the first fast round}) \leq 2 \cdot 1/4 = \frac{1}{2}$. Therefore, with probability at least $1/2$, all of the first r rounds are slow. In this case, $\arg\min_t(y_t \leq 16) = \Omega(\log\log(m))$, which proves the theorem. □

We also have the following observation.

OBSERVATION 4.3. *Let $X(t)$ be the process in Figure* 2.1 *with $m = n$. Let $X(0)$ be the assignment given by $X(0) = (2, 0, 1, \ldots, 1)$. Let T be the first time at which $X(t)$ is an NE. Then $\mathbb{E}[T] = \Omega(n)$.*

The observation follows from the fact that the state does not change until one of the two tasks assigned to the first resource chooses the second resource.

5. Summary. We have analyzed a very simple, strongly distributed rerouting protocol for m tasks on n resources. We have proved an upper bound of $(\log\log m+n^4)$ on the expected convergence time (convergence to an NE), and for $m > n^3$ an upper bound of $O(\log\log m)$ on the time to reach an approximate NE. Our lower bound of $\Omega(\log\log m+n)$ matches the upper bound as a function of m. We have also shown an exponential lower bound on the convergence time for a related protocol that allows "neutral moves."

REFERENCES

[1] H. Ackermann, H. Röglin, and B. Vöcking, *On the impact of combinatorial structure on congestion games*, in Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2006, pp. 613–622.

[2] A. Blum, E. Even-Dar, and K. Ligett, *Routing without regret: On convergence to Nash equilibria of regret-minimizing algorithms in routing games*, in Proceedings of the 25th Annual ACM Symposium on Principles of Distributed Computing, 2006, pp. 45–52.

[3] S. Chien and A. Sinclair, *Convergence to approximate Nash equilibria in congestion games*, in Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), New Orleans, LA, 2007, pp. 169–178.

[4] A. Czumaj, P. Krysta, and B. Vöcking, *Selfish traffic allocation for server farms*, in Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC), Montreal, 2002, pp. 287–296.

[5] A. Czumaj, C. Riley, and C. Scheideler, *Perfectly balanced allocation*, in Proceedings of the 7th Annual International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM-APPROX), Princeton, NJ, Lecture Notes in Comput. Sci. 2764, Springer-Verlag, Berlin, 2003, pp. 240–251.

[6] A. Czumaj and B. Vöcking, *Tight bounds for worst-case equilibria*, in Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, 2002, pp. 413–420.

[7] E. Even-Dar, A. Kesselman, and Y. Mansour, *Convergence time to Nash equilibria*, in Proceedings of the 30th International Colloquium on Automata, Languages and Programming (ICALP), Eindhoven, The Netherlands, 2003, pp. 502–513.

[8] E. Even-Dar and Y. Mansour, *Fast convergence of selfish rerouting*, in Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Vancouver, BC, 2005, pp. 772–781.

[9] A. Fabrikant, C. H. Papadimitriou, and K. Talwar, *The complexity of pure Nash equilibria*, in Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC), 2004, pp. 604–612.

[10] R. Feldman, M. Gairing, T. Lücking, B. Monien, and M. Rode, *Nashification and the coordination ratio for a selfish routing game*, in Proceedings of the 30th Annual International Colloquium on Automata, Languages and Programming (ICALP 30), Lecture Notes in Comput. Sci. 2719, Springer-Verlag, Berlin, 2003, pp. 514–526.

[11] S. Fischer, H. Räcke, and B. Vöcking, *Fast convergence to Wardrop equilibria by adaptive sampling methods*, in Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC), Seattle, WA, 2006, pp. 53–662.

[12] S. Fischer and B. Vöcking, *Adaptive routing with stale information*, in Proceedings of the 24th Annual ACM SIGACT/SIGOPS Symposium on Principles of Distributed Computing (PODC), 2005, pp. 276–283.

[13] D. Fotakis, S. Kontogiannis, E. Koutsoupias, M. Mavronicolas, and P. Spirakis, *The structure and complexity of Nash equilibria for a selfish routing game*, in Proceedings of the 29th International Colloquium on Automata, Languages, and Programming (ICALP), Malaga, Spain, 2002, pp. 123–134.

[14] P. Goldberg, *Bounds for the convergence rate of randomized local search in a multiplayer load-balancing game*, in Proceedings of the 23rd Annual ACM SIGACT/SIGOPS Symposium on Principles of Distributed Computing (PODC), St. John's, Newfoundland, 2004, pp. 131–140.

[15] T. Hagerup and C. Rüb, *A guided tour of Chernoff bounds*, Inform. Process. Lett., 33 (1989), pp. 305–308.

[16] S. Ieong, R. McGrew, E. Nudelman, Y. Shoham, and Q. Sun, *Fast and compact: A simple class of congestion games*, in Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI), 2005, pp. 489–494.

[17] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis, *How easy is local search?*, J. Comput. System Sci., 37 (1988), pp. 79–100.

[18] E. Koutsoupias and C. H. Papadimitriou, *Worst-case equilibria*, in Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS), Trier, Germany, 1999, pp. 404–413.

[19] M. Luby, D. Randall, and A. Sinclair, *Markov chain algorithms for planar lattice structures*, SIAM J. Comput., 31 (2001), pp. 167–192.

[20] M. Mavronicolas and P. Spirakis, *The price of selfish routing*, in Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC), Crete, Greece, 2001, pp. 510–519.

[21] I. Milchtaich, *Congestion games with player-specific payoff functions*, Games Econom. Behav., 13 (1996), pp. 111–124.

[22] V. S. Mirrokni and A. Vetta, *Convergence issues in competitive games*, in APPROX-RANDOM, Lecture Notes in Comput. Sci. 3122, Springer-Verlag, Berlin, 2004, pp. 183–194.

[23] M. Mitzenmacher, A. Richa, and R. Sitaraman, *The power of two random choices: A survey of techniques and results*, in Handbook of Randomized Computing, P. Pardalos, S. Rajasekaran, and J. Rolim, eds., Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000, pp. 255–312.

[24] D. Monderer and L. S. Shapley, *Potential games*, Games Econom. Behav., 14 (1996), pp. 124–143.

[25] A. Orda, R. Rom, and N. Shimkin, *Competitive routing in multi-user communication networks*, IEEE/ACM Trans. Networking, 1 (1993), pp. 510–521.

[26] R. W. Rosenthal, *A class of games possessing pure-strategy Nash equilibria*, Internat. J. Game Theory, 2 (1973), pp. 65–67.

[27] T. Roughgarden, *Selfish Routing and the Price of Anarchy*, MIT Press, Cambridge, MA, 2005.

[28] T. Roughgarden and É. Tardos, *How bad is selfish routing?*, J. ACM, 49 (2002), pp. 236–259.

[29] P. Sanders, S. Egner, and J. Korst, *Fast concurrent access to parallel disks*, in Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), San Francisco, CA, 2000, pp. 849–858.

SIAM J. Comput.
Vol. 37, No. 4, pp. 1182–1198

ZONE DIAGRAMS: EXISTENCE, UNIQUENESS, AND ALGORITHMIC CHALLENGE*

TETSUO ASANO†, JIŘÍ MATOUŠEK‡, AND TAKESHI TOKUYAMA§

Abstract. A *zone diagram* is a new variation of the classical notion of the Voronoi diagram. Given points (sites) $\mathbf{p}_1, \ldots, \mathbf{p}_n$ in the plane, each $\mathbf{p}_i$ is assigned a region R_i, but in contrast to the ordinary Voronoi diagrams, the union of the R_i has a nonempty complement, the *neutral zone*. The defining property is that each R_i consists of all $\mathbf{x} \in \mathbb{R}^2$ that lie closer (nonstrictly) to $\mathbf{p}_i$ than to the union of all the other R_j, $j \neq i$. Thus, the zone diagram is defined implicitly, by a "fixed-point property," and neither its existence nor its uniqueness seem obvious. We establish existence using a general fixed-point result (a consequence of Schauder's theorem or Kakutani's theorem); this proof should generalize easily to related settings, say higher dimensions. Then we prove uniqueness of the zone diagram, as well as convergence of a natural iterative algorithm for computing it, by a geometric argument, which also relies on a result for the case of two sites in an earlier paper. Many challenging questions remain open.

Key words. computational geometry, Voronoi diagram, zone diagram, distance trisector curve

AMS subject classifications. 68U05, 65D99, 68W99, 47H10

DOI. 10.1137/06067095X

1. Introduction.

Let us consider n points (sites) $\mathbf{p}_1, \ldots, \mathbf{p}_n$ in the plane. The left picture in Figure 1 shows the (usual) Voronoi diagram, while the right picture is the *zone diagram*, a new notion investigated in the present paper.[1]

For a point $\mathbf{a}$ and a set $X \subseteq \mathbb{R}^2$ we define the *dominance region* of $\mathbf{a}$ with respect to X as

$$\operatorname{dom}(\mathbf{a}, X) = \{\mathbf{z} \in \mathbb{R}^2 : d(\mathbf{z}, \mathbf{a}) \leq d(\mathbf{z}, X)\},$$

where $d(\cdot, \cdot)$ denotes the Euclidean distance and $d(\mathbf{z}, X) = \inf_{\mathbf{x} \in X} d(\mathbf{z}, \mathbf{x})$. We note that $\operatorname{dom}(\mathbf{a}, X)$ is always convex and contains $\mathbf{a}$.

In the classical Voronoi diagram, the region of the site $\mathbf{p}_i$ is $\operatorname{dom}(\mathbf{p}_i, \{\mathbf{p}_j : j \neq i\})$, and the regions tile the whole plane. In a zone diagram, each $\mathbf{p}_i$ also has a region R_i, but the union of all the regions has a nonempty complement, called the *neutral zone*.

*Received by the editors September 28, 2006; accepted for publication (in revised form) April 20, 2007; published electronically November 21, 2007.
http://www.siam.org/journals/sicomp/37-4/67095.html

†School of Information Science, JAIST, 1-1 Asahidai, Nomi, Ishikawa, 923-1292 Japan (t_asano@jaist.ac.jp). This author's research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research on Priority Areas and Scientific Research (B).

‡Department of Applied Mathematics and Institute of Theoretical Computer Science (ITI), Charles University, Malostranské nám. 25, 118 00 Praha 1, Czech Republic (matousek@kam.mff.cuni.cz). Parts of this author's research were done during visits to the Japanese Advanced Institute for Science and Technology (JAIST) and to the ETH Zürich; the support of these institutions is gratefully acknowledged.

§Graduate School of Information Sciences, Tohoku University, Aramaki Aza Aoba, Aoba-ku, Sendai, 980-8579 Japan (tokuyama@dais.is.tohoku.ac.jp). This author's research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research on Priority Areas.

[1] In earlier papers [2, 1] we have used the longer name *Voronoi diagram with neutral zone*, but here we propose the shorter term.

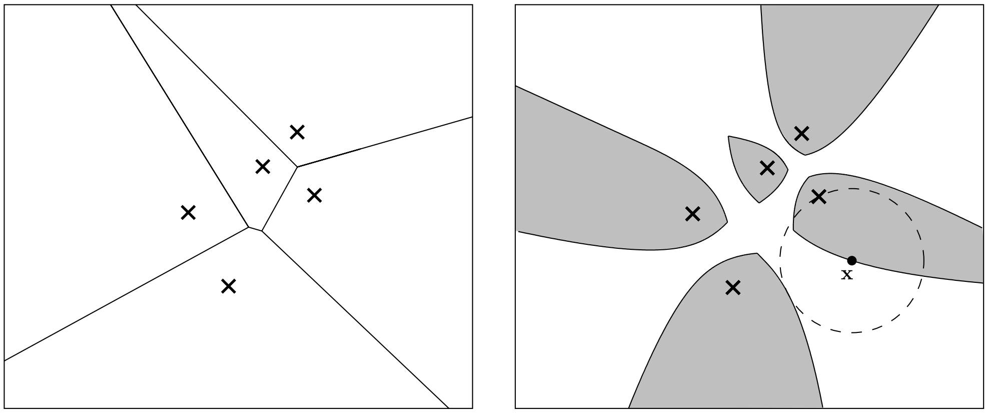


FIG. 1. *Five sites are marked by crosses. The left picture is the classical Voronoi diagram. The right picture shows the zone diagram: Each site* $\mathbf{p}_i$ *has a dominance region* R_i*, and the distance of each point* $\mathbf{x}$ *on the boundary of* R_i *to* $\mathbf{p}_i$ *equals the distance of* $\mathbf{x}$ *to the union of the other regions.*

We require that

$$R_i = \operatorname{dom}\Big(\mathbf{p}_i, \bigcup_{j\neq i} R_j\Big) \quad \text{for all } i = 1, 2, \ldots, n; \tag{1.1}$$

in words, the region of each site should consist of all points that are closer (nonstrictly) to the site than to all of the other regions. In particular, each R_i is convex and contains $\mathbf{p}_i$.

The notion of the zone diagram can be illustrated by a story about equilibrium in an "age of wars." There are n mutually hostile kingdoms. The ith kingdom has a castle at the site $\mathbf{p}_i$ and a territory R_i around it. The n territories are separated by a no-man's land, the neutral zone. If the territory R_i is attacked from another kingdom, an army departs from the castle $\mathbf{p}_i$ to intercept the attack. The interception succeeds if and only if the defending army arrives at the attacking point on the boundary of R_i sooner than the enemy. However, the attacker can secretly move his troops inside his territory, and the defense army can start from its castle only when the attacker leaves his territory. The zone diagram is an equilibrium configuration of the territories, such that every kingdom can guard its territory and no kingdom can grow without risk of invasion by other kingdoms.

The notion of the zone diagram is, in our opinion, very interesting and poses many mathematical and algorithmic challenges. Moreover, zone diagrams or variations could be useful for modeling natural phenomena. The classical Voronoi diagram, one of the basic geometric structures, appears in many fields and, among other uses, it is frequently employed as a mathematical model of a simultaneous growth from several sites (cells in a tissue, a crystal lattice, geological patterns, regional equilibria in social sciences, etc.). Voronoi diagrams and their numerous generalizations (see, e.g., [3, 5]) subdivide all of the space into dominance regions of the sites. However, geometric structures are sometimes observed in nature where the dominance regions do not cover everything, which might be a result of a growth process where the growth terminates before the cell boundaries meet, due to some noncontact action.

The above definition of the zone diagram is implicit, since each region is defined in terms of the remaining ones. So it is not obvious whether any system of regions with the required property exists at all, or whether it is determined uniquely. Here we answer both of these questions affirmatively.

THEOREM 1.1. *For every choice of n distinct sites $\mathbf{p}_1, \dots, \mathbf{p}_n \in \mathbb{R}^2$ there exists exactly one system $(R_1, \dots, R_n)$ of subsets of the plane satisfying* (1.1).

Perhaps surprisingly, the case of two sites ($n = 2$) is already nontrivial. We showed existence and uniqueness for $n = 2$ in [1]. Here the two regions are mirror images of one another, and they are bounded by an interesting curve called the *distance trisector curve.* We conjecture that this curve is not algebraic and not even expressible by elementary functions (but we have no proof so far). On the other hand, its intersection with a given line parallel to the segment $\mathbf{p}_1\mathbf{p}_2$ can be computed to any desired precision in time polynomial in the number of required digits.

The *existence* part of Theorem 1.1 is proved in section 4. We apply a well-known fixed-point theorem for infinite-dimensional Banach spaces to a suitable space of n-tuples of regions. This proof is conceptually simple and appears quite robust, in the sense that it should be possible to adapt it to various natural generalizations of zone diagrams, such as zone diagrams in $\mathbb{R}^d$, zone diagrams of nonpoint sites, or α-zone diagrams (where each point of R_i should be α times closer to $\mathbf{p}_i$ than to $\bigcup_{j \neq i} R_j$, for some real parameter $\alpha > 0$). We should remark, though, that such modifications are not necessarily trivial, since some elementary geometric estimates are needed that might prove technically challenging in some settings.

In section 5 we prove the uniqueness in Theorem 1.1 and, at the same time, we also reprove existence by a different method, similar to the one we used in [1]. This method currently seems very much restricted to the planar case of zone diagrams, and several obstacles would have to be overcome before it could be generalized to $\mathbb{R}^3$, say.

In the uniqueness proof, we consider a natural iterative procedure for approximating the zone diagram. Let the sites $\mathbf{p}_1, \dots, \mathbf{p}_n$ be fixed and let $\mathbf{R} = (R_1, \dots, R_n)$ be an ordered n-tuple of regions (nonempty subsets of $\mathbb{R}^2$), where we assume $\mathbf{p}_1 \in R_1, \dots, \mathbf{p}_n \in R_n$. We define $\mathbf{Dom}(\mathbf{R})$ as the ordered n-tuple $\mathbf{S} = (S_1, S_2, \dots, S_n)$ of new regions, where $S_i = \operatorname{dom}(\mathbf{p}_i, \bigcup_{j \neq i} R_j)$. Thus, rephrasing our definition of a zone diagram, an n-tuple $\mathbf{R} = (R_1, \dots, R_n)$ of regions is a zone diagram of $\mathbf{p}_1, \dots, \mathbf{p}_n$ if it is a fixed point of the operator $\mathbf{Dom}$; that is, if $\mathbf{R} = \mathbf{Dom}(\mathbf{R})$.

For two n-tuples $\mathbf{R} = (R_1, \dots, R_n)$ and $\mathbf{S} = (S_1, \dots, S_n)$, let us write $\mathbf{R} \preceq \mathbf{S}$ if $R_i \subseteq S_i$ for all $i = 1, 2, \dots, n$. It is immediate from the definition that if $\mathbf{R} \preceq \mathbf{S}$, then $\mathbf{Dom}(\mathbf{R}) \succeq \mathbf{Dom}(\mathbf{S})$ (that is, the dominance operator is antimonotone with respect to $\preceq$). Let $\mathbf{I}^{(0)} = (\{\mathbf{p}_1\}, \dots, \{\mathbf{p}_n\})$ be the (smallest possible) system of one-point regions, let $\mathbf{O}^{(0)} = (O_1^{(0)}, \dots, O_n^{(0)}) = \mathbf{Dom}(\mathbf{I}^{(0)})$ be the regions of the classical Voronoi diagram of $\mathbf{p}_1, \dots, \mathbf{p}_n$, and for $k = 1, 2, \dots$, inductively define $\mathbf{I}^{(k)} = \mathbf{Dom}(\mathbf{O}^{(k-1)})$, $\mathbf{O}^{(k)} = \mathbf{Dom}(\mathbf{I}^{(k)})$.

Antimonotonicity of $\mathbf{Dom}$ and induction yield $\mathbf{I}^{(0)} \preceq \mathbf{I}^{(1)} \preceq \mathbf{I}^{(2)} \preceq \cdots$ and $\mathbf{O}^{(0)} \succeq \mathbf{O}^{(1)} \succeq \mathbf{O}^{(2)} \succeq \cdots$. Moreover, if $\mathbf{R}$ is a zone diagram, i.e., it satisfies $\mathbf{R} = \mathbf{Dom}(\mathbf{R})$, then we have $\mathbf{I}^{(0)} \preceq \mathbf{R}$ by definition, and induction and antimonotonicity give $\mathbf{I}^{(k)} \preceq \mathbf{R} \preceq \mathbf{O}^{(k)}$ for all k. The $\mathbf{I}^{(k)}$ form an increasing sequence of inner approximations of the zone diagram, while the $\mathbf{O}^{(k)}$ form a decreasing sequence of outer approximations; see Figure 2.

In section 5 we show that the inner and outer approximations converge to the same limit, which has to be the unique zone diagram. This also gives a quite practical algorithm for approximate construction of the zone diagram. The regions of the $\mathbf{I}^{(k)}$ and $\mathbf{O}^{(k)}$ can be approximated by convex polygons with many sides—this is how the pictures of zone diagrams in this paper were obtained. With some care in

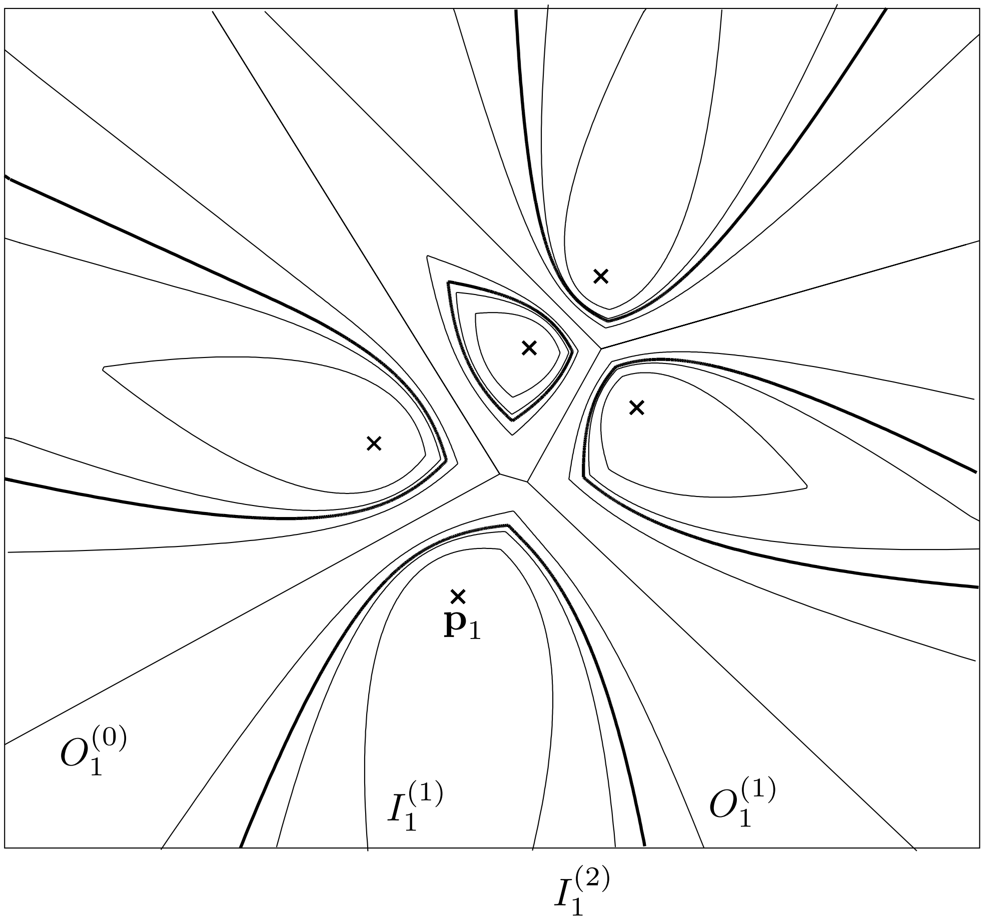


FIG. 2. *The inner and outer approximations* $\mathbf{I}^{(k)}$ *and* $\mathbf{O}^{(k)}$. *In particular,* $\mathbf{O}^{(0)}$ *forms the classical Voronoi diagram.*

implementation one can actually get pairs of polygons that are provably inner and outer approximations, respectively, of the regions of the zone diagram. Experiments indicate that the convergence of this algorithm is quite fast, at least for small sets of sites (each iteration is computationally demanding, though). Unfortunately, we have no theoretical estimate of the convergence rate of this algorithm. An example illustrating some of the difficulties in proving estimates is given in section 6. We also mention some additional results and questions there.

2. A guided tour of zone diagrams. Before we start with proofs, we explain, mainly by pictures, some interesting phenomena arising in zone diagrams, illustrating that they behave very differently from the classical Voronoi diagrams.

The left picture in Figure 3 shows the zone diagram of two sites (the distance trisector curve), and the right picture shows the zone diagram after adding a third site marked by a small disk. The boundary curves of the regions from the previous two-site diagram are also shown, and one can see that the region of the top site has *gained* new area after the new site was added (this cannot happen in classical Voronoi diagrams). This is very intuitive in the war interpretation: The animosity of the two

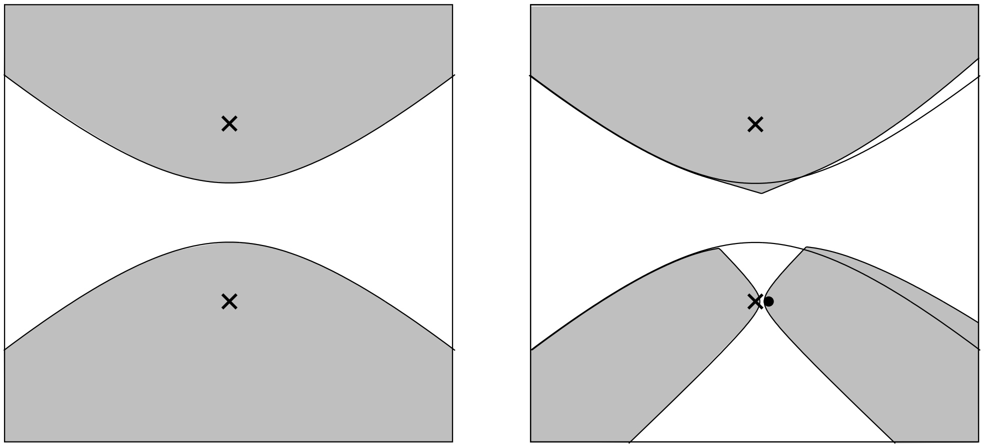


Fig. 3. *The zone diagram of two sites (left) and the zone diagram after adding a third site marked by ● (right). The top site gains area, and the regions are not bounded only by arcs of distance trisector curves.*

nearby sites weakens them, and the top site gets relatively stronger.

In a classical Voronoi diagram for sites $\mathbf{p}_1, \ldots, \mathbf{p}_n$, the region of $\mathbf{p}_i$ is the intersection of the regions of $\mathbf{p}_i$ in the two-site Voronoi diagrams for all pairs $\{\mathbf{p}_i, \mathbf{p}_j\}$, $j \neq i$. Consequently, each region is bounded by segments that arise as bisectors of pairs of sites. Figure 3 illustrates that no analogy holds for zone diagrams. Indeed, segments of the distance trisector curve do appear as portions of the boundary of the regions for three sites, but we can also have other kinds of curves.

In Figure 3 we can see a simple instance of this (straight segments appear near the bottom tip of the top region); other examples exhibit more complicated curves as well. The proof in section 5 tells something about the nature of all curves that can ever appear, but some interesting questions remain open.

The left picture in Figure 4 shows an aesthetically pleasing zone diagram. In this case all of the regions are bounded, which again does not happen in classical Voronoi diagrams. Such "flowers" scaled down to a tiny size can be used in constructing examples; the right picture shows a small flower and an isolated site $\mathbf{q}$. As the flower gets smaller, the region of $\mathbf{q}$ approaches a half-plane, that is, the region of $\mathbf{q}$ in a two-site classical Voronoi diagram with a single site at the center of the flower.

3. Preliminaries.

Here we introduce some notation and some simple and/or known facts.

We note that for any $X \subseteq \mathbb{R}^2$ the dominance region $\mathrm{dom}(\mathbf{a}, X)$ is a closed convex set, since it can be represented as the intersection $\bigcap_{\mathbf{x} \in X} \mathrm{dom}(\mathbf{a}, \{\mathbf{x}\})$ of half-planes.

The boundary of a set $X \subseteq \mathbb{R}^2$ is denoted by ∂X.

In analogy to the dominance region notation $\mathrm{dom}(\mathbf{a}, X)$ we will also use the *bisector* notation defined by $\mathrm{bisect}(\mathbf{a}, X) = \{\mathbf{z} \in \mathbb{R}^2 : d(\mathbf{z}, \mathbf{a}) = d(\mathbf{z}, X)\}$.

For a nonempty closed convex set $C \subseteq \mathbb{R}^2$ and a point $\mathbf{x} \in \mathbb{R}^2$, we let $\mathrm{prox}_C(\mathbf{x})$ denote the point of C nearest to $\mathbf{x}$. It is well known that this point is unique. Moreover, for C fixed, the mapping prox_C (the *metric projection*) is continuous, and actually 1-Lipschitz.

We will need the following lemma, expressing a kind of continuity of the dominance operator.

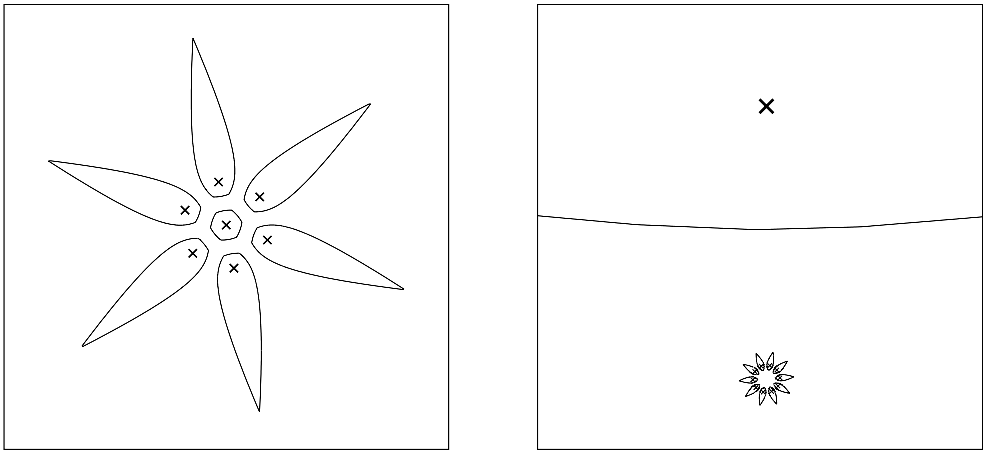


FIG. 4. *A flower (left); a small flower induces almost the classical Voronoi region of an isolated site (right).*

LEMMA 3.1. *Let* $\mathbf{a} \in \mathbb{R}^2$ *be a point and let* $X_1 \supseteq X_2 \supseteq X_3 \supseteq \cdots$ *be a decreasing sequence of closed subsets of* $\mathbb{R}^2$ *with* $\mathbf{a} \notin X_1$. *Let us set* $X = \bigcap_{k=1}^{\infty} X_k$. *Then*

$$\mathrm{dom}(\mathbf{a}, X) = \mathrm{cl}\Big(\bigcup_{k=1}^{\infty} \mathrm{dom}(\mathbf{a}, X_k)\Big),$$

where cl(.) *denotes the topological closure.*

Proof. The inclusion "$\supseteq$" is clear from $X \subseteq X_k$ for all k and antimonotonicity of dom(.). To prove the opposite inclusion, we fix $\mathbf{x} \in \mathrm{dom}(\mathbf{a}, X)$ arbitrarily, we choose $\varepsilon > 0$ arbitrarily small, and we show that there exists $k = k(\mathbf{x}, \varepsilon)$ with $d(\mathbf{x}, \mathrm{dom}(\mathbf{a}, X_k)) < \varepsilon$. We may assume $\mathbf{x} \neq \mathbf{a}$, for otherwise, we even have $\mathbf{x} \in \mathrm{dom}(\mathbf{a}, X_1)$.

Since $\mathbf{a} \notin X_1$ and X_1 is closed, we have $\delta = d(\mathbf{a}, X_1) > 0$. The set X lies outside the region shown in Figure 5. Elementary geometric considerations show that all interior points $\mathbf{y}$ of the segment $\mathbf{ax}$ satisfy $d(\mathbf{y}, \mathbf{a}) < d(\mathbf{y}, X)$. Let us choose such a point $\mathbf{y}$ with $d(\mathbf{y}, \mathbf{x}) < \varepsilon$.

A simple compactness argument, which we omit, shows that for any point $\mathbf{q}$ we have $d(\mathbf{q}, X) = \lim_{k\to\infty} d(\mathbf{q}, X_k)$. Hence there exists k with $d(\mathbf{y}, \mathbf{a}) < d(\mathbf{y}, X_k)$, and thus $\mathbf{y} \in \mathrm{dom}(\mathbf{a}, X_k)$. Hence $d(\mathbf{x}, \mathrm{dom}(\mathbf{a}, X_k)) < \varepsilon$ as claimed. ◻

4. Existence of the zone diagram. In this section we prove the existence of (at least one) zone diagram for every set $\{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ of distinct sites in the plane. Let $\mathcal{R}$ denote the set of all n-tuples $\mathbf{R} = (R_1, \ldots, R_n)$ of sets with $\mathbf{p}_i \in R_i \subseteq \mathbb{R}^2$.

Plan of the proof. We want to show the existence of a fixed point of the dominance operator $\mathbf{Dom}\colon \mathcal{R} \to \mathcal{R}$ (defined in section 1). We are going to apply the following theorem (which can be seen as a special case of two famous theorems in fixed-point theory, Schauder's and Kakutani's; see, for example, Zeidler [6, Corollary 2.13]).

THEOREM 4.1. *Let* Z *be a Banach space, let* $K \subset Z$ *be a nonempty, compact, and convex set, and let* $F\colon K \to K$ *be a continuous map. Then* F *has at least one fixed point.*

In our application of this fixed-point theorem, we will define a suitable set $\mathcal{S} \subseteq \mathcal{R}$ of n-tuples of regions and define an embedding $\varphi\colon \mathcal{S} \to Z$ for a suitable Banach space

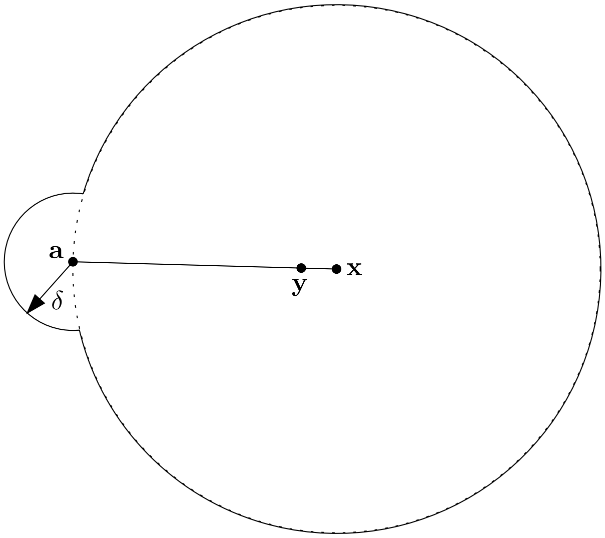


FIG. 5. *Illustration of the proof of Lemma* 3.1.

Z. The image $\varphi(\mathcal{S})$ will play the role of K in the fixed-point theorem, and F is the mapping $K \to Z$ corresponding to **Dom** under φ (formally, $F = \varphi \circ \mathbf{Dom} \circ \varphi^{-1}$). We thus need to verify that K is convex and compact, that $F(K) \subseteq K$, and that F is continuous.

Here we will present our original "manual" approach to this task. We will define $\mathcal{S}$ in a slightly tricky manner, which makes the verification of the above conditions quite easy, *except* for checking the continuity of F, which is not really hard, but we need about two pages of elementary geometric arguments and estimates.

An alternative strategy. An alternative, somewhat simpler, and, in a sense, more natural approach (leading to a formally slightly weaker result) was suggested to us by Eva Kopecká. We sketch it here and then return to our original proof. First of all, we restrict everything to a bounded region Q, say a large square containing all the sites, and prove the existence of the zone diagram only in this region (that is why the result is formally weaker). Then we let Q_i be the intersection of Q with the cell of $\mathbf{p}_i$ in the classical Voronoi diagram of $\mathbf{p}_1, \dots, \mathbf{p}_n$, and we define $\mathcal{S}$ as the set of all n-tuples $(S_1, \dots, S_n)$ of nonempty closed sets with $\mathbf{p}_i \in S_i \subseteq Q_i$. We equip this $\mathcal{S}$ with the Hausdorff distance metric; formally, the distance of $(S_1, \dots, S_n)$ and $(S'_1, \dots, S'_n)$ equals $\max_{i=1,2,\dots,n} h(S_i, S'_i)$, where h is the Hausdorff distance. It follows from the work of Curtis, Schori, and West from the 1970s (culminating in [4], where other references can also be found) that this $\mathcal{S}$ as a topological space is homeomorphic to the Hilbert cube, which is a compact convex subset of ℓ_2. Hence for application of Theorem 4.1 it is enough to verify that **Dom** maps $\mathcal{S}$ into $\mathcal{S}$ (clear) and that it is continuous with respect to the Hausdorff metric. This is similar in spirit to our continuity argument below but simpler.

Radial functions. We return to our original approach. Let S^1 denote the unit circle; we will interpret its points as angles in the interval $[0, 2\pi)$. We will call a continuous function $\rho\colon S^1 \to [0, \frac{\pi}{2}]$ a *radial function.* For such a ρ and a point $\mathbf{p} \in \mathbb{R}^2$, we define a star-shaped region $R = \mathrm{reg}_{\mathbf{p}}(\rho)$ such that the ray emanating from $\mathbf{p}$ at angle α intersects R in a segment of length $\tan(\rho(\alpha))$. Formally,

$$\operatorname{reg}_{\mathbf{p}}(\rho) = \bigcup_{\alpha\in[0,2\pi)} \mathbf{p}\mathbf{x}_\alpha, \quad \text{where } \mathbf{x}_\alpha = \mathbf{p} + \tan(\rho(\alpha))(\cos\alpha, \sin\alpha)$$

(if $\rho(\alpha) = \frac{\pi}{2}$, then $\mathbf{p}\mathbf{x}_\alpha$ is defined as the full semi-infinite ray). We note that the length of the segment in direction α is not $\rho(\alpha)$ but rather $\tan(\rho(\alpha))$. This ensures that we deal with bounded radial functions, although the considered planar regions are often unbounded. The choice of the tangent function to map a bounded interval to $[0,\infty)$ is somewhat arbitrary, but certainly not every function would do. For example, we have to be careful about how we measure the distance of regions, in order to obtain continuity of the operator **Dom**.

For simplicity, let us assume that every two sites $\mathbf{p}_i \neq \mathbf{p}_j$ have distance at least 4 (this will save us one parameter, standing for the minimum distance of sites, in the forthcoming calculations).

Now we can define our Banach space and the set K.

Definition 4.2. *Let Z denote the Banach space of all n-tuples $\boldsymbol{\rho} = (\rho_1, \dots, \rho_n)$ of continuous functions $\rho_i : S^1 \to \mathbb{R}$, endowed with the supremum norm: $\|\boldsymbol{\rho}\|_\infty = \max_{i=1,2,\dots,n} \max_{\alpha\in S^1} |\rho_i(\alpha)|$.*

Let $K \subset Z$ consist of all $\boldsymbol{\rho} \in Z$ satisfying the following conditions:

(i) *The image of each ρ_i is contained in $[0, \frac{\pi}{2}]$ (that is, ρ_i is a radial function).*

(ii) *We have $\mathbf{I}^{(1)} \preceq \mathbf{reg}(\boldsymbol{\rho}) \preceq \mathbf{O}^{(1)}$, where $\mathbf{reg}(\boldsymbol{\rho}) = (\operatorname{reg}_{\mathbf{p}_1}(\rho_1), \dots, \operatorname{reg}_{\mathbf{p}_n}(\rho_n)) \in \mathcal{R}$ is the system of regions defined by $\boldsymbol{\rho}$ (the componentwise inclusion operator $\preceq$ and the $\mathbf{I}^{(k)}$ and $\mathbf{O}^{(k)}$ were introduced in section* 1*). (We note that this simply means pointwise lower and upper bounds on each ρ_i.)*

(iii) *Each ρ_i is 2-Lipschitz.*

Further, we set $\mathcal{S} = \mathbf{reg}(K)$ (so **reg** *plays the role of φ^{-1} in the abstract outline of the argument given above).*

The set K is clearly nonempty and convex (since convex combinations preserve the conditions in the definition of K), and it is easily seen to be compact by the Arzèla–Ascoli theorem, which implies, in particular, that any closed set of uniformly bounded 2-Lipschitz functions on a compact set is compact.

Lemma 4.3. *For every n-tuple $\mathbf{R} \in \mathcal{S}$ of regions we have $\mathbf{Dom}(\mathbf{R}) \in \mathcal{S}$. Consequently, the mapping $F : K \to K$ given by $F = \mathbf{reg}^{-1} \circ \mathbf{Dom} \circ \mathbf{reg}$ is well defined.*

Proof. Let $\mathbf{S} = \mathbf{Dom}(\mathbf{R}) = (S_1, \dots, S_n)$. Each S_i is convex and hence given by a radial function; thus $\boldsymbol{\sigma} = \mathbf{reg}^{-1}(\mathbf{S})$ is well defined.

Since $\mathbf{R} \in \mathcal{S}$, we have $\mathbf{I}^{(1)} \preceq \mathbf{R} \preceq \mathbf{O}^{(1)}$, and by antimonotonicity we get $\mathbf{I}^{(2)} = \mathbf{Dom}(\mathbf{O}^{(1)}) \preceq \mathbf{Dom}(\mathbf{R}) = \mathbf{S}$. Similarly $\mathbf{S} \preceq \mathbf{O}^{(2)}$, so $\mathbf{I}^{(1)} \preceq \mathbf{I}^{(2)} \preceq \mathbf{S} \preceq \mathbf{O}^{(2)} \preceq \mathbf{O}^{(1)}$, and $\mathbf{S}$ satisfies (ii) from Definition 4.2.

For proving (iii), we first note that, assuming $d(\mathbf{p}_i, \mathbf{p}_j) \geq 4$ for all $i \neq j$, each $I_i^{(1)}$ contains the unit disk centered at $\mathbf{p}_i$. Indeed, the regions of $\mathbf{O}^{(0)}$ are the classical Voronoi regions, and so $I_i^{(1)}$ consists of the points closer to $\mathbf{p}_i$ than to the Voronoi regions of the other points. It remains to note that the Voronoi region of $\mathbf{p}_i$ contains the disk of radius 2 around $\mathbf{p}_i$.

It remains to prove the following claim: *If ρ is a radial function, $\mathbf{p} \in \mathbb{R}^2$, and the set $R = \operatorname{reg}_{\mathbf{p}}(\rho)$ is convex and contains the unit disk centered at $\mathbf{p}$, then ρ is a 2-Lipschitz function.* (Actually, an easy refinement of the proof yields 1-Lipschitz.)

First we note that R may be assumed to be bounded, since intersecting R with a very large disk changes the radial function by an arbitrarily small amount.

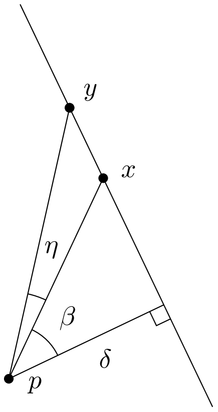


FIG. 6. *The radial function of a convex set containing the unit disk is Lipschitz.*

Let $\mathbf{x}$ be the boundary point of R in direction α and let $\mathbf{y}$ be the boundary point in direction $\alpha+\eta$, where (as we may assume) $0<\eta\le\frac{\pi}{6}$, say (see Figure 6). Then it is easy to see that the line $\mathbf{xy}$ has distance $\delta\ge\frac{1}{2}$ to $\mathbf{p}$: We have $|\mathbf{px}|=\delta/\cos\beta$ and $|\mathbf{py}|=\delta/\cos(\beta+\eta)$, and thus $|\rho(\alpha+\eta)-\rho(\alpha)|=|\arctan(\delta/\cos(\beta+\eta))-\arctan(\delta/\cos\beta)|\le\eta/\delta$, where the last inequality follows from the mean value theorem applied to the function $\beta\mapsto\arctan(\delta/\cos\beta)$. Its first derivative is $\frac{1}{\delta}\sin\beta/(1+(\cos\beta)^2/\delta^2)$, and this is obviously bounded by $\frac{1}{\delta}$. □

In order to apply Theorem 4.1, it thus remains to prove the following.

LEMMA 4.4. *The mapping $F=\mathbf{reg}^{-1}\circ\mathbf{Dom}\circ\mathbf{reg}\colon K\to K$ is continuous.*

We give the proof of the above lemma in the appendix. The existence of a zone diagram then follows from Theorem 4.1.

5. Uniqueness of the zone diagram. In this section we prove both existence and uniqueness of the zone diagram for any n distinct sites $\mathbf{p}_1,\dots,\mathbf{p}_n$, as well as convergence of the iterative procedure described in the introduction. The proof is divided into two steps. The first step is the following quite intuitive statement.

LEMMA 5.1. *Let $\mathbf{I}^{(k)}=(I_1^{(k)},\dots,I_n^{(k)})$ be the inner approximations of the zone diagram and let $\mathbf{O}^{(k)}=(O_1^{(k)},\dots,O_n^{(k)})$ be the outer approximations. For $i=1,2,\dots,n$ let us set*

$$I_i=\mathrm{cl}\Bigl(\bigcup_{k=0}^{\infty}I_i^{(k)}\Bigr),\quad O_i=\bigcap_{k=0}^{\infty}O_i^{(k)}.$$

Then $\mathbf{I}=(I_1,\dots,I_n)$ and $\mathbf{O}=(O_1,\dots,O_n)$ satisfy $\mathbf{I}=\mathbf{Dom}(\mathbf{O})$ and $\mathbf{O}=\mathbf{Dom}(\mathbf{I})$.

Proof. This statement is not as obvious as it might perhaps seem. First we check $\mathbf{O}=\mathbf{Dom}(\mathbf{I})$; this is entirely straightforward. Fixing i, we want to verify

$$O_i=\mathrm{dom}\Bigl(\mathbf{p}_i,\bigcup_{j\ne i}I_j\Bigr). \tag{5.1}$$

Since for every k, $I_j\supseteq I_j^{(k)}$, we have $\mathrm{dom}(\mathbf{p}_i,\bigcup_{j\ne i}I_j)\subseteq\mathrm{dom}(\mathbf{p}_i,\bigcup_{j\ne i}I_j^{(k)})=O_i^{(k)}$, and thus $\mathrm{dom}(\mathbf{p}_i,\bigcup_{j\ne i}I_j)\subseteq\bigcap_{k=0}^{\infty}O_i^{(k)}=O_i$; this is "$\supseteq$" in (5.1). For the converse inclusion, we assume $\mathbf{x}\notin\mathrm{dom}(\mathbf{p}_i,\bigcup_{j\ne i}I_j)$. Then there exist j_0 and $\mathbf{y}\in I_{j_0}$ with $d(\mathbf{x},\mathbf{p}_i)>d(\mathbf{x},\mathbf{y})$. Setting $\varepsilon=d(\mathbf{x},\mathbf{p}_i)-d(\mathbf{x},\mathbf{y})$, we can choose k sufficiently large so that $I_{j_0}^{(k)}$ contains a point $\mathbf{y}'$ with $d(\mathbf{y},\mathbf{y}') < \varepsilon$, and then $d(\mathbf{x},\mathbf{y}') \le d(\mathbf{x},\mathbf{y}) + d(\mathbf{y},\mathbf{y}') < d(\mathbf{x},\mathbf{y}) + \varepsilon = d(\mathbf{x},\mathbf{p}_i)$. Hence $\mathbf{x} \notin \mathrm{dom}(\mathbf{p}_i, \bigcup_{j\neq i} I_j^{(k)}) = O_i^{(k)}$, and $\mathbf{x} \notin O_i$ either. This proves (5.1).

We now turn to showing $\mathbf{I} = \mathbf{Dom}(\mathbf{O})$; that is,

$$I_i = \mathrm{dom}\Big(\mathbf{p}_i, \bigcup_{j\neq i} O_j\Big).$$

Here "$\subseteq$" is again straightforward, but "$\supseteq$" needs more properties of dom(.). We apply Lemma 3.1 with $\mathbf{a} = \mathbf{p}_i$ and $X_k = \bigcup_{j\neq i} O_j^{(k)}$. Since the $O_i^{(1)}$ are disjoint, we have $X = \bigcap_{k=1}^{\infty} X_k = \bigcup_{j\neq i} \bigcap_{k=1}^{\infty} O_j^{(k)} = \bigcup_{j\neq i} O_j$, and so the lemma tells us that $\mathrm{dom}(\mathbf{p}_i, \bigcup_{j\neq i} O_j) = \mathrm{cl}\big(\bigcup_{k=1}^{\infty} \mathrm{dom}(\mathbf{p}_i, \bigcup_{j\neq i} O_j^{(k)})\big) = \mathrm{cl}\big(\bigcup_{k=1}^{\infty} I_i^{(k+1)}\big) = I_i$ as required. Lemma 5.1 is proved. □

In the second step, which is the essence of the proof, we establish the following proposition.

Proposition 5.2. *Let the sites* $\mathbf{p}_1, \dots, \mathbf{p}_n$ *be fixed and let* $\mathbf{S} = (S_1, \dots, S_n)$ *and* $\mathbf{T} = (T_1, \dots, T_n)$ *be n-tuples of regions satisfying* $\mathbf{S} = \mathbf{Dom}(\mathbf{T})$ *and* $\mathbf{T} = \mathbf{Dom}(\mathbf{S})$. *Then* $\mathbf{S} = \mathbf{T}$*, and, consequently,* $\mathbf{S}$ *is a zone diagram of* $\mathbf{p}_1, \dots, \mathbf{p}_n$.

Proof of Theorem 1.1. Let $\mathbf{I}$ and $\mathbf{O}$ be as in Lemma 5.1. Then Proposition 5.2 with $\mathbf{S} = \mathbf{I}$ and $\mathbf{T} = \mathbf{O}$ shows that $\mathbf{I} = \mathbf{O}$ is a zone diagram. Moreover, if $\mathbf{R}$ is any zone diagram of $\mathbf{p}_1, \dots, \mathbf{p}_n$, we have $\mathbf{I}^{(k)} \preceq \mathbf{R} \preceq \mathbf{O}^{(k)}$ for all k as was explained in section 1. Hence $\mathbf{I} \preceq \mathbf{R} \preceq \mathbf{O}$ and $\mathbf{R} = \mathbf{I} = \mathbf{O}$. Thus the zone diagram is unique. □

Preparations for the proof of Proposition 5.2. We assume that $\mathbf{S}$ and $\mathbf{T}$ with $\mathbf{S} = \mathbf{Dom}(\mathbf{T})$ and $\mathbf{T} = \mathbf{Dom}(\mathbf{S})$ have been fixed. Since each S_i and each T_i is a dominance region, it is a closed convex set. Since $\mathbf{I}^{(1)} \preceq \mathbf{S}, \mathbf{T}$, each S_i and each T_i contains a small open disk around $\mathbf{p}_i$. Moreover, each S_i is disjoint from all T_j, $j \neq i$, and vice versa.

We introduce the following terminology: Let $\mathbf{a} \in S_1$ be a point. The *nearest points* of $\mathbf{a}$ are the points of $\bigcup_{i=2}^{n} T_i$ with the minimum distance to $\mathbf{a}$. Since each T_i is convex, it contains at most one of the nearest points of $\mathbf{a}$. The point $\mathbf{a}$ is called a *singular point* if it has more than one nearest point; otherwise, it is called a *regular point.* All of this refers to the situation $\mathbf{a} \in S_1$; if we speak about nearest points of some $\mathbf{a} \in T_2$, say, we mean the points of $\bigcup_{i\neq 2} S_i$ with the smallest distance to $\mathbf{a}$.

Let $\breve{\mathbf{a}} \in \bigcup_{i=2}^{n} T_i$ be a nearest point of $\mathbf{a} \in \partial S_1$. We call it *visible* if the segment $\mathbf{a}\breve{\mathbf{a}}$ intersects S_1 only at $\mathbf{a}$, and we call it *obscured* otherwise.

Lemma 5.3.

(i) *Let* $\mathbf{a} \in \partial S_1$ *and let* $\breve{\mathbf{a}} \in \bigcup_{i=2}^{n} T_i$ *be a nearest point of* $\mathbf{a}$*, say with* $\breve{\mathbf{a}} \in T_2$. *Then* $d(\mathbf{p}_1, \mathbf{a}) = d(\mathbf{a}, \breve{\mathbf{a}}) \ge d(\breve{\mathbf{a}}, \mathbf{p}_2)$.

(ii) *In the setting of* (i), $\breve{\mathbf{a}}$ *is visible.*

(iii) *Let* $\mathbf{a}$ *and* $\mathbf{b}$ *be distinct boundary points of* S_1*, and let* $\breve{\mathbf{a}}$ *be a nearest point of* $\mathbf{a}$ *and* $\breve{\mathbf{b}}$ *a nearest point of* $\mathbf{b}$. *Then the segments* $\mathbf{a}\breve{\mathbf{a}}$ *and* $\mathbf{b}\breve{\mathbf{b}}$ *do not intersect, except possibly if* $\breve{\mathbf{a}} = \breve{\mathbf{b}}$.

Proof. In part (i) we have $d(\mathbf{p}_1, \mathbf{a}) \le d(\mathbf{a}, \breve{\mathbf{a}})$ because $\mathbf{a} \in S_1 = \mathrm{dom}(\mathbf{p}_1, \bigcup_{i=2}^{n} T_i) \subseteq \mathrm{dom}(\mathbf{p}_1, \{\breve{\mathbf{a}}\})$. If we had $d(\mathbf{p}_1, \mathbf{a}) < d(\mathbf{a}, \breve{\mathbf{a}})$, then a small neighborhood of $\mathbf{a}$ would be contained in S_1 as well, but $\mathbf{a}$ is a boundary point. Next, we have $\breve{\mathbf{a}} \in T_2 = \mathrm{dom}(\mathbf{p}_2, \bigcup_{i=2}^{n} S_i) \subseteq \mathrm{dom}(\mathbf{p}_2, \{\mathbf{a}\})$, and $d(\mathbf{a}, \breve{\mathbf{a}}) \ge d(\mathbf{p}_2, \breve{\mathbf{a}})$ follows, which finishes the proof of (i).

For (ii), let us suppose that $\mathbf{a} \in \partial S_1$ has a nearest point $\breve{\mathbf{a}} \in T_i$ and that $\mathbf{b} \neq \mathbf{a}$ is a point of S_1 on the segment $\mathbf{a}\breve{\mathbf{a}}$. Then $d(\breve{\mathbf{a}}, \mathbf{b}) = d(\breve{\mathbf{a}}, \mathbf{a}) - d(\mathbf{a}, \mathbf{b}) = d(\mathbf{p}_1, \mathbf{a}) - d(\mathbf{a}, \mathbf{b})$.

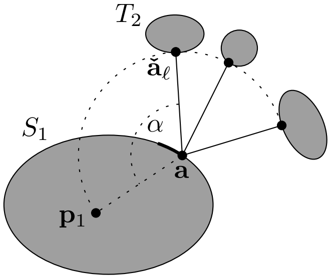


FIG. 7. *The left nearest point of* $\mathbf{a}$.

On the other hand, the point $\mathbf{b}$ does not lie on the segment $\mathbf{p}_1\mathbf{a}$ (otherwise, $\mathbf{p}_1$ would be closer to $\mathbf{a}$ than to $\check{\mathbf{a}}$), and hence we have the strict triangle inequality $d(\mathbf{b},\mathbf{p}_1) > d(\mathbf{a},\mathbf{p}_1) - d(\mathbf{a},\mathbf{b})$. Thus $d(\check{\mathbf{a}},\mathbf{b}) < d(\mathbf{b},\mathbf{p}_1)$, contradicting $\mathbf{b} \in S_1$.

In (iii), we suppose for contradiction that $\check{\mathbf{a}} \neq \check{\mathbf{b}}$ and the segments $\mathbf{a}\check{\mathbf{a}}$ and $\mathbf{b}\check{\mathbf{b}}$ cross at a point $\mathbf{x}$. Then $d(\mathbf{a},\check{\mathbf{b}}) \leq d(\mathbf{a},\mathbf{x}) + d(\mathbf{x},\check{\mathbf{b}})$ and $d(\mathbf{b},\check{\mathbf{a}}) \leq d(\mathbf{b},\mathbf{x}) + d(\mathbf{x},\check{\mathbf{a}})$ by the triangle inequality, with at least one of the inequalities strict. The two right-hand sides together equal $d(\mathbf{a},\check{\mathbf{a}}) + d(\mathbf{b},\check{\mathbf{b}})$. Then $\check{\mathbf{a}}$ cannot be a nearest point of $\mathbf{a}$ or $\check{\mathbf{b}}$ cannot be a nearest point of $\mathbf{b}$. The lemma is proved. ∎

Let $\mathbf{a}$ be a boundary point of S_1. For each of the nearest points $\check{\mathbf{a}}$ of $\mathbf{a}$, we consider the angle $\alpha = \angle\check{\mathbf{a}}\mathbf{a}\mathbf{p}_1$ (measured counterclockwise; $0 < \alpha < 2\pi$). The *left nearest point* of $\mathbf{a}$ is the one with α minimum. We denote it by $\check{\mathbf{a}}_\ell$; see Figure 7.

We make the following convention: Let $\mathbf{a}$ and $\mathbf{b}$ be two points on the boundary of some S_i or T_i. We say that $\mathbf{b}$ lies to the *left* of $\mathbf{a}$ if the angle $\angle\mathbf{a}\mathbf{p}_i\mathbf{b}$, measured counterclockwise, is between 0 and π (this will always concern very close points $\mathbf{a}$ and $\mathbf{b}$, and then we see $\mathbf{b}$ on the left of $\mathbf{a}$ when looking from $\mathbf{p}_i$).

LEMMA 5.4. *Let* $\mathbf{a} \in \partial S_1$*, let* $\check{\mathbf{a}}_\ell$ *be the left nearest point of* $\mathbf{a}$*, and assume* $\check{\mathbf{a}}_\ell \in T_2$*. Then there exists a neighborhood* U *of* $\mathbf{a}$ *such that all points of* ∂S_1 *lying to the left of* $\mathbf{a}$ *and in* U *have exactly one nearest point, and moreover, this nearest point lies on* ∂T_2*, to the right of* $\check{\mathbf{a}}_\ell$ *and near to it (as near as desired if* U *is chosen small enough).*

Proof. First we note that if there are points $\mathbf{x} \in \partial S_1$ arbitrarily near to $\mathbf{a}$ that have a nearest point in some T_j, then by the continuity of the metric projection, $\mathbf{a}$ has a nearest point in T_j as well. Thus, supposing for contradiction that if an arbitrarily small left-hand neighborhood of $\mathbf{a}$ in ∂S_1 contained a point $\mathbf{x}$ with a nearest point $\check{\mathbf{x}}$ in T_j, $j \neq 2$, then $\check{\mathbf{x}}$ would get arbitrarily close to the nearest point of $\mathbf{a}$ in T_j. But then, if the neighborhood is taken sufficiently small, the segment $\mathbf{x}\check{\mathbf{x}}$ has to cross the segment $\mathbf{a}\check{\mathbf{a}}_\ell$ (it cannot cross $\mathbf{p}_1\mathbf{a}$ since $\check{\mathbf{x}}$ is visible, and it has to get very near to a point of the dotted circular arc $\mathbf{p}_1\check{\mathbf{a}}_\ell$ in Figure 7 but not too close to $\mathbf{p}_1$ or $\check{\mathbf{a}}_\ell$). This contradicts Lemma 5.3(iii). ∎

COROLLARY 5.5. *As in Lemma* 5.4*, let* $\mathbf{a} \in \partial S_1$ *and let* $\check{\mathbf{a}}_\ell \in \partial T_2$ *be the left nearest point of* $\mathbf{a}$*. Then for every neighborhood* V *of* $\check{\mathbf{a}}_\ell$ *there is a neighborhood* U *of* $\mathbf{a}$ *such that if* $\check{C}$ *denotes the portion of* ∂T_2 *lying in* V *and to the right of* $\check{\mathbf{a}}_\ell$*, and if we let* $C = \mathrm{bisect}(\mathbf{p}_1, \check{C})$*, then* $\mathbf{a} \in C$ *and the portion of* ∂S_1 *lying to the left of* $\mathbf{a}$ *and in* U *coincides with the portion of* C *lying to the left of* $\mathbf{a}$ *and in* U*.*

Proof. By Lemma 5.4 we know that the considered piece of ∂S_1 is contained in the bisector. The bisector $\mathrm{bisect}(\mathbf{p}_1, \check{C})$ is the boundary of the convex set $\mathrm{dom}(\mathbf{p}_1, \check{C})$ ([1, Lemma 3(iii)]), and so it has to coincide with the considered piece of ∂S_1 (i.e.,

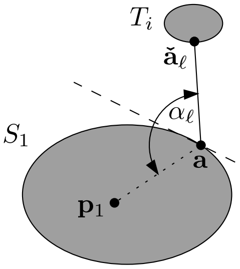


FIG. 8. *The case $\alpha_\ell < \pi$.*

there cannot be any points of C in U and not in ∂S_1, since C is a convex curve). ◻

Proof of Proposition 5.2. For contradiction let us assume $\mathbf{S} \neq \mathbf{T}$. We call $\mathbf{x}$ a *point of nonuniqueness* if $\mathbf{x} \in S_i \triangle T_i$, where $\triangle$ denotes symmetric difference.

Let

$$r_0 = \inf\{d(\mathbf{p}_i, \mathbf{x}) : \mathbf{x} \in S_i \triangle T_i,\ i = 1, 2, \ldots, n\}$$

be the infimum of distances of points of nonuniqueness to their respective sites. We have $r_0 > 0$, since each S_i and each T_i contains a disk of nonzero radius around $\mathbf{p}_i$.

We note that there is no nonuniqueness at r_0 itself; that is, $S_i \cap B(\mathbf{p}_i, r_0) = T_i \cap B(\mathbf{p}_i, r_0)$ for all i, where $B(\mathbf{p}, r)$ denotes the disk of radius r centered at $\mathbf{p}$. This is because any closed convex set in $\mathbb{R}^d$ with nonempty interior equals the closure of its interior (and we apply this to $S_i \cap B(\mathbf{p}_i, r_0)$ and $T_i \cap B(\mathbf{p}_i, r_0)$).

Clearly, there is (at least one) index i that "causes" r_0, that is, with $r_0 = d(\mathbf{p}_i, S_i \triangle T_i)$. For notational convenience we assume that $i = 1$ is such. By a simple compactness argument, we can choose a sequence $(\mathbf{x}_j)_{j=1}^{\infty}$ of points in $S_1 \triangle T_1$ with $\lim_{j\to\infty} d(\mathbf{x}, \mathbf{p}_1) = r_0$ and such that the $\mathbf{x}_j$'s converges to a point $\mathbf{a}$. For convenience we assume that all the $\mathbf{x}_j$ lie to the left of $\mathbf{a}$ when viewed from $\mathbf{p}_1$.

By possibly exchanging the roles of $\mathbf{S}$ and $\mathbf{T}$, we may assume $\mathbf{x}_j \in T_1 \setminus S_1$ for all j. Then $\mathbf{a}$ is a boundary point of S_1 since, on the one hand, it is in T_1 and T_1 coincides with S_1 up until radius r_0, and, on the other hand, it is in the closure of the complement of S_1.

Let $\breve{\mathbf{a}} \in \bigcup_{i=2}^{n} T_i$ be a nearest point of $\mathbf{a}$ and let $\breve{\mathbf{a}} \in T_{i_0}$. By Lemma 5.3(i) we have $r_0 = d(\mathbf{p}_1, \mathbf{a}) = d(\mathbf{a}, \breve{\mathbf{a}}) \geq d(\breve{\mathbf{a}}, \mathbf{p}_{i_0})$. Thus $\breve{\mathbf{a}} \in S_{i_0}$ as well, and it follows that $\mathbf{a}$ is a boundary point of T_1, too (since $T_1 \subseteq \operatorname{dom}(\mathbf{p}_1, \{\breve{\mathbf{a}}\})$).

It follows that the set of nearest points of $\mathbf{a}$ in $\bigcup_{i=2}^{n} T_i$ coincides with the set of nearest points of $\mathbf{a}$ in $\bigcup_{i=2}^{n} S_i$. Let $\breve{\mathbf{a}}_\ell$ be the left nearest point of $\mathbf{a}$. We fix notation so that $\breve{\mathbf{a}}_\ell \in \partial T_2$ (then $\breve{\mathbf{a}}_\ell \in \partial S_2$ as well).

By Corollary 5.4, a small portion of ∂S_1 to the left of $\mathbf{a}$ is uniquely determined by a small portion of ∂T_2 to the right of $\breve{\mathbf{a}}_\ell$, and similarly for T_1 and S_2. Hence by the nonuniqueness assumption, ∂S_2 and ∂T_2 cannot coincide on any small neighborhood to the right of $\breve{\mathbf{a}}_\ell$.

We will distinguish several cases. First, if $d(\breve{\mathbf{a}}_\ell, \mathbf{p}_2) < d(\mathbf{p}_1, \mathbf{a}) = r_0$, then also a small neighborhood of $\breve{\mathbf{a}}_\ell$ has distance to $\mathbf{p}_2$ smaller than r_0, and hence T_2 and S_2 coincide near $\breve{\mathbf{a}}_\ell$, which is a contradiction. From now on we thus assume $d(\mathbf{p}_2, \breve{\mathbf{a}}_\ell) = r_0 = d(\mathbf{a}, \breve{\mathbf{a}}_\ell)$.

Next, we consider the angle $\alpha_\ell = \angle \breve{\mathbf{a}}_\ell \mathbf{a} \mathbf{p}_1$; see Figure 8. We claim that $\alpha_\ell \geq \pi$. Indeed, we have $S_1, T_1 \subseteq \operatorname{dom}(\mathbf{p}_1, \{\breve{\mathbf{a}}_\ell\})$, and if $\alpha < \pi$, then this condition forces ∂S_1 and ∂T_1 in a small left-hand neighborhood of $\mathbf{a}$ to be at distance smaller than r_0 to $\mathbf{p}_1$, which contradicts the assumed nonuniqueness.

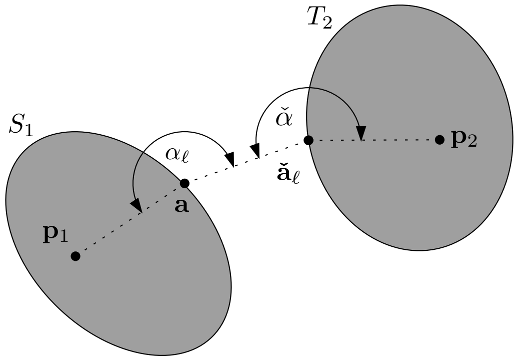


FIG. 9. *The angle $\breve{\alpha}$.*

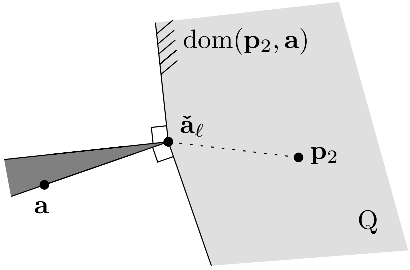

$\mathrm{dom}(\mathbf{p}_2, \mathbf{a})$
$\breve{\mathbf{a}}_\ell$
$\mathbf{p}_2$
$\mathbf{a}$
Q

FIG. 10. *The region Q.*

We also need to consider the angle $\breve{\alpha} = \angle \mathbf{p}_2 \breve{\mathbf{a}}_\ell \mathbf{a}$; see Figure 9. If $\breve{\alpha} < \pi$, then by the same argument as above, small portions of ∂T_2 and ∂S_2 to the right of $\breve{\mathbf{a}}_\ell$ coincide (since $T_2, S_2 \subseteq \mathrm{dom}(\mathbf{p}_2, \{\mathbf{a}\})$, etc.), which is a contradiction. Hence $\breve{\alpha} \geq \pi$.

We now deal with the case $\breve{\alpha} > \pi$. Here T_2 and S_2 are contained in the region Q that is the intersection of the half-plane $\mathrm{dom}(\mathbf{p}_2, \{\mathbf{a}\})$ with the half-plane h with boundary passing through $\breve{\mathbf{a}}_\ell$ and perpendicular to $\mathbf{a}\breve{\mathbf{a}}_\ell$; see Figure 10. Clearly, for any point $\mathbf{x}$ in the dark gray wedge in Figure 10, the nearest point in Q is $\breve{\mathbf{a}}_\ell$. Therefore, small portions of ∂S_1 and ∂T_1 to the left of $\mathbf{a}$ are contained in the bisector $\mathrm{bisect}(\mathbf{p}_1, \{\breve{\mathbf{a}}_\ell\})$ (a straight line), and in particular, they coincide—a contradiction finishing the case $\breve{\alpha} > \pi$.

Now we thus assume $\breve{\alpha} = \pi$. In order to proceed, we repeat for T_2, S_2, and $\breve{\mathbf{a}}_\ell$ some of the considerations made above for S_1, T_1, and $\mathbf{a}$, with left changed to right. First we can see that $\mathbf{a}$ is the right nearest point of $\breve{\mathbf{a}}_\ell$, for otherwise, small pieces of ∂T_2 and ∂S_2 to the right of $\breve{\mathbf{a}}_\ell$ would have distance at most r_0 to $\mathbf{p}_2$ and would thus coincide there, a contradiction.

Second, we can get a contradiction as above if $\alpha_\ell > \pi$: For a small piece of ∂T_2 and ∂S_2 to the right of $\breve{\mathbf{a}}_\ell$, the nearest point is $\mathbf{a}$; hence these pieces would be the same straight segment.

Finally, we are left with the situation where $\alpha_\ell = \breve{\alpha} = \pi$ (in other words, $\mathbf{p}_1$, $\mathbf{a}$, $\breve{\mathbf{a}}_\ell$, and $\mathbf{p}_2$ are collinear), $\breve{\mathbf{a}}_\ell$ is the left nearest point of $\mathbf{a}$, and $\mathbf{a}$ is the right nearest point of $\breve{\mathbf{a}}_\ell$. Let Σ be a sufficiently narrow strip with one side given by the line $\mathbf{p}_1\mathbf{p}_2$ and the other side on the left of $\mathbf{a}$, let C_1 be the component of $\Sigma \cap \partial S_1$ adjacent to $\mathbf{a}$, and similarly, let C_2 be the component of $\Sigma \cap \partial T_2$ adjacent to $\breve{\mathbf{a}}_\ell$. Then C_1 and C_2

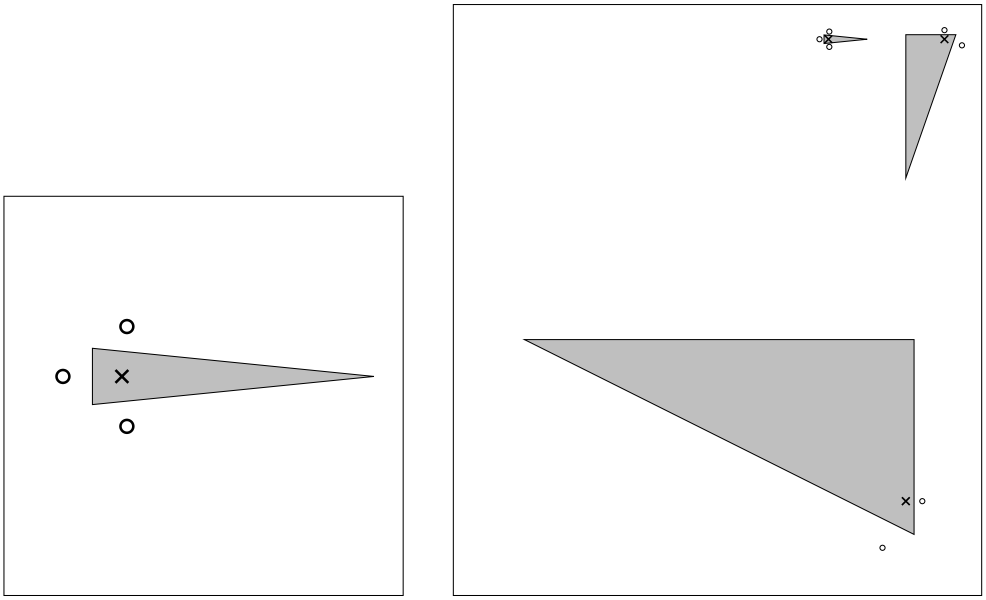


Fig. 11. *The "daggers" example, documenting the long-range influence in zone diagrams and the great sensitivity to small changes in site positions. Crosses represent sites, and small circles represent tiny flowers.*

satisfy

$$C_1 = \mathrm{bisect}(\mathbf{p}_1, C_2) \cap \Sigma, \qquad C_2 = \mathrm{bisect}(\mathbf{p}_2, C_1) \cap \Sigma.$$

By the results of [1] (a small modification of Proposition 6, with the symmetric interval $(-a, a)$ replaced by $[0, a)$, and with almost no change in the proof), C_1 and C_2 are determined uniquely by these conditions; they are the "distance trisector curves" investigated in [1]. Since we have the same property for the appropriate pieces of ∂T_1 and ∂S_2, we again get a contradiction to the assumption that S_1 and T_1 should differ in every neighborhood of $\mathbf{a}$. Proposition 5.2 is proved. □

6. Concluding remarks. Nonlocal influence and sensitivity in zone diagrams. We sketch an interesting example. The left picture in Figure 11 shows a zone diagram, a "dagger," with one isolated site $\mathbf{q}$ and three "flowers" marked by empty circles, where each flower has six sites arranged at the vertices of a tiny regular hexagon. As was observed in section 2, if the flowers are very small, the region of the isolated site is close to the classical Voronoi region of $\mathbf{q}$. In the present case it is (almost) a skinny triangle. In the right-hand picture, we have a small horizontal dagger on the top. Then there are two tiny flowers and an isolated site on the right, and these flowers plus the tip of the small horizontal dagger induce a region of the isolated site, which is also (almost) a skinny triangle. This makes a larger vertical dagger on the right. This can be iterated in a spiral-like fashion with any number of progressively larger daggers. (Of course, a formal proof that the regions truly look as claimed would be longer.)

This daggers example witnesses two things. First, the location of the tip of the last (largest) dagger depends on the location of *all* of the flowers and isolated sites. Therefore, zone diagrams possess no locality in a sense similar to classical Voronoi diagrams. Second, changing the location of one of the flowers in the first (smallest) dagger has a large influence on the position of the tip of that dagger, which in turn exerts an even much larger influence on the tip of the second dagger, and so on. We have a complicated "leverage effect," again quite unlike in classical Voronoi diagrams. We can also see that the convergence of the iterative algorithm from section 1 is likely to be relatively slow on this example: The tip of the first dagger has to stabilize very precisely before the second dagger has a chance to approach its final state, etc. Therefore, a bound on the convergence rate must take the number of sites into account, as well as some other parameter, such as the ratio of the maximum and minimum site distances.

Combinatorial complexity of zone diagrams. For a zone diagram $\mathbf{R}$, *singular points* on the boundary of a region R_i are those with at least two distinct nearest points in $\bigcup_{j\neq i} R_j$ (this notion has been considered in section 5). We could regard the singular points as an analogue of Voronoi vertices in a classical Voronoi diagram, and the segments of ∂R_i between consecutive singular points as an analogue of Voronoi edges.

It is not hard to show that each of these Voronoi edges e is contained in the bisector of $\mathbf{p}_i$ and some R_j, $j \neq i$, which easily implies that e is of class C^1 (with a continuous first derivative). We can prove that the number of singular points, as well as the number of "Voronoi edges," is $O(n)$, where n is the number of sites (we intend to publish a proof elsewhere). However, it should be noted that a single "Voronoi edge" $e \subseteq \partial R_i$ can still be complicated: We have $e \subseteq \mathrm{bisect}(\mathbf{p}_i, e')$, where e' is a piece of ∂R_j (for some j) which may contain many singular points.

Crystal growth: Intuition and alternative algorithm. Here is our original intuition for the uniqueness proof. We imagine that a crystal starts growing from each site $\mathbf{p}_i$ at time $t = 0$, and we let $R_i(t)$ denote its shape at time $t \geq 0$. Initially each of the crystals grows everywhere along its boundary at unit speed, but as soon as the distance of a boundary point $\mathbf{x} \in \partial R_i(t)$ to some $R_j(t)$ becomes $d(\mathbf{x}, \mathbf{p}_i)$, the growth at $\mathbf{x}$ stops. It seems intuitively clear that the result of this growth process should be a zone diagram, and actually the only possible zone diagram. But proving it seems to require some kind of "induction on the radius," and here the usual troubles with the continuous nature of the reals start (resembling the troubles with the intuitively obvious arguments of the old masters of calculus, arguments which were later replaced by the much more complicated-looking proofs in contemporary textbooks of analysis).

Our uniqueness proof shows that given $\mathbf{R}(t) = (R_1(t), R_2(t), \ldots, R_n(t))$ at some time t, we can uniquely extend it to $\mathbf{R}(t + \varepsilon)$ for some $\varepsilon = \varepsilon(t) > 0$, and this is even "efficient" in the sense that the new pieces of the boundary are given as bisectors of sites and old pieces, or as pieces of the distance trisector curve. We could thus start with $\mathbf{R}(t_0)$ for a suitable t_0 where all $R_i(t)$ are disks of radius t_0, extend to t_1, then to t_2, etc., but if we take the proof as is, the steps $t_{k+1} - t_k$ might possibly get smaller and smaller, and we might get an infinite but *bounded* sequence $t_0 < t_1 < t_2 < \cdots$.

It turns out that we can get away with finitely (and even polynomially) many time steps if we are willing to make the computation of a bisector of an already computed curve and a site in a single step (as well as the computation of the distance trisector curve). But such operations may be too complex to be considered as reasonable computational primitives, and further work is still needed.

7. Appendix.

7.1. Proof of Lemma 4.4. We will actually prove that F is C-Lipschitz for a (large) constant C depending on the point set P. Let $\boldsymbol{\rho}, \boldsymbol{\rho}' \in K$. Let us put $\boldsymbol{\sigma} = F(\boldsymbol{\rho})$,

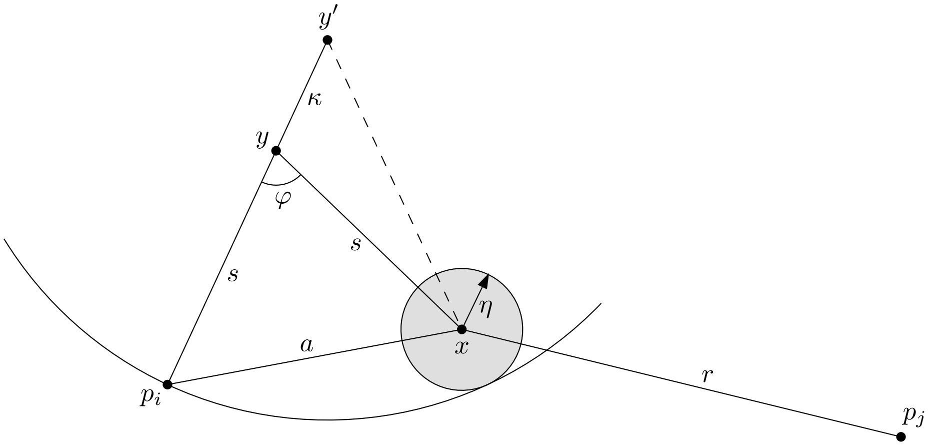


FIG. 12. *Illustration of the proof of continuity of F.*

$\boldsymbol{\sigma}' = F(\boldsymbol{\rho}')$, $\delta = \|\boldsymbol{\rho} - \boldsymbol{\rho}'\|_\infty$, $\varepsilon = \|\boldsymbol{\sigma} - \boldsymbol{\sigma}'\|_\infty$. To prove continuity, we want to show a strictly positive lower bound on δ for every $\varepsilon > 0$. (It seems that a priori we cannot assume ε small.)

Let ε be attained for i and β; that is, $\sigma_i'(\beta) - \sigma_i(\beta) = \varepsilon$ (should the sign be opposite, we interchange $\boldsymbol{\rho}$ and $\boldsymbol{\rho}'$). Let $\mathbf{y}$ be the boundary point of $S_i = \mathrm{reg}_{\mathbf{p}_i}(\sigma_i)$ in direction β, and let $s = |\mathbf{p}_i\mathbf{y}| = \tan(\sigma_i(\beta))$. See Figure 12.

Let us set $s' = \tan(\sigma_i(\beta) + \varepsilon/4)$, and let $\mathbf{y}'$ be the point of $S_i' = \mathrm{reg}_{\mathbf{p}_i}(\sigma_i')$ at distance s' from $\mathbf{p}_i$ in direction β. This choice, instead of $\mathbf{y}'$ lying on the boundary of S_i' (which looks more natural), guarantees two things: First, $\mathbf{y}'$ is at finite distance from $\mathbf{p}_i$, and second, we have $\kappa = s' - s \leq s$. To verify the latter claim, we note that we may assume that $\varepsilon = \frac{\pi}{2} - \arctan s$, and we use the mean value theorem to bound $s' - s = \tan(\arctan s + \varepsilon/4) - \tan(\arctan s) = \frac{\varepsilon}{4} \cdot \frac{1}{\cos^2(\arctan s + \varepsilon/4)} = \frac{\varepsilon}{4} \cdot \frac{1}{\sin^2(3\varepsilon/4)}$. Now $\varepsilon \leq \pi/2 - \arctan 1 = \pi/4$, and since for $0 \leq x \leq \frac{3}{16}\pi$ we have $\sin x \geq 0.9x$, we obtain $s' - s \leq \frac{\varepsilon}{4} \cdot \frac{1}{(0.9 \cdot 0.75\varepsilon)^2} \leq \frac{0.6}{\varepsilon}$. On the other hand, $s = \tan(\frac{\pi}{2} - \varepsilon) = \frac{1}{\tan \varepsilon} \geq \frac{\pi}{4\varepsilon} > \frac{0.6}{\varepsilon}$, and $s' - s \leq s$ is proved.

Since $\mathbf{y}$ is a boundary point of S_i, it is easy to see that there has to be a boundary point $\mathbf{x}$ of some R_j, $j \neq i$ (where $(R_1, \ldots, R_n) = \mathbf{reg}(\boldsymbol{\rho})$), such that $|\mathbf{xy}| = |\mathbf{yp}_i| = s$ (briefly, points arbitrarily close to $\mathbf{y}$ but outside S have points of some R_j at distance arbitrarily close to s, and a limit argument using compactness provides the desired $\mathbf{y}$). On the other hand, the open disk of radius $s' = s + \kappa$ centered at $\mathbf{y}'$ is disjoint from all R_k', $k \neq i$, for otherwise, $\mathbf{y}'$ would not lie in S_i'.

If we prove some lower bound η on the difference $s' - |\mathbf{y}'\mathbf{x}|$, then the open disk of radius η centered at $\mathbf{x}$ is disjoint from all R_j', and in particular, the point $\mathbf{x}'$ on the segment $\mathbf{p}_j\mathbf{x}$ lying at distance η from $\mathbf{x}$ cannot be inside R_j'. It follows that

$$\delta = \|\boldsymbol{\rho} - \boldsymbol{\rho}'\|_\infty \geq \arctan(r) - \arctan(r - \eta) \geq \frac{\eta}{1 + r^2} \geq \frac{\eta}{2r^2}$$

by the mean value theorem (we have $r - \eta \geq 1$ since R_j' contains the unit disk centered at $\mathbf{p}_j$).

To estimate η, we consider the triangle $\triangle \mathbf{y}'\mathbf{y}\mathbf{x}$, and by the cosine theorem we obtain

$$|y'x| = \sqrt{s^2 + \kappa^2 - 2s\kappa\cos(\pi - \varphi)}$$

$$
\begin{aligned}
&= \sqrt{s^2+\kappa^2+2s\kappa\cos\varphi}\\
&= s'\sqrt{1-\frac{2s\kappa}{s'^2}(1-\cos\varphi)}\\
&\le s'\left(1-\frac{s\kappa}{s'^2}(1-\cos\varphi)\right)\\
&= s'-\frac{s\kappa}{s'}(1-\cos\varphi),
\end{aligned}
$$

where we have used $\sqrt{1-z} \le 1-z/2$ for $0<z<1$. The cosine theorem for the triangle $\triangle\mathbf{p}_i\mathbf{y}\mathbf{x}$ then yields

$$a^2 = |\mathbf{p}_i\mathbf{x}|^2 = 2s^2(1-\cos\varphi),$$

and altogether we have

$$\eta = s' - |\mathbf{y}'\mathbf{x}| \ge \frac{a^2\kappa}{2ss'} \ge \frac{a^2}{4s^2}\kappa$$

(using the inequality $s' \le 2s$ mentioned above).

For a we use the obvious estimate $a \ge 1$ (from the fact that S_i contains the unit disk centered at $\mathbf{p}_i$), as well as $a = |\mathbf{p}_i\mathbf{y}| \ge |\mathbf{y}\mathbf{p}_j| - |\mathbf{p}_i\mathbf{p}_j| \ge r-\Delta$, where Δ denotes the diameter of P. Together we have $a \ge \max(1, r-\Delta) \ge r/2\Delta$ (distinguishing the cases $r \le 2\Delta$ and $r \ge 2\Delta$). Finally, for $\kappa = s'-s$ we have $\arctan(s')-\arctan(s) = \varepsilon/4$, and the mean value theorem (as usual) gives $\kappa = s'-s \ge \frac{1}{4}\varepsilon\cdot(1+s^2) \ge \frac{1}{4}\varepsilon s^2$.

Putting the chain of inequalities together, we have

$$\|\boldsymbol{\rho}-\boldsymbol{\rho}'\|_\infty = \delta \ge \frac{\eta}{2r^2} \ge \frac{a^2}{4s^2\cdot 2r^2}\cdot\frac{\varepsilon}{4}s^2 \ge \frac{r^2}{4\Delta^2\cdot 8r^2}\cdot\frac{\varepsilon}{4} = \frac{\varepsilon}{128\Delta^2}.$$

This shows that the operator **Dom** is continuous and even C-Lipschitz for a suitable constant C. □

Acknowledgments. We would like to thank Tomáš Kaiser for useful suggestions and discussions concerning fixed-point theorems and Eva Kopecká for suggesting an alternative (and simpler) approach to the existence proof.

REFERENCES

[1] T. Asano, J. Matoušek, and T. Tokuyama, *The distance trisector curve*, Adv. Math., 212 (2007), pp. 338–360.

[2] T. Asano and T. Tokuyama, *Drawing equally-spaced curves between two points*, in Proceedings of the Fall Conference on Computational Geometry, Boston, 2004, MIT, Cambridge, MA, pp. 24–25.

[3] F. Aurenhammer, *Voronoi diagrams—A survey of a fundamental geometric data structure*, ACM Computing Surveys, 23 (1991), pp. 345–405.

[4] D. W. Curtis and R. M. Schori, *Hyperspaces of Peano continua are Hilbert cubes*, Fund. Math., 101 (1978), pp. 19–38.

[5] A. Okabe, B. Boots, and K. Sugihara, *Spatial Tessellations, Concepts and Applications of Voronoi Diagrams*, John Wiley & Sons, New York, 1992.

[6] E. Zeidler, *Nonlinear functional analysis and its applications.* I: *Fixed-point theorems*, 2nd corr. printing, Springer, New York, 1993.

SIAM J. COMPUT.
Vol. 37, No. 4, pp. 1199–1227

SAMPLING AND MESHING A SURFACE WITH GUARANTEED TOPOLOGY AND GEOMETRY*

SIU-WING CHENG†, TAMAL K. DEY‡, EDGAR A. RAMOS§, AND TATHAGATA RAY‡

Abstract. This paper presents an algorithm for sampling and triangulating a generic C^2-smooth surface $\Sigma \subset \mathbb{R}^3$ that is input with an implicit equation. The output triangulation is guaranteed to be homeomorphic to Σ. We also prove that the triangulation has well-shaped triangles, large dihedral angles, and a small size. The only assumption we make is that the input surface representation is amenable to certain types of computations, namely, computations of the intersection points of a line and Σ, computations of the *critical points* in a given direction, and computations of certain *silhouette points*.

Key words. smooth surface, geometry, topology, Voronoi diagram, Delaunay triangulation, Delaunay refinement

AMS subject classifications. 65D18, 65D17, 68W05, 68W30, 68W40

DOI. 10.1137/060665889

1. Introduction.

The need for triangulating a surface is ubiquitous in science and engineering. A set of points needs to be sampled from the input surface and then connected to generate such a triangulation. The underlying space of the resulting triangulation should have the same topology as that of the input surface. Nice geometric properties such as bounded aspect ratio and large dihedral angles are also desirable. The input surface can be specified in various ways and each leads to a different problem of surface triangulation.

When the surface is given by a set of point samples, the problem is known as *surface reconstruction* for which algorithms with topological and geometrical guarantees have been proposed [1, 2, 5, 19]. When the surface is polyhedral, i.e., made out of planar patches, the Delaunay refinement techniques solve the problem elegantly [12, 13, 14, 16, 32].

The case in which the input surface is smooth and specified by an implicit equation occurs in a variety of applications in geometric modeling, computer graphics, and finite element methods [34, 36, 33]. Obtaining a surface triangulation that has the correct topology and nice geometric properties (e.g., bounded aspect ratio and large dihedral angles) is an important issue in these applications. In this paper we present an algorithm to triangulate a generic C^2-smooth implicit surface without boundary. All vertices are sampled from the input surface, and the following guarantees are offered for any fixed $\lambda \leq 0.06$:

*Received by the editors July 24, 2006; accepted for publication (in revised form) May 10, 2007; published electronically November 21, 2007. This research was partially supported by NSF grants DMS-0310642, CCR-0430735, and by Research Grant Council, Hong Kong, China (HKUST 6181/04E). A preliminary version of the paper appeared in *Proceedings of the 20th Annual ACM Symposium on Computational Geometry*.

http://www.siam.org/journals/sicomp/37-4/66588.html

†Department of Computer Science and Engineering, HKUST, Hong Kong, People's Republic of China (scheng@cse.ust.hk).

‡Department of Computer Science and Engineering, Ohio State University, Columbus, OH 43210 (tamaldey@cse.ohio-state.edu, rayt@cse.ohio-state.edu).

§Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana-Champaign, IL 61801 (eramosn@cs.uiuc.edu).

(i) The underlying space of the output triangulation is homeomorphic to the input surface.

(ii) All angles of the triangles are at least arcsin $(\frac{1}{2+2\lambda})$.

(iii) The dihedral angle between adjacent triangles is at least $\pi - O(\lambda)$.

(iv) The number of vertices is no more than $O(\frac{\varepsilon^2}{\lambda^2})$ times the size of an ε-sample[1] for any $\varepsilon < \frac{1}{5}$.

Although the output triangulation does not have bounded Hausdorff distance from the input surface as enjoyed by an ε-sample [1, 2], property (iv) shows that our algorithm does not sample an excessive number of vertices.

Because of the importance of the problem, a number of algorithms have been proposed for meshing implicit surfaces across various application areas. These algorithms may give satisfactory experimental results, but they do not have guarantees on the validity of the topology and/or the quality of the output triangulation. We briefly survey a selected subset. The problem of triangulating implicit surfaces has been investigated by Bajaj [3], Bloomenthal [4], Tristano, Owen, and Canann [35], Lau and Lo [28], and Cuillière [17]. The marching cube algorithm of Lorensen and Cline [29] can be used to triangulate an implicit surface. The algorithm determines the edges of a cubic grid intersecting the surface and then generates a tessellation by connecting these intersection points. Although the algorithm is very simple, there is no guarantee that the output has the same topology as that of the surface. Stander and Hart [34] proposed varying the value of the implicit function from $-\infty$ to ∞ and dynamically maintaining a triangulation of the changing isosurface. It is necessary to track all critical points of the implicit function. Maintaining triangulations of isosurfaces is a huge overhead given that only one isosurface, namely, the input, needs to be triangulated. Witkin and Heckbert [36] proposed spreading particles governed by differential equations on the implicit surface. At equilibrium, the particles can be connected to form the surface triangulation, but it is unclear how to ensure that the surface topology is captured. The above algorithms do not offer any guarantee on the triangle shape though some of them include heuristics and illustrate their effectiveness experimentally.

In computational geometry, Chew [15] described an algorithm that is based on the "furthest-point" strategy: among the intersections between a Voronoi edge and the input surface, select and insert the furthest one from the sites defining the Voronoi edge. In effect, this algorithm attempts to compute the *restricted Delaunay triangulation* of the surface. Edelsbrunner and Shah [24] showed that a *topological ball property* is sufficient for the restricted Delaunay triangulation to be homeomorphic to the input surface. The algorithm of Chew does not guarantee this property or any other that ensures topological correctness.

Following the furthest-point strategy, Cheng, Dey, Edelsbrunner, and Sullivan [11] proposed an algorithm for triangulating the *skin surface* [22] that provides both topological and geometric guarantees. This algorithm exploits the fact that the *local feature size* is easily computable for skin surfaces. The local feature size of a point x on the surface is the distance from x to the medial axis.

Boissonnat and Oudot [8] carried forward the furthest-point strategy for general curved surfaces. They showed how to grow from an initial seed triangle on each surface component to a full triangulation with topological and geometric guarantees. The algorithm assumes that one can compute the local feature size of any point on the surface. Computing the medial axis is hard, and hence computing the local feature

[1] This is defined in section 2.1.

size exactly is difficult, if not impossible, for surfaces in general. Of course, one can approximate the medial axis with existing algorithms [1, 5, 18]. However, these algorithms require a dense sampling with respect to the local feature size in the first place. An alternative suggested by Boissonnat and Oudot is to compute the minimum local feature size and then to run their algorithm to obtain a dense sample from which the medial axis can be approximated. In a second pass a new mesh can be computed with appropriate density using the approximated local feature size.

A related work by Boissonnat, Cohen-Steiner, and Vegter [7] considered triangulating the isosurfaces of a function $E : \mathbb{R}^3 \to \mathbb{R}$. Their method evaluates E at grid points and triangulates a box in $\mathbb{R}^3$ recursively to provide a piecewise linear interpolant $\widehat{E}$ of E. The isosurface $E = 0$ is approximated with the isosurface $\widehat{E} = 0$. The authors provide conditions on sampling to guarantee that the computed surface $\widehat{E} = 0$ is isotopic to the surface $E = 0$. Their method samples the function E rather than the surface $E = 0$. Moreover, it computes the critical points of E as well as their indices. There are two other algorithms for producing isotopic triangulations, one by Mourrain and Técourt [30] and another by Plantinga and Vegter [31]. More details about these isotopic triangulation algorithms can be found in the survey [6].

In this paper, we eliminate the need for local feature size computation. We show that it suffices to identify the critical points and a silhouette of the surface or a cross-section of the surface in a given direction. These computations are less demanding. (The critical points of the surface in a given direction should not be confused with the critical points of the implicit function as computed by the algorithms of Stander and Hart [34] and Boissonnat, Cohen-Steiner, and Vegter [7].) To this end, we depart from the strategy of Boissonnat and Oudot [8, 9] in a fundamental way. Topological ball property violations drive the refinement in our algorithm, whereas they are used only for analysis in [8, 9].

Our algorithm incrementally grows a set of point samples and maintains the restricted Delaunay triangulation of the samples. In the topology recovery phase, we use a simple "topological-disk" test and certain critical and silhouette point computations to guide the sampling of points from the surface. Our approach extends the "furthest-point" strategy that selects samples only from the intersections between Voronoi edges and the surface. This extension allows us to sample points adaptively and prove that the output triangulation has the same topology as that of the surface. In the geometry recovery phase, we enforce the bounded aspect ratio and smoothness of the surface triangulation.

2. Preliminaries.

2.1. Input surface and assumptions.

The input is a compact surface $\Sigma \subset \mathbb{R}^3$ without boundary. We assume that Σ is specified as the zero-level set of a function $E : \mathbb{R}^3 \to \mathbb{R}$ such that the second partial derivatives of E are continuous (i.e., Σ is C^2-smooth), and $E(x)$ and $\operatorname{grad} E(x)$ do not vanish simultaneously at any $x \in \mathbb{R}^3$.

A maximal ball whose interior is disjoint from Σ is called a *medial ball*. The *medial axis* of Σ is the set of the centers of medial balls. We borrow some definitions from Amenta and Bern [1] who proposed them in the context of surface reconstruction. The *local feature size* $f(x)$ at a point $x \in \Sigma$ is the distance from x to the medial axis. Since Σ is assumed to be C^2-smooth and compact, $\min_{x \in \Sigma} f(x)$ is nonzero. The function f is 1-Lipschitz; that is, $f(x) \leq f(y) + \|x - y\|$ for any two points x and y on Σ. A point set $S \subset \Sigma$ is an ε-sample if for any $x \in \Sigma$, there is a point $p \in S$ such that $\|p - x\| \leq \varepsilon f(x)$.

Throughout this paper, we use x_1, x_2, and x_3 to denote the three orthogonal directions forming the coordinate frame. Given any point $y \in \mathbb{R}^3$, we use (y_1, y_2, y_3) to denote the coordinates of y. For any vector d in $\mathbb{R}^3$, we use (d_1, d_2, d_3) to denote its components in the x_1-, x_2-, and x_3-directions. Given two vectors d and d', let $\langle d, d' \rangle$ and $d \times d'$ denote their inner and cross products, respectively. The gradient $\operatorname{grad} E(x)$ is the vector $(\frac{\partial E(x)}{\partial x_1}, \frac{\partial E(x)}{\partial x_2}, \frac{\partial E(x)}{\partial x_3})$. If the point x lies on Σ, $\operatorname{grad} E(x)$ is parallel to the unit outward surface normal at x. The *critical points of* Σ *in a direction* d are the points $x \in \Sigma$ such that $\operatorname{grad} E(x)$ is parallel to d.

We assume that Σ has finitely many critical points in any direction. We also assume that the Hessian at any point $x \in \Sigma$ is nonsingular; i.e., for any two orthogonal tangent directions u_1 and u_2 at $x \in \Sigma$, the matrix $(\frac{\partial^2 E(x)}{\partial u_i \, \partial u_j})$ is nonsingular at x.

A unit vector d induces a height function h_d on Σ: $h_d(x) = \langle \operatorname{grad} E(x), d \rangle$ for any $x \in \Sigma$. The set $h_d^{-1}(0)$ consists of the points $x \in \Sigma$ such that $\operatorname{grad} E(x)$ is orthogonal to d; i.e., d is a tangent direction at x. Let v be a tangent direction at x orthogonal to d. Orient space so that d aligns with the x_1-axis and v aligns with the x_2-axis. We have $h_d(x) = \frac{\partial E(x)}{\partial d}$ and thus $(\frac{\partial h_d(x)}{\partial d}, \frac{\partial h_d(x)}{\partial v}) = (\frac{\partial^2 E(x)}{\partial d^2}, \frac{\partial^2 E(x)}{\partial d \, \partial v}) \neq 0$ because the Hessian is assumed to be nonsingular at $x \in \Sigma$. In other words, the points in $h_d^{-1}(0)$ are not critical for h_d, and thus it follows from the inverse function theorem in differential topology [26] that $h_d^{-1}(0)$ is a collection of smooth closed curves. We call these curves the *silhouette of* Σ *with respect to* d and denote it by J_d.

Take any direction d' orthogonal to d. A point $x \in J_d$ is critical in direction d' if the tangent to J_d at x is orthogonal to d'. We assume that J_d has finitely many critical points in direction d'. Given a plane Π, the critical points of the intersection curve(s) in $\Sigma \cap \Pi$ in any direction parallel to Π can be similarly defined. We also assume that the intersection curve(s) between Σ and any plane Π have finitely many critical points in any direction parallel to Π.

2.2. Generic intersection and topological ball property. Let S be a finite point set in $\mathbb{R}^3$. The *Voronoi cell* of a point $p \in S$ is defined as $V_p = \{ x \in \mathbb{R}^3 : \forall q \in P, \|p - x\| \leq \|q - x\| \}$. A Voronoi cell is a convex polyhedron. For $2 \leq j \leq 4$, the closed faces shared by j Voronoi cells are called $(4 - j)$-dimensional *Voronoi faces.* The 0-, 1-, and 2-dimensional Voronoi faces are called *Voronoi vertices, edges, and facets*, respectively. The *Voronoi diagram* $\operatorname{Vor} S$ is the collection of all Voronoi faces.

Assuming general position, the convex hull of $j \leq 4$ points in S defines a $(j-1)$-dimensional *Delaunay simplex* σ if the vertices of σ define a $(4 - j)$-dimensional Voronoi face in $\operatorname{Vor} S$. We use V_σ to denote this Voronoi face. We call σ and V_σ the *dual* of each other. The 1-, 2-, and 3-dimensional Delaunay simplices are called *Delaunay edges, triangles, and tetrahedra*, respectively. The Delaunay simplices define a decomposition of the convex hull of S called the *Delaunay triangulation* of S. We denote it by $\operatorname{Del} S$.

The set of Delaunay simplices whose dual Voronoi faces intersect Σ form the *restricted Delaunay triangulation* $\operatorname{Del} S|_\Sigma$ of S with respect to Σ. Formally, $\operatorname{Del} S|_\Sigma = \{ \sigma \in \operatorname{Del} S : V_\sigma \cap \Sigma \neq \emptyset \}$. For any simplex $\sigma \in \operatorname{Del} S|_\Sigma$, the intersection $V_\sigma \cap \Sigma$ is called a *restricted Voronoi face.* The *restricted Voronoi diagram* $\operatorname{Vor} S|_\Sigma$ is the collection of all restricted Voronoi faces.

We say that a Voronoi face V_σ satisfies the *generic intersection property* (GIP) if either $V_\sigma \cap \Sigma = \emptyset$ or V_σ intersects Σ *transversally*, that is, the *affine space* of V_σ is not tangent to Σ at the points in $V_\sigma \cap \Sigma$. In particular, this implies that a Voronoi vertex should not lie on Σ. The Voronoi diagram $\operatorname{Vor} S$ satisfies GIP for Σ if all of its faces satisfy GIP.

We say that a Voronoi face V_σ satisfies the *topological ball property* (TBP) if either $V_\sigma \cap \Sigma = \emptyset$ or the restricted Voronoi face $V_\sigma \cap \Sigma$ is a closed topological ball of dimension $\dim(V_\sigma) - 1$, where $\dim(V_\sigma)$ is the dimension of V_σ. The Voronoi diagram $\operatorname{Vor} S$ satisfies TBP for Σ if all its Voronoi faces satisfy TBP. Our meshing algorithm is based on the following result of Edelsbrunner and Shah that relates the topology of $\operatorname{Del} S|_\Sigma$ to Σ.

THEOREM 2.1 (see [24]). *The underlying space of* $\operatorname{Del} S|_\Sigma$ *is homeomorphic to* Σ *if* $\operatorname{Vor} S$ *satisfies TBP and GIP for* Σ.

Notice that the above theorem is originally proved for a nondegenerate point set; that is, no five points are cospherical. However, one may drop this requirement by appealing to the simulation of simplicity (SOS) technique [23] that simulates generic conditions by perturbing the points symbolically.

2.3. Background results. We state a few geometric results in the literature that we use frequently. Let ℓ and ℓ' be two line segments, vectors, or lines. We use $\angle \ell, \ell'$ to denote the acute angle between the support lines of ℓ and ℓ'. For any point $x \in \Sigma$, we use n_x to denote the unit outward surface normal at x. For any triangle pqr, we use n_{pqr} to denote a unit normal to pqr. Define two functions $\alpha(\lambda)$ and $\beta(\lambda)$ where

$$\alpha(\lambda) = \frac{\lambda}{1 - 3\lambda} \qquad \text{and} \qquad \beta(\lambda) = \alpha(2\lambda) + \arcsin\lambda + \arcsin\left(\frac{2\sin(2\arcsin\lambda)}{\sqrt{3}}\right).$$

The key property of $\alpha(\lambda)$ and $\beta(\lambda)$ is that both are $O(\lambda)$ and approach zero as λ does.

LEMMA 2.2 (see [1]). *Let x and y be two points on Σ. If $\|x - y\| \leq \lambda f(x)$ for some $\lambda < 1/3$, $\angle n_x, n_y \leq \alpha(c)$.*[2]

LEMMA 2.3 (see [2]). *Let pqr be a triangle with vertices on Σ. If the circumradius of pqr is less than $\lambda f(p)$ for $\lambda \leq \frac{1}{\sqrt{2}}$, then $\angle n_{pqr}, n_p \leq \beta(\lambda)$.*

LEMMA 2.4 (see [11]). *Let x and y be two points in the intersection of a line ℓ and Σ. Then $\|x - y\| \geq 2f(x)\cos(\angle \ell, n_x)$.*

3. Overview. Our algorithm's goal is to obtain a sufficiently dense sample S on Σ for which GIP and TBP hold. The approach used to obtain S is incremental. First, we initialize S to contain the critical points of Σ in the x_3-direction. Then, while either GIP or TBP does not hold, we add one more point to S: a *witness* to the *violation.* The correctness of the algorithm is then trivial as long as it halts, which follows from a lower bound on how close a new inserted point can be to the existing points.

As it turns out, it is not so easy to identify a TBP violation for a facet or cell. Therefore we take a conservative approach: a witness is always returned if there is a violation, but a witness may also be returned even in some cases where there is no violation. However, we guarantee that no harm is done in inserting the false witnesses since these false witnesses are also far away from existing points in S.

A GIP is violated when the affine space of a Voronoi edge or facet is tangent to Σ or when a Voronoi vertex lies on Σ. Notice that any tangential contact between Σ and a Voronoi edge or facet is isolated by our assumption that Σ is generic. The case of the affine space of a Voronoi edge or facet being tangent to Σ can be handled simply by returning the tangency point as witness, which can be determined by solving an

[2] The slightly stronger condition of $\|x - y\| \leq \lambda \min\{f(x), f(y)\}$ is stated in [1], but the proof uses $\|x - y\| \leq \lambda f(x)$ only.

appropriate system of equations. We will show that this witness is sufficiently far from the existing sample points. In contrast, a Voronoi vertex lying on Σ can happen at any sampling density. It is a degenerate intersection between Voronoi facets and cells with Σ. The algorithm pretends that Σ is perturbed locally to get around the degeneracy. Nevertheless, we have to deal with the Voronoi vertices on Σ directly in the proofs. The reason is that a local perturbation may change the local feature size a lot. Since we need to obtain lower bounds on distances in terms of the local feature size with respect to Σ, we cannot assume the local perturbation in the analysis.

It is natural to test for TBP violations in increasing order of Voronoi face dimensions. Testing a TBP violation at an edge is comparatively easier than at a facet or a cell. We assume GIP for all edges and facets:

Edge e. Testing for a TBP violation at an edge e is simply a matter of counting the number of intersections between e and Σ, which is easily determined by computing all intersections between e and Σ. If there is a violation, the witness returned is the intersection point furthest from any sample point in S that generates e.

Facet F. It is assumed that TBP holds for edges. $F \cap \Sigma$ is a collection of closed or open curves (with endpoints in the boundary of F). TBP is violated if $F \cap \Sigma$ includes either more than one open curve or a closed curve. In the first case, there are more than two intersections between Σ and the boundary of F, and the witness returned is the furthest intersection point from any sample point in S that generates F. In the second case, checking whether $F \cap \Sigma$ contains a closed curve is not easy, so we settle for a necessary witness instead. We find the critical points of $F \cap \Sigma$ in some direction parallel to F. Then we compute the lines in the plane of F that are normal to $F \cap \Sigma$ at these critical points. If any such line intersects $F \cap \Sigma$ in two or more points, the furthest one is returned as the witness. Clearly, such a witness exists if $F \cap \Sigma$ contains a closed curve. While this witness may exist even if there is no closed curve, it will be shown that in either case the witness is sufficiently far from the existing sample points.

Cell V_p. It is assumed that TBP holds for edges and facets. $V_p \cap \Sigma$ is a collection of surface patches with or without boundaries. There is a violation to TBP if $V_p \cap \Sigma$ is not a topological disk. In particular, a violation occurs if the boundary of $V_p \cap \Sigma$ consists of more than one closed curve. This is easily determined by checking whether the dual triangles incident to p form one or more topological disks. $V_p \cap \Sigma$ cannot contain a surface component without any boundary; otherwise, we would have placed the critical points of this component in the x_3-direction as seeds inside V_p. The hardest case is that the boundary of $V_p \cap \Sigma$ is a single closed curve, but $V_p \cap \Sigma$ has positive genus. We handle this case using the silhouette J_{n_p}, i.e., the set of points $x \in \Sigma$ such that n_x is orthogonal to n_p. It is known that J_{n_p} is a smooth closed curve and the points in J_{n_p} are far from p. So the test can be done as follows: either J_{n_p} intersects some facet of V_p at a point w, or J_{n_p} contains an extreme point w in a direction orthogonal to n_p. The point w is returned as the witness in either case.

Topology recovery as described above generates a sample of the surface. This sample is further refined to ensure some geometric properties of the output triangulation such as aspect ratios of triangles and large dihedral angles at the edges. The rest of the paper is organized as follows. In section 4, we present the analytic tools to cope with the violations of GIP and TBP. In section 5, the details of the topology recovery part of our algorithm are given. Figure 1 illustrates the dependencies between the lemmas in sections 2.3 and 4, the subroutines in section 5, and TBP. In section 6, we show how to enforce bounded aspect ratio and large dihedral angles by inserting

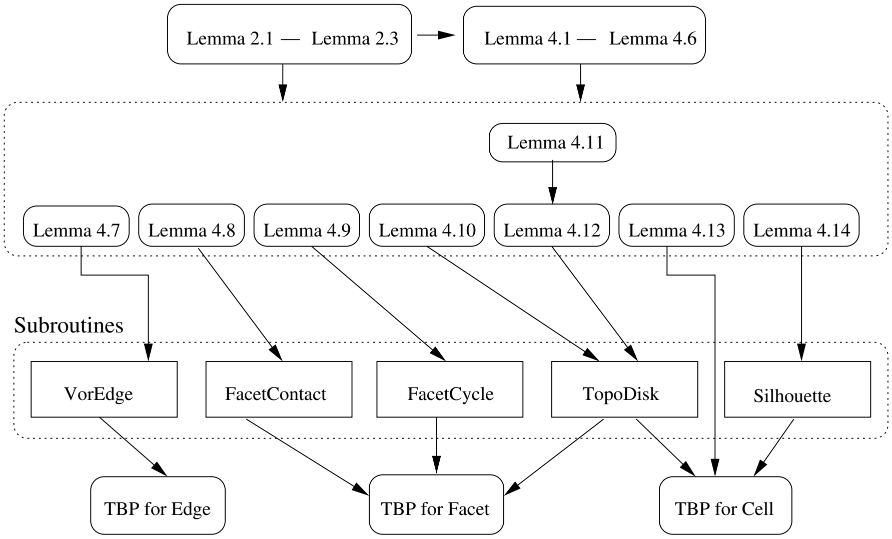


FIG. 1. *Dependencies between the subroutines, lemmas, and TBP.*

new points that are far from existing sample points. After inserting some point(s) to repair the geometry, we have to rerun the topology recovery part because GIP or TBP may no longer hold. Thus, the entire algorithm alternates between repairing topology and geometry. The full analysis of the algorithm is presented in section 6.

4. Violation. Theorem 2.1 is our main tool for recovering topology. The following subsections treat separately the violations of TBP for the cases of V_σ being a Voronoi edge, facet, and cell. In each case, we show how to identify a point $x \in V_\sigma \cap \Sigma$ such that $\|p - x\| \geq \lambda f(p)$ for any vertex p of σ where

$$\begin{aligned} \frac{\lambda}{1-\lambda} &< \cos(\alpha(\lambda) + 3\beta(\lambda)), \\ \alpha(\lambda) + \beta(\lambda) &< \pi/3, \\ \arccos\lambda &> \alpha(\lambda) + \beta(\lambda). \end{aligned}$$

The above inequalities hold for any $\lambda \leq \lambda_0 = 0.06$, which we assume throughout the rest of the paper.

The GIP may be violated if some Voronoi vertex lies on Σ. We defer this discussion to section 5.1. The results in this section hold regardless of whether some Voronoi vertex lies on Σ.

4.1. Technical results. We first prove a few technical results that we use later.

LEMMA 4.1. *Let p and x be two points on Σ. If $\|p - x\| \leq \lambda f(p)$, then $f(x) \geq (1 - \lambda) f(p)$.*

Proof. Because f is 1-Lipschitz, $f(x) \geq f(p) - \|p - x\| \geq (1 - \lambda) f(p)$. □

LEMMA 4.2. *Let p, x, and y be three points on Σ. If both $\|p - x\|$ and $\|p - y\|$ are at most $\lambda f(p)$, then $\|x - y\| \leq 2\lambda f(p) \leq 2\lambda f(x)/(1 - \lambda)$.*

Proof. By the triangle inequality, $\|x - y\| \leq \|p - x\| + \|p - y\| \leq 2\lambda f(p)$, which is at most $2\lambda f(x)/(1 - \lambda)$ by Lemma 4.1. □

LEMMA 4.3. *Let e be an edge of a Voronoi cell V_p. For any point $x \in e \cap \Sigma$, if $\|p - x\| \leq \lambda f(p)$, then $\angle e, n_x \leq \alpha(\lambda) + \beta(\lambda) < \pi/3$.*

Proof. By Lemma 2.2, $\angle n_p, n_x \leq \alpha(\lambda)$. The circumradius of the Delaunay triangle dual to e is at most $\|p - x\| \leq \lambda f(p)$. By Lemma 2.3, $\angle e, n_p \leq \beta(\lambda)$. Thus, $\angle e, n_x \leq \angle e, n_p + \angle n_p, n_x \leq \alpha(\lambda) + \beta(\lambda)$, which is less than $\pi/3$ as $\lambda \leq \lambda_0$. □

Lemma 4.4. *Let F be a facet of a Voronoi cell V_p. Let Π be the plane of F. Suppose that $\Pi \cap \Sigma$ contains a point x such that $\|p - x\| \leq \lambda f(p)$.*

(i) *The acute angle between Π and n_p is at most* $\arcsin \lambda < \beta(\lambda)$.

(ii) *The acute angle between Π and n_x is at most* $\alpha(\lambda) + \arcsin \lambda < \alpha(\lambda) + \beta(\lambda) < \pi/3$.

Proof. Let pq be the Delaunay edge dual to F. We have $\|p - q\| \leq \|p - x\| + \|q - x\| = 2\,\|p - x\| \leq 2\lambda f(p)$. It then follows from Lemma 2.4 that $\angle pq, n_p \geq \arccos \lambda$, which implies (i). By Lemma 2.2, $\angle n_p, n_x \leq \alpha(\lambda)$. By the triangle inequality, $\angle pq, n_x \geq \angle pq, n_p - \angle n_p, n_x \geq \arccos \lambda - \alpha(\lambda)$. So the acute angle between Π and n_x is at most $\alpha(\lambda) + \arcsin(\lambda) < \alpha(\lambda) + \beta(\lambda)$, which is less than $\pi/3$ as $\lambda \leq \lambda_0$. □

Lemma 4.5. *Let F be a facet of a Voronoi cell V_p such that $F \cap \Sigma$ contains no tangential contact point. Let x be a point in $F \cap \Sigma$. Let L be a line in the plane of F through x and normal to $F \cap \Sigma$ at x. If $\|p - x\| \leq \lambda f(p)$, then $\angle L, n_x < \alpha(\lambda) + \beta(\lambda)$.*

Proof. Since L is normal to $F \cap \Sigma$ at x, n_x lies in the plane containing L perpendicular to F. This implies that $\angle L, n_x$ is the acute angle between F and n_x, which is less than $\alpha(\lambda) + \beta(\lambda)$ by Lemma 4.4(ii). □

Lemma 4.6. *Let p be a sample point. Let x and y be two points on Σ. If $\angle xy, n_x \leq \alpha(\lambda) + \beta(\lambda)$, then $\|p - x\|$ or $\|p - y\|$ is greater than $\lambda f(p)$.*

Proof. Assume to the contrary that both $\|p - x\|$ and $\|p - y\|$ are at most $\lambda f(p)$. By Lemma 4.2, $\|x - y\| \leq 2\lambda f(x)/(1 - \lambda)$. On the other hand, by Lemma 2.4, $\|x - y\| \geq 2f(x)\cos(\angle xy, n_x) \geq 2f(x)\cos(\alpha(\lambda) + \beta(\lambda))$. The lower and upper bounds on $\|x - y\|$ together yield $\lambda/(1 - \lambda) \geq \cos(\alpha(\lambda) + \beta(\lambda))$, contradicting $\lambda \leq \lambda_0$. □

4.2. Violation at Voronoi edges. If a Voronoi edge violates GIP or TBP, it intersects Σ in some point far away from existing sample points. The next lemma establishes this fact.

Lemma 4.7. *Let e be an edge of a Voronoi cell V_p.*

(i) *If $e \cap \Sigma$ contains two points, the distance between p and the further one is at least $\lambda f(p)$.*

(ii) *If the support line of e meets Σ tangentially at a point $x \in e \cap \Sigma$, then $\|p - x\| \geq \lambda f(p)$.*

Proof. Consider the case in which $e \cap \Sigma$ contains two points x and y (Figures 2(a) and 2(b)). If $\|x - y\| \geq \lambda f(p)$, we are done. If $\|x - y\| < \lambda f(p)$, then $\angle xy, n_x = \angle e, n_x \leq \alpha(\lambda) + \beta(\lambda)$ by Lemma 4.3. Then, Lemma 4.6 implies that $\|p - y\| > \lambda f(p)$.

Consider the case in which the support line of e intersects Σ tangentially at a point $x \in e \cap \Sigma$ (Figure 2(c)). If $\|p - x\| < \lambda f(p)$, then $\angle e, n_x < \pi/3$ by Lemma 4.3. This is impossible because the support line of e meets Σ tangentially at x. □

4.3. Violation at Voronoi facets. We first characterize the intersection between Σ and a Voronoi facet F. Let $\mathrm{Int}\,\mathbb{X}$ denote the interior of a topological space $\mathbb{X}$. There are three possible configurations for any connected component I in $\Sigma \cap F$. First, I may contain a tangential contact point between Σ and the plane of F. Second, I may be a smooth closed curve. Third, I may be a smooth open curve (possibly degenerate). In the third possibility, if I is not a single point, $\mathrm{Int}\, I \subseteq \mathrm{Int}\, F$ and the endpoints of I lie on the boundary of F. In this case, we call I a *topological interval.* If I is a single point, Σ barely cuts F at a single vertex v (e.g., Σ cuts across F at its topmost vertex). Then I is just v, and we call I a *degenerate topological interval.*

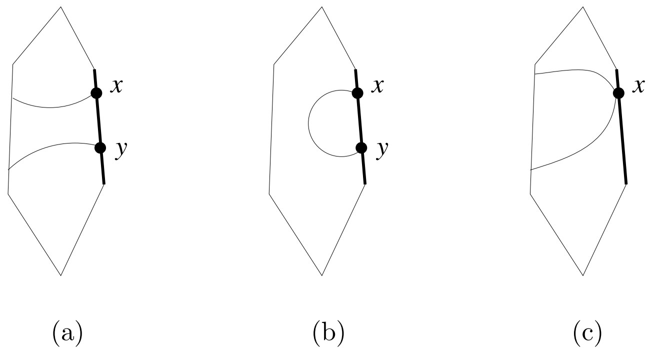


FIG. 2. *The curves are subsets of* Σ. *A Voronoi edge (shown in bold) intersects* Σ *at two points or tangentially.*

The GIP and TBP require that F intersects Σ transversally, if at all, and in at most one topological interval. Recall that transversal intersection permits F (but not its *affine plane*) to meet Σ tangentially at a Voronoi vertex. Also we let GIP and TBP be violated for Voronoi vertices, allowing them to be on Σ. This means we would allow a facet F meeting Σ only at a single topological interval degenerate or not. In all other cases we show that a new point can be sampled far away from existing sample points. First, we show that any tangential contact point between Σ and a Voronoi facet is far away from existing sample points.

LEMMA 4.8. *Let* F *be a facet of a Voronoi cell* V_p. *If the plane of* F *meets* Σ *tangentially at a point* $x \in F \cap \Sigma$, *then* $\|p - x\| \geq \lambda f(p)$.

Proof. Since x is a tangential contact point, the angle between F and n_x is equal to $\pi/2$. Then the contrapositive of Lemma 4.4(ii) implies that $\|p-x\| \geq \lambda f(p)$. ☐

Because of Lemma 4.8, in the rest of this section, we focus on the case where a Voronoi facet F meets Σ transversally, i.e., $F \cap \Sigma$ is a collection of closed curves and/or (possibly degenerate) open curves.

Let L be a line in the plane of F which is normal to a curve in $F \cap \Sigma$. In the next lemma we establish that if L intersects $F \cap \Sigma$ at two or more points, we can find a point in $F \cap \Sigma$ that is far away from existing sample points. Notice that this result holds irrespective of whether $F \cap \Sigma$ is a closed curve or not. Although our main motivation is to get rid of closed curves in $F \cap \Sigma$, it is algorithmically easier to check whether L intersects $F \cap \Sigma$ in at least two points than it is to check whether $F \cap \Sigma$ contains a closed curve.

LEMMA 4.9. *Let* F *be a facet of a Voronoi cell* V_p *where* F *intersects* Σ *transversally. Let* x *be a point on a curve* C *(possibly closed) in* $F \cap \Sigma$. *Let* L *be the line in the plane of* F *that is normal to* C *at* x. *If* L *intersects* $F \cap \Sigma$ *at a point other than* x, *the distance from* p *to the furthest point in* $L \cap F \cap \Sigma$ *is at least* $\lambda f(p)$.

Proof. Refer to Figure 3. By assumption, there is a point y other than x in $L \cap F \cap \Sigma$. If $\|p - x\| \geq \lambda f(p)$, then we are done. If $\|p - x\| < \lambda f(p)$, then $\angle xy, n_x = \angle L, n_x < \alpha(\lambda) + \beta(\lambda)$ by Lemma 4.5. Then, Lemma 4.6 implies that $\|p - y\| > \lambda f(p)$. ☐

The next lemma considers the scenario of F intersecting Σ in two topological intervals, assuming that each edge of F intersects Σ in at most one point. We show that at least one endpoint of the two topological intervals is far away from existing sample points.

LEMMA 4.10. *Let* F *be a facet of a Voronoi cell* V_p *where* F *intersects* Σ *transver-*

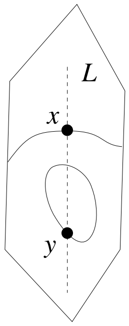


FIG. 3. *The line L is normal to the curve at x, and L intersects $F \cap \Sigma$ at another point y.*

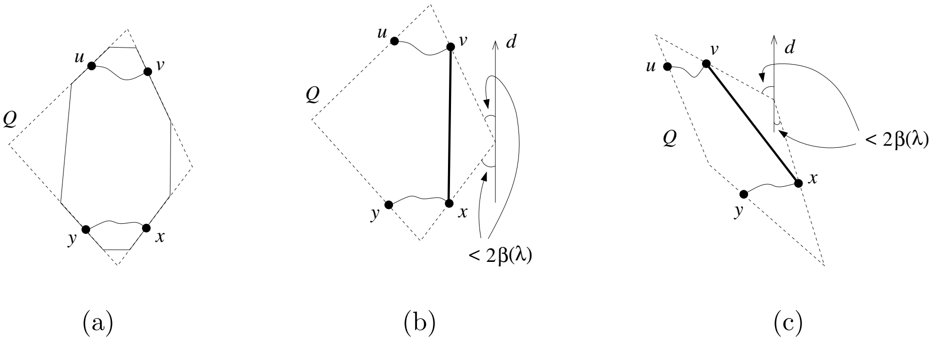


FIG. 4. *A Voronoi facet (bounded by solid line segments) intersects Σ in two topological intervals (shown as curves). The convex quadrilateral Q is shown in dashed line segments. In* (b) *and* (c)*, the direction d is the projection of n_p onto the plane of F.*

sally. Assume that each edge of F intersects Σ in at most one point. If $F \cap \Sigma$ contains two topological intervals I and I', the distance between p and the furthest endpoint of I and I' is at least $\lambda f(p)$.

Proof. Let u and v be the endpoints of I. Let x and y be the endpoints of I'. Since no edge of F intersects Σ in two or more points, the four edges of F containing u, v, x, and y are distinct. Let Q be the convex quadrilateral on the plane of F bounded by the support lines of these four edges. We denote the edges of Q by e_u, e_v, e_x, and e_y according to the interval endpoints that the edges contain. (In the degenerate case in which $I = u = v$, the two edges of F incident to u give rise to two edges of Q. We arbitrarily call one of them e_u and the other e_v. The degenerate case in which $I' = x = y$ is handled similarly.) Refer to Figure 4(a).

Assume to the contrary that the distances $\|p-u\|$, $\|p-v\|$, $\|p-x\|$, and $\|p-y\|$ are less than $\lambda f(p)$. Consider the Delaunay triangles dual to the edges of F containing u, v, x, and y. Their circumradii are less than $\lambda f(p)$. By Lemma 2.3, the angles $\angle e_u, n_p$, $\angle e_v, n_p$, $\angle e_x, n_p$, and $\angle e_y, n_p$ are at most $\beta(\lambda)$.

By Lemma 4.4(i), the acute angle between the plane of F and n_p is less than $\beta(\lambda)$.

Let d be the projection of n_p onto the plane of F. So $\angle n_p, d < \beta(\lambda)$. It follows that $\angle e_u, d \leq \angle e_u, n_p + \angle n_p, d < 2\beta(\lambda)$. Similarly, the angles $\angle e_v, d$, $\angle e_x, d$, and $\angle e_y, d$ are less than $2\beta(\lambda)$. Then the convexity of Q implies that one of its interior angles must be greater than $\pi - 4\beta(\lambda)$, say the interior angle between e_v and e_x.

In this case, a line parallel to d may either cut through or be tangent to the corner of Q between e_v and e_x. Figures 4(b) and 4(c) show the two possibilities. In both configurations, $\angle vx, e_v < 2\beta(\lambda)$ or $\angle vx, e_x < 2\beta(\lambda)$. Assume that $\angle vx, e_x < 2\beta(\lambda)$.

By Lemma 4.3, $\angle e_x, n_x \leq \alpha(\lambda) + \beta(\lambda)$. We conclude that $\angle vx, n_x \leq \angle vx, e_x + \angle e_x, n_x < \alpha(\lambda) + 3\beta(\lambda)$. On the other hand, $\|v - x\| \leq 2\lambda f(x)/(1-\lambda)$ by Lemma 4.2. Then Lemma 2.4 implies that $\angle vx, n_x \geq \arccos(\lambda/(1-\lambda))$. Combining this with the previous upper bound on $\angle vx, n_x$ yields $\arccos(\lambda/(1-\lambda)) < \alpha(\lambda) + 3\beta(\lambda)$ and hence $\lambda/(1-\lambda) > \cos(\alpha(\lambda) + 3\beta(\lambda))$. But this contradicts the fact that $\lambda \leq \lambda_0$. □

4.4. Violation at Voronoi cells. The TBP requires that $V_p \cap \Sigma$ is a topological disk. There are several possibilities when this condition is violated. We assume that Σ meets the edges or facets of V_p transversally because tangential contacts are already handled by Lemmas 4.7 and 4.8.

- $V_p \cap \Sigma$ has more than one boundary cycle.
- Some connected component of $V_p \cap \Sigma$ is a surface without boundary.
- $V_p \cap \Sigma$ has nonzero genus.

Notice that $V_p \cap \Sigma$ is orientable because Σ is orientable. We deal with the first possibility in section 4.4.1. The second possibility is eliminated because the initialization in our algorithm inserts all critical points of Σ in the x_3-direction. Any component of Σ would contain two such critical points which would violate the emptiness of V_p. It turns out that the last possibility is hard to detect. In section 4.4.2, we propose to enforce a stronger condition based on the notion of *silhouette*. Detecting the violation of this stronger condition is easier, and a new point can be sampled readily in case of a violation.

Since we allow Voronoi vertices to lie on Σ, we give a definition of boundary cycles of $V_p \cap \Sigma$ that capture degenerate cases as well. Let $\operatorname{Bd} \mathbb{X}$ denote the boundary of a topological space $\mathbb{X}$. Consider a connected component C in $\Sigma \cap \operatorname{Bd} V_p$. The nontangential contact assumption implies that C is either a nondegenerate closed curve or a vertex of V_p. If C is a nondegenerate closed curve, then C is clearly a boundary cycle of $V_p \cap \Sigma$. The case of C being a vertex v of V_p happens when Σ barely cuts V_p at v. We consider v as a *degenerate boundary cycle* of $V_p \cap \Sigma$.

4.4.1. Two or more boundary cycles. We first prove a technical result as stated in Lemma 4.11. This lemma says that the distance from p to any point in a topological interval is dominated by its distances to the interval endpoints. (Recall that a topological interval is allowed to degenerate to a vertex of V_p.)

LEMMA 4.11. *Let F be a facet of a Voronoi cell V_p. Suppose that Σ meets edges of F transversally and $F \cap \Sigma$ contains a topological interval I. If the distances from p to the endpoints of I are less than $\lambda f(p)$, the distance from p to any point in I is less than $\lambda f(p)$.*

Proof. The lemma is trivially true if I degenerates to a vertex of V_p. So we can assume that I has two distinct endpoints.

Let B be the ball centered at p with radius $\lambda f(p)$. Let Π denote the plane of F. Since the distances from p to the endpoints of I are less than $\lambda f(p)$, $B \cap \Pi$ is a disk D, and the endpoints of I lie in $\operatorname{Int} D$. To prove the lemma, it suffices to show that $I \subseteq \operatorname{Int} D$. Assume to the contrary that $I \nsubseteq \operatorname{Int} D$.

Let C be the connected component of $\Pi \cap \Sigma$ containing I. Notice that $C \nsubseteq \operatorname{Int} D$ because $I \subset C$ and $I \nsubseteq \operatorname{Int} D$ by assumption. For any point $x \in \Pi \cap \Sigma \cap D$, since $\|p - x\| \leq \lambda f(p)$, x cannot be a tangential contact point between Π and Σ by Lemma 4.4(ii). Thus, $C \cap \operatorname{Int} D$ is a collection of disjoint simple curves (open or closed).

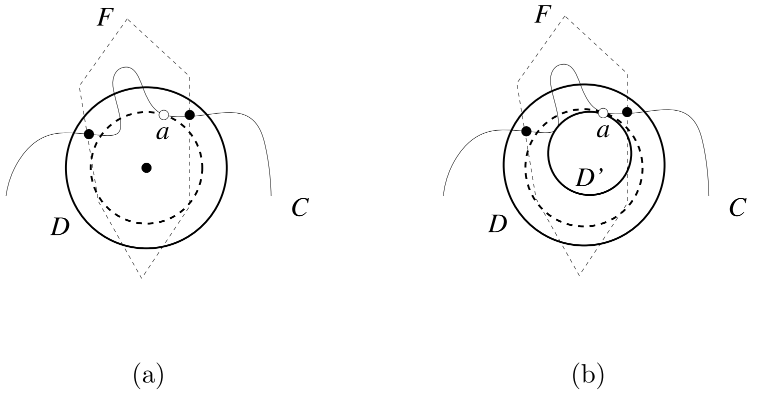


FIG. 5. *Shrinking D to D':* (a) *shows the radial shrinking, and* (b) *shows the shrinking by moving the center.*

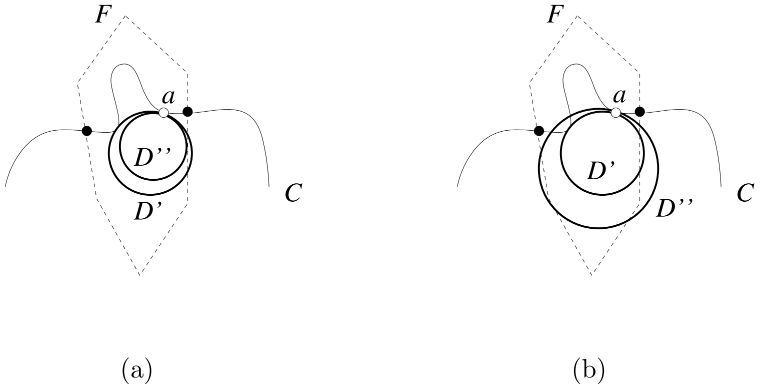


FIG. 6. *D' and D''.*

We claim that $C \cap \operatorname{Int} D$ is not connected. Assume to the contrary that $C \cap \operatorname{Int} D$ is connected, i.e., $C \cap \operatorname{Int} D$ is a single curve. Recall that $I \subset C$, $I \not\subseteq \operatorname{Int} D$, and the endpoints of I lie in $C \cap \operatorname{Int} D$. It follows that $(C \cap \operatorname{Int} D) \cup I$ is a closed curve. Take an edge e of F that contains an endpoint x of I. Since Σ meets e transversally by assumption, the support line ℓ of e crosses C. Because ℓ intersects I only at x, ℓ must intersect $C \cap \operatorname{Int} D$ at x and at least one other point y. Since $x, y \in D$, both $\|p - x\|$ and $\|p - y\|$ are at most $\lambda f(p)$. By Lemma 4.3, $\angle xy, n_x = \angle e, n_x \leq \alpha(\lambda) + \beta(\lambda)$. But then Lemma 4.6 implies that $\|p - x\|$ or $\|p - y\|$ is greater than $\lambda f(p)$, a contradiction.

So we can assume that $C \cap \operatorname{Int} D$ consists of at least two disjoint curves. Then, D can be shrunk to a smaller disk D' as follows so that D' meets C tangentially at two points and $C \cap \operatorname{Int} D' = \emptyset$. First, shrink D radially until it touches C at some point a. Refer to Figure 5(a). It follows that $\|p - a\| < \lambda f(p)$. If this shrunken D does not meet the requirement of D' yet, we shrink it further by moving its center towards a until we obtain the disk D' as required. Refer to Figure 5(b). Notice that a is one of the contact points between D' and C.

The plane Π intersects the two medial balls of Σ at a in two disks. Among these two disks, let D'' be the one that intersects D'. Let B'' be the medial ball so that $D'' = B'' \cap \Pi$. The boundary circles of D' and D'' meet tangentially at a. So either $D'' \subseteq D'$ (Figure 6(a)) or $D' \subset D''$ (Figure 6(b)).

We claim that $D'' \subseteq D'$ and radius$(D'') > \lambda f(p)$. Suppose that $D' \subset D''$. By construction, D' meets Σ tangentially at two points. So one of these contact points must lie in $\operatorname{Int} D''$. This is a contradiction because $D'' = B'' \cap \Pi$ and $\operatorname{Int} B'' \cap \Sigma = \emptyset$ as B'' is a medial ball. This shows that $D'' \subseteq D'$. By Lemma 4.4(ii), the acute angle between Π and n_a is less than $\alpha(\lambda) + \beta(\lambda)$. Observe that the angle between the diametric segments of B'' and D'' incident to a is equal to the angle between n_a and Π. Therefore, radius$(D'') >$ radius$(B'') \cdot \cos(\alpha(\lambda) + \beta(\lambda)) \geq f(a) \cdot \cos(\alpha(\lambda) + \beta(\lambda)) > \lambda f(a)/(1 - \lambda)$ as $\lambda \leq \lambda_0$. It follows from Lemma 4.1 that radius$(D'') > \lambda f(p)$. This completes the proof of our claim.

By our claim, radius$(D') \geq$ radius$(D'') > \lambda f(p)$. But D' is obtained by shrinking $D = B \cap \Pi$ and radius$(B) = \lambda f(p)$, a contradiction. In all, the contrapositive assumption that $I \not\subseteq \operatorname{Int} D$ cannot hold. It follows that the distance from p to any point in I is less than $\lambda f(p)$. □

Lemma 4.11 is used in proving Lemma 4.12, which says that if $V_p \cap \Sigma$ has more than one boundary cycle, some edge of V_p intersects Σ in a point far away from existing sample points. It is convenient to distinguish between different types of cycles in $V_p \cap \Sigma$. A boundary cycle is of *type* 1 if it is degenerate or a concatenation of topological intervals in the intersections between Σ and the facets of V_p. A boundary cycle is of *type* 2 if it is nondegenerate and contained in a facet of V_p.

LEMMA 4.12. *Let p be a sample point. Assume that the following conditions hold.*

- *Σ meets edges or facets of V_p transversally.*
- *$V_p \cap \Sigma$ contains at least two boundary cycles of type* 1.

Then the distance from p to the furthest intersection point between Σ and the edges of V_p is at least $\lambda f(p)$.

Proof. Assume to the contrary that the distances from p to the intersection points between Σ and the edges of V_p are less than $\lambda f(p)$. This will lead to contradictions thereby proving the lemma. We first prove a technical result that will be used later.

CLAIM 1. *Let $x \in \Sigma$ be a vertex of V_p such that $\|p - x\| \leq \lambda f(p)$. Let E be a set of edges of V_p incident to x that point towards the same side of Σ. The smallest cone enclosing E with apex x and axis n_x cannot contain a point $y \in \Sigma$ other than x such that $\|p - y\| \leq \lambda f(p)$.*

Proof. By Lemma 2.3, each edge in E makes an angle at most $\beta(\lambda)$ with n_p. By Lemma 2.2, $\angle n_x, n_p \leq \alpha(\lambda)$. Therefore, each edge in E makes an angle at most $\alpha(\lambda) + \beta(\lambda)$ with n_x. So the aperture of the smallest cone enclosing E with apex x and axis n_x is at most $2\alpha(\lambda) + 2\beta(\lambda)$. Let $y \in \Sigma$ be any point other than x in this cone. Thus, $\angle n_x, xy \leq \alpha(\lambda) + \beta(\lambda) < \pi/3$ as $\lambda \leq \lambda_0$. Then Lemmas 2.4 and 4.1 imply that $\|x - y\| \geq 2f(x)\cos(\angle n_x, xy) \geq f(x) \geq (1 - \lambda)f(p)$. Since $\|p - y\| \geq \|x - y\| - \|p - x\| \geq (1 - 2\lambda)f(p)$, $\|p - y\| \geq \lambda f(p)$ as $\lambda \leq \lambda_0 = 0.06$. □

Lemma 4.11 implies that the boundary cycles of type 1 lie strictly inside a closed ball B centered at p with radius $\lambda f(p)$. By a result of Boissonnat and Cazals [5], any closed ball centered at p with radius less than $f(p)$ intersects Σ in a topological disk. Thus $B \cap \Sigma$ is a topological disk. It follows that each nondegenerate boundary cycle of type 1 bounds exactly one topological disk in $B \cap \Sigma$ (strictly inside B). Since the boundary cycles are disjoint, the topological disks bounded by them are either disjoint or nested. This means there exists one such topological disk that does not contain any cycle of type 1. The next claim proves some properties of such a topological disk.

CLAIM 2. *Let C be a boundary cycle of type* 1 *bounding a topological disk D which does not contain any other boundary cycle of type* 1. *Then* (i) *C is a nondegenerate boundary cycle*, (ii.a) *D does not contain any other boundary cycles*, *and* (ii.b) *D lies in V_p.*

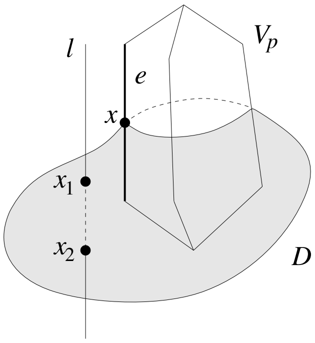


FIG. 7. *Proof of Claim* 2: *disk D with the opening C is like a "sack" that contains a part of V_p.*

Proof. Consider (i). If not, C is an isolated vertex v of V_p in the intersection $V_p \cap \Sigma$. The edges of V_p incident to v must point towards the same side of Σ. Indeed, otherwise, Σ intersects the interior of V_p in a small neighborhood of v, contradicting the assumption that v is an isolated point in the intersection $V_p \cap \Sigma$. On the other hand, Claim 1 is contradicted because $\|p - v\| < \lambda f(p)$ and p lies inside the smallest cone with apex v and axis n_v that encloses the edges of V_p incident to v. This proves (i).

Consider (ii.a). By the definition of C, D does not contain other boundary cycles of type 1. Assume to the contrary that D contains a boundary cycle C' of type 2. So C' is contained in a facet of V_p. Since D lies strictly inside B, C' lies strictly inside B. But by applying Lemma 4.9 to C', the distance from p to some point in C' is at least $\lambda f(p) = \text{radius}(B)$, a contradiction.

Consider (ii.b). It is sufficient to show that $\text{Int}\, D$ lies in V_p. Suppose not. Then $\text{Int}\, D$ lies completely outside V_p. Otherwise, $\text{Int}\, D$ would contain a boundary cycle which is prohibited by (ii.a). Refer to Figure 7. Take an edge e of V_p that intersects C. Let x be an intersection point between e and C. By Lemma 4.3, $\angle e, n_x < \pi/3$. Thus, the support line of e intersects Σ transversally at x. Let ℓ be a line outside V_p that is parallel to and arbitrarily close to the support line of e. Then ℓ must intersect $\text{Int}\, D$ transversally at a point x_1 arbitrarily close to x.

Since $\text{Bd}\, V_p$ is a topological sphere, it is divided by C into two topological disks. Let T denote one of them. Then $T \cup D$ is a topological sphere. Since ℓ intersects $\text{Int}\, D$ at x_1, ℓ must intersect $T \cup D$ at another point $x_2 \neq x_1$.

The point x_2 must lie on D because $T \subseteq \text{Bd}\, V_p$ and ℓ lies outside V_p. By Lemma 2.4, $\|x_2 - x_1\| \geq 2f(x_1)\cos(\angle \ell, n_{x_1})$. Note that x_1 is arbitrarily close to x and $\angle \ell, n_x = \angle e, n_x < \pi/3$. Thus, $\angle \ell, n_{x_1} < \pi/3$ and so $\|x_2 - x_1\| \geq f(x) \geq (1-\lambda)f(p)$ by Lemma 4.1. But this is a contradiction because D lies inside B whose diameter is $2\lambda f(p) < (1-\lambda)f(p)$. ◻

Let C be a boundary cycle as stated in Claim 2. Let D be the topological disk in $B \cap \Sigma$ bounded by C. Let $\mathcal{F}$ be the set of facets of V_p that intersect C. Each facet in $\mathcal{F}$ bounds a half-space containing p. The intersection of these half-spaces is a convex polytope P containing V_p. Since D lies in V_p by Claim 2, D lies in P, too.

Let C' be another boundary cycle of type 1 which must exist by the assumption of the lemma. Recall that both C and C' lie inside $B \cap \Sigma$ by Lemma 4.11. Let ρ be a curve in $(B \cap \Sigma) \setminus \text{Int}\, V_p$ such that ρ connects C with C'. Since $D \subset P$ and the contact between D and $\text{Bd}\, P$ is nontangential, ρ leaves P when ρ leaves D. Since

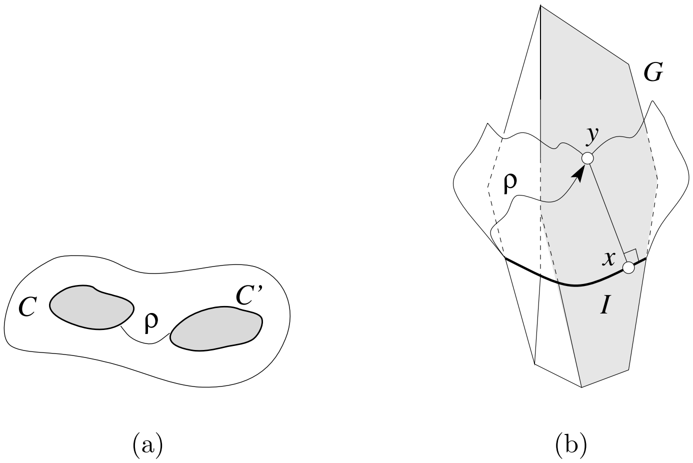


FIG. 8. (a) *Two cycles C and C' drawn schematically on the patch $B \cap \Sigma$. The path ρ starting from C goes outside V_p and then has to reach V_p again to reach C'.* (b) *A different view with the polyhedron P. The lower bold curve denotes C, and its intersection with the shaded facet G is a topological interval I. The curved patch shown is part of $(B \cap \Sigma) \setminus \operatorname{Int} V_p$. The curved path on the curved patch is ρ.*

$C' \subset V_p \subseteq P$, ρ must return to some facet of P in order to meet C' eventually. Let G be a facet of P that ρ intersects after leaving D. Let y be a point in $\rho \cap G$. Let F be the facet of V_p contained in G. By the definition of P, C must intersect F.

If $C \cap F$ consists of two or more topological intervals, Lemma 4.10 is applicable, and we are done. So we assume in the rest of the proof that $C \cap F$ is a single topological interval I. Refer to Figure 8.

CLAIM 3. *Each edge of F that contains an endpoint of I is contained in some edge of G.*

Proof. Consider an endpoint z of I. The point z lies on the boundary of F, which means that the other facet(s) of V_p that share z with F are intersected by C. So the planes of these facets also bound P. It follows that the edges of F containing z are contained in some edges of G. □

It follows from Claim 3 that the endpoints of I lie on the boundary of G. Let x be the closest point to y on I. Notice that $x \in I \subset C \subset B \cap \Sigma$ and $y \in \rho \subset B \cap \Sigma$. So the distances from p to x and y are at most radius$(B) = \lambda f(p)$. Let L be the line passing through x and y. There are three cases to consider.

- *Case* 1. I is a degenerate interval. So $I = x$ is a vertex of F. By Claim 3, the two edges of F incident to x are contained in edges of G. So x is also a vertex of G. Let e_1 and e_2 be the two edges of G incident to x (which contain the edges of F incident to x). Since I is a degenerate interval, e_1 and e_2 must point towards the same side of Σ. But then Claim 1 is contradicted because y lies inside the smallest cone with apex x and axis n_x that encloses e_1 and e_2.
- *Case* 2. I is nondegenerate, and x lies in the interior of G. Then L intersects I at x at a right angle. By Lemma 4.5, $\angle L, n_x < \alpha(\lambda) + \beta(\lambda)$. But then $\|p - x\|$ or $\|p - y\|$ is greater than $\lambda f(p)$ by Lemma 4.6, a contradiction.
- *Case* 3. I is nondegenerate, and x lies on the boundary of G. Let e be an edge of G containing x. By Claim 3, e contains an edge of F which contains x. Let d be the projection of n_x onto the plane of G. Refer to Figure 9(a).

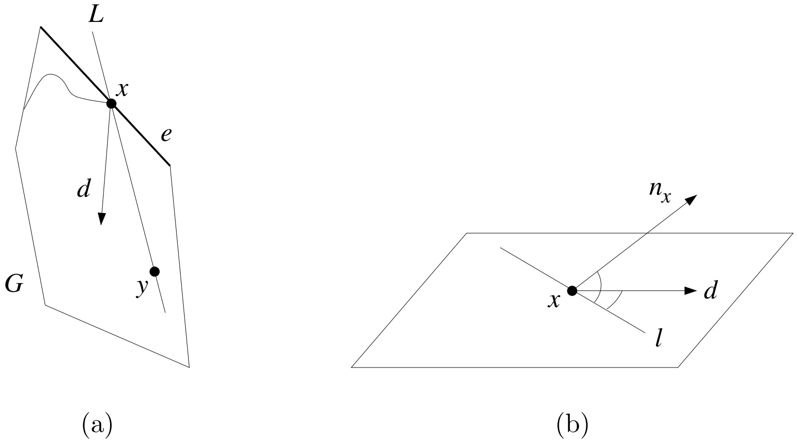


FIG. 9. *In* (a)*, the curve denotes $C \cap F$, and $\angle L, d \leq \angle e, d$. In* (b)*, the angle $\angle \ell, n_x$ is an increasing function of $\angle \ell, d$.*

Since x is the closest point on I to y, $\angle L, d \leq \angle e, d$. Next, observe that for any line ℓ in the plane of G, the angle $\angle \ell, n_x$ increases as the angle $\angle \ell, d$ increases. Refer to Figure 9(b). We conclude that $\angle L, n_x \leq \angle e, n_x$, which is at most $\alpha(\lambda) + \beta(\lambda)$ by Lemma 4.3. But then $\|p - x\|$ or $\|p - y\|$ is greater than $\lambda f(p)$ by Lemma 4.6, a contradiction. □

4.4.2. Silhouette. Consider the possibility of some connected component of $V_p \cap \Sigma$ being a closed surface. This closed surface must contain two critical points of Σ in the x_3-direction. Therefore, this possibility is eliminated by our algorithm because we start by including all critical points of Σ in the x_3-direction as sample points.

The remaining possible violation of TBP is that $V_p \cap \Sigma$ is connected, has exactly one boundary cycle, and has positive genus. Intuitively, $V_p \cap \Sigma$ contains a handle, and Morse theory says that there should be a critical point of Σ in the x_3-direction. However, it is not guaranteed that this critical point lies inside V_p. (If it did, the initialization in our algorithm would eliminate this possibility.)

Figure 10 illustrates this possibility. A closed dashed curve cuts a topological disk from the torus. The parts of the topological disk in front are shown shaded, and the rest of the topological disk is at the back. All the critical points of the torus lie on

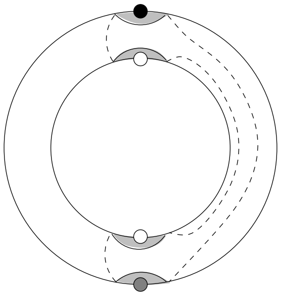


FIG. 10. *The closed dashed curve cuts a topological disk from the torus which contains all four critical points of the torus: a maximum (black), a minimum (gray), and two saddles (white).*

the topological disk. Suppose that the facets of a Voronoi cell V_p intersect a torus at this closed dashed curve such that the bounded topological disk lies outside V_p and the rest of the torus lies inside V_p. Then V_p violates TBP although the surface clipped within V_p is connected, has a single boundary cycle, and has no critical point. Apparently it may seem that a convex polytope cannot intersect the standard torus in a closed curve as shown in the figure. However, one may imagine deforming the torus to admit a closed curve with the stated property.

We propose a stronger condition to ensure that $V_p \cap \Sigma$ is a topological disk. The violation of this stronger condition can be detected, and a new point can be sampled readily.

Lemma 4.13 below states that if a connected component M of $V_p \cap \Sigma$ is connected, has exactly one boundary cycle, and avoids J_d, then M is a topological disk.

LEMMA 4.13. *Let M be a connected component of $V_p \cap \Sigma$ such that the boundary of M is a single boundary cycle. If the boundary cycle of M is degenerate or there is a direction d such that $M \cap J_d = \emptyset$, then M is a topological disk.*

Proof. If the boundary cycle of M is degenerate, M is a vertex of V_p and thus M is a degenerate topological disk. Otherwise, M is a connected compact 2-manifold and its boundary is a simple closed curve by the assumption of the lemma. Let d be a direction satisfying the condition of the lemma. Let H be a plane perpendicular to d. Consider the map $\varphi : M \to H$ that projects each point of M orthogonally to H. Since M is connected and compact and M has a single boundary cycle, it suffices to prove that φ is injective.

Assume to the contrary that φ is not injective. Then there is a line L parallel to d that intersects M in two or more points. Let x and y be two consecutive intersection points along L. Let M' be the connected component of Σ containing M. By the convexity of V_p, x and y are consecutive intersection points in $L \cap M'$, too.

Since $M \cap J_d = \emptyset$, neither x nor y belongs to J_d, which means that neither n_x nor n_y is orthogonal to d. Because x and y are consecutive in $L \cap M'$, n_x and n_y are oppositely oriented in the sense that the inner products $\langle n_x, d\rangle$ and $\langle n_y, d\rangle$ have opposite signs. Since M is connected, there is a smooth curve ρ in M connecting x and y. The normal to Σ changes smoothly from n_x to n_y along ρ. By the mean-value theorem, there is a point $z \in \rho$ such that n_z is orthogonal to d. But then $z \in M \cap J_d$, contradicting the emptiness of $M \cap J_d$. □

By Lemma 4.13, when we are left with the case that $V_p \cap \Sigma$ is connected and has a single boundary cycle, it suffices to check whether $V_p \cap \Sigma$ intersects J_d where $d = n_p$. If not, $V_p \cap \Sigma$ is a topological disk. Otherwise, the following result says that any point in $V_p \cap J_d$ can be inserted as a new sample point.

LEMMA 4.14. *Let p be a sample point. Let $d = n_p$. If x is a point in $V_p \cap J_d$, $\|p - x\| \geq \lambda f(p)$.*

Proof. If $\|p - x\| < \lambda f(p)$, Lemma 2.2 would imply that $\angle n_p, n_x \leq \alpha(\lambda)$. But this cannot be the case because n_x is orthogonal to n_p by definition. □

5. Topology recovery. In this section, we present an algorithm to sample a point set S from Σ. We discuss how to handle Voronoi vertices lying on Σ in section 5.1, which is the remaining violation of GIP that we did not address in the previous section. When some Voronoi vertices lie on Σ, $\operatorname{Del} S|_\Sigma$ contains their dual Delaunay tetrahedron. The handling of this case boils down to extracting a triangulated surface $\operatorname{Tri} S|_\Sigma$ from $\operatorname{Del} S|_\Sigma$ to approximate Σ. The definition of $\operatorname{Tri} S|_\Sigma$ is given in section 5.1. In section 5.2, we present some numerical primitives needed by our algorithm. Section 5.3 describes several subroutines. These subroutines sample points from Σ based on the results in section 4. In section 5.4, we put these subroutines together to form the topology recovering algorithm. Section 5.5 describes the analysis of the algorithm.

5.1. Handling Voronoi vertices on the surface. Our idea is to conceptually perturb Σ to obtain another surface Σ' so that no Voronoi vertex lies on Σ'. The perturbation is kept small so that Σ remains homeomorphic to Σ'.

We elaborate on the conceptual perturbation. Let S be the current set of sample points on Σ. Let U be the subset of Voronoi vertices of $\operatorname{Vor} S$ that lie on Σ. Let $B_{v,\delta}$ denote the ball centered at $v \in U$ with radius δ. Refer to Figure 11(a). First, choose δ small enough so that the following conditions hold for each $v \in U$:

C1: $\Sigma \cap B_{v,\delta}$ is a topological disk.

C2: $B_{v,\delta}$ does not intersect $B_{w,\delta}$ for any other Voronoi vertex $w \in U$.

C3: $B_{v,\delta}$ intersects only the Voronoi edges and facets incident to v.

C4: Within $B_{v,\delta}$, the Voronoi edges incident to v intersect Σ only at v.

Condition C1 guarantees that for each $v \in U$, $\Sigma \cap B_{v,\delta}$ separates $B_{v,\delta}$ into two regions, one on each side of Σ. Let R_v denote the region inside Σ. Let D_v denote the portion of the boundary of $B_{v,\delta}$ inside R_v. Notice that D_v is a topological disk.

We obtain a new surface by replacing $\Sigma \cap B_{v,\delta}$ with D_v for every Voronoi vertex $v \in U$. Refer to Figure 11(b). The new surface needs to be smoothed at the sharp boundary of D_v. The smoothing can be restricted in an arbitrarily small neighborhood of the boundary of D_v by introducing arbitrarily high curvature. Hence, it has no effect on the conceptual perturbation. Let Σ' denote the resulting surface.

Condition C2 guarantees that the above disk replacements can be performed simultaneously for all vertices in U. Condition C3 guarantees that only Voronoi edges, facets, and cells incident to v are affected by the perturbation.

The idea is to use $\operatorname{Del} S|_{\Sigma'}$ instead of $\operatorname{Del} S|_{\Sigma}$ as the triangulation approximating Σ. We do not actually perturb Σ to obtain a new surface Σ'. Instead, we perform a simulation. This allows us to exclude some triangles from $\operatorname{Del} S|_{\Sigma}$ to obtain $\operatorname{Del} S|_{\Sigma'}$. The details of the simulation are as follows.

Let uv be a Voronoi edge incident to v. Since our topology recovery algorithm will first check for tangential contacts between Σ and the Voronoi edges, we can assume that uv is not orthogonal to n_v. Condition C4 means that uv does *not* intersect Σ' if $\langle u - v, n_v \rangle > 0$. (Recall that n_v is the unit outward normal at v.) In this case, if uv does not intersect Σ in a point other than v, we exclude the dual Delaunay triangle

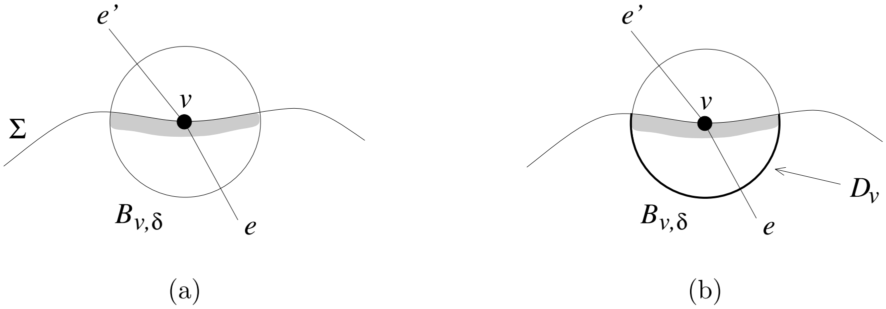


FIG. 11. *The solid curve denotes Σ. The shaded area denotes the inside of Σ. The bold curve on the right denotes the topological disk D_v. The Voronoi edge e' is treated as not intersecting Σ at v, while the Voronoi edge e is treated as intersecting Σ at v.*

of uv. On the other hand, if uv intersects Σ in a point other than v, we keep its dual Delaunay triangle.

Repeating the above for every $v \in U$ yields $\mathrm{Del}\, S|_{\Sigma'}$. Since we compute with Σ instead of Σ', it is more concrete to have a notation for $\mathrm{Del}\, S|_{\Sigma'}$ using Σ. We use $\mathrm{Tri}\, S|_{\Sigma}$ to denote $\mathrm{Del}\, S|_{\Sigma'}$. The surface Σ' cannot introduce new tangential intersections. In the rest of this section, we argue that if $\mathrm{Vor}\, S$ violates TBP with Σ', it violates TBP with Σ, too. Therefore, when Σ intersects the edges and facets of $\mathrm{Vor}\, S$ transversally and $\mathrm{Vor}\, S$ satisfies TBP with Σ, $\mathrm{Vor}\, S$ satisfies GIP and TBP with Σ'. Applying Theorem 2.1, we obtain that $\mathrm{Tri}\, S|_{\Sigma} = \mathrm{Del}\, S|_{\Sigma'}$ is homeomorphic to Σ.

Take a Voronoi edge e incident to $v \in U$. Let v' be the point near v in which Σ' intersects e after perturbation. Suppose that e violates TBP with respect to Σ'. The violation means that e intersects Σ' in v' and at least one other point. It follows that e intersects Σ in v and at least one other point. So e violates TBP with respect to Σ too.

Take a Voronoi facet F incident to $v \in U$. If Σ cuts through F at v, the topology of $F \cap \Sigma$ does not change after perturbation at v. The interesting case is that Σ' intersects both edges of F incident to v at points near v after perturbation. In this case, $F \cap \Sigma'$ is not just a geometric perturbation of $F \cap \Sigma$ because $F \cap \Sigma'$ and $F \cap \Sigma$ have different topologies. Specifically, $F \cap \Sigma'$ contains a short topological interval I' near v. If I' participates in a violation of TBP with respect to Σ', $F \cap \Sigma'$ must contain a connected component other than I'. Then, F intersects Σ in at least two components, one of them being the degenerate topological interval v. So F violates TBP with respect to Σ too.

Similarly, consider a Voronoi cell V_p incident to $v \in U$. The interesting case is that Σ' intersects all edges of V_p incident to v at points near v after perturbation. So $V_p \cap \Sigma'$ contains a small topological disk D' near v. If D' participates in a violation of TBP with respect to Σ', $V_p \cap \Sigma'$ must have a connected component other than D'. It follows that $V_p \cap \Sigma$ has at least two connected components, one of them being the degenerate boundary cycle v. So V_p violates TBP with respect to Σ too.

5.2. Numerical primitives. We introduce six numerical primitives. They are responsible for computing the intersections between Σ and lines and computing the critical points of certain height functions on Σ. We assume a numerical or symbolic solver for solving a system of equations (e.g., [27]).

- SurfaceCritical(surface Σ): Let d denote the x_3-direction. This primitive returns the critical points of Σ in the x_3-direction by returning the solutions of the system of equations $E(x) = 0$, $\mathrm{grad}\, E(x) \times d = 0$.
- SurfaceLine(surface Σ, line ℓ): This primitive returns the intersection points between Σ and the line ℓ. We assume that ℓ is given by its direction and a point on it. Using this information, we can compute two planes $H_1(x) = 0$ and $H_2(x) = 0$ whose intersection is equal to ℓ. The intersection points are the solutions of the system of equations $E(x) = 0$, $H_1(x) = 0$, $H_2(x) = 0$.
- SurfacePlaneContact(surface Σ, plane Π): This primitive returns the tangential contact points between Σ and the plane Π. Let $H(x)$ be the equation of Π. Let d be the normal to Π. The contact points are the solutions of the system of equations $E(x) = 0$, $H(x) = 0$, $\mathrm{grad}\, E(x) \times d = 0$.
- CurveCritical(surface Σ, plane Π, vector d): We assume that d is parallel to Π. This primitive returns the critical points of $\Sigma \cap \Pi$ in direction d and the tangential contact points in $\Sigma \cap \Pi$. Let d' be a normal of Π. The tangent to a point $x \in \Sigma \cap \Pi$ is parallel to $\mathrm{grad}\, E(x) \times d'$. So x is a critical point in

direction d if and only if $\langle \operatorname{grad} E(x) \times d', d\rangle = 0$. Let $H(x)$ be the equation of Π. The critical points desired satisfy the system of equations $E(x) = 0$, $H(x) = 0$, $\langle \operatorname{grad} E(x) \times d', d\rangle = 0$. The same system captures all tangential contact points $x \in \Sigma \cap \Pi$, too, because $\operatorname{grad} E(x) \times d' = 0$ in this case.

- SilhouettePlane(surface Σ, plane Π, vector d): Let J_d be the silhouette of Σ in direction d. Let $H(x)$ be the equation of Π. This primitive returns the intersection points between J_d and Π, which are the solutions of the system of equations $E(x) = 0$, $H(x) = 0$, $\langle \operatorname{grad} E(x), d\rangle = 0$.
- SilhouetteCritical(surface Σ, vector d, vector d'): Let J_d be the silhouette of Σ in direction d. This primitive returns the critical points of J_d in direction d', i.e., points on J_d whose tangents are orthogonal to d'. Let $G(x) = \langle \operatorname{grad} E(x), d\rangle$. The silhouette J_d is the intersection of the isosurface $G(x) = 0$ and the input surface $E(x) = 0$. Therefore, the tangent to a point $x \in J_d$ is parallel to $\operatorname{grad} G(x) \times \operatorname{grad} E(x)$. So x is a critical point if and only if $\langle \operatorname{grad} G(x) \times \operatorname{grad} E(x), d'\rangle = 0$. The critical points desired are the solutions of the system of equations $E(x) = 0$, $G(x) = 0$, $\langle \operatorname{grad} G(x) \times \operatorname{grad} E(x), d'\rangle = 0$.

5.3. Subroutines. We now describe five subroutines, VorEdge, TopoDisk, FacetContact, FacetCycle, and Silhouette, that implement the results in section 4. They check for any violation of GIP and TBP at Voronoi edges and facets. In case of violation, they sample new points on Σ. We use S to denote the set of sample points maintained by our algorithm.

The first subroutine, VorEdge, checks the GIP and TBP for a Voronoi edge. In case of violation, it returns a point as stated in Lemma 4.7.

```
VorEdge(Voronoi edge $e$)
  1. Compute $X :=$ SurfaceLine($\Sigma, \ell$), where $\ell$ is the support line
     of $e$.
  2. Let $V_p$ be a Voronoi cell incident to $e$.
  3. If $\ell$ meets $\Sigma$ tangentially at some point $x$ on $e$, return $x$.
  4. If $|e \cap X| \geq 2$, return the point in $e \cap X$ furthest from $p$.
  5. Return null.
```

The next subroutine, FacetContact, detects any violation GIP at a Voronoi facet (Lemma 4.8).

```
FacetContact(Voronoi facet $F$)
  1. Compute $X :=$ SurfacePlaneContact($\Sigma, \Pi$), where $\Pi$ is the
     plane of $F$.
  2. If some point in $X$ lies on $F$, return it. Otherwise, return null.
```

In the next subroutine, TopoDisk, we need to check whether the triangles in $\mathrm{Tri}\, S|_\Sigma$ incident to a sample point p form a topological disk. It is assumed that VorEdge has been called and returns null for all Voronoi edges. This implies that Σ meets Voronoi edges transversally, which allows the extraction of triangles of $\mathrm{Tri}\, S|_\Sigma$ using the conceptual perturbation.

TopoDisk performs the checking in two steps. Let T_p be the set of triangles in $\mathrm{Tri}\, S|_\Sigma$ incident to p. First, check whether every triangle edge in T_p is incident to exactly two triangles in T_p. Second, check whether T_p forms exactly one cycle of triangles around p. If both tests are passed, T_p forms a topological disk; otherwise, it does not.

In the subsequent proof of correctness, we will see that TopoDisk handles two possible violations of TBP. First, a facet of V_p intersects Σ in two topological intervals (Lemma 4.10). Second, $V_p \cap \Sigma$ contains two or more boundary cycles (Lemma 4.12).

```
TopoDisk(sample point $p$)
  1. If the triangles in $\mathrm{Tri}\, S|_\Sigma$ incident to $p$ form a topological disk,
     return null.
  2. Otherwise, find the intersection point $x$ between $\Sigma$ and the edges
     of $V_p$ that is furthest from $p$. Return $x$.
```

The next subroutine, FacetCycle, guards against the possibility of a Voronoi facet F intersecting Σ in a cycle. It assumes that FacetContact(F) has been called and returns null. This implies that Σ intersects F transversally. It returns a point as stated in Lemma 4.9.

```
FacetCycle(Voronoi facet $F$)
  1. Compute $X := \mathrm{CurveCritical}(\Sigma, \Pi, d)$, where $\Pi$ is the plane
     of $F$ and $d$ is a direction parallel to $\Pi$.
  2. If no point in $X$ lies on $F$, return null.
  3. Since FacetContact($F$) returned null, $F \cap \Sigma$ is a collection of
     disjoint simple curves (open or closed) and $X \cap F$ is the set of
     critical points of these curves in direction $d$. Let $V_p$ be a Voronoi
     cell incident to $F$. For each $x \in X \cap F$, do the following:
     (a) Compute the line $\ell_x$ in $\Pi$ through $x$ parallel to $d$. Notice
         that $\ell_x$ is normal to $F \cap \Sigma$ at $x$.
     (b) Compute $X' := \mathrm{SurfaceLine}(\Sigma, \ell_x)$. If $|X' \cap F| \geq 2$,
         return the point in $X' \cap F$ furthest from $p$.
  4. Return null.
```

The next subroutine, Silhouette, checks if a Voronoi cell V_p intersects the silhouette J_d, where $d = n_p$ (Lemma 4.14). If $V_p \cap J_d \neq \emptyset$, either J_d intersects some facets of V_p or V_p completely contains a component of J_d. The second possibility can be detected by checking if V_p contains any critical point of J_d in a direction orthogonal to d.

```
Silhouette(sample point $p$)
  1. Choose a direction $d'$ orthogonal to $n_p$.
  2. Compute $X := \mathrm{SilhouetteCritical}(\Sigma, n_p, d')$.
  3. If $X$ contains a point inside $V_p$, return it.
  4. Otherwise, for each facet $F$ of $V_p$, do the following:
     (a) Compute $X' := \mathrm{SilhouettePlane}(\Sigma, \Pi, n_p)$, where $\Pi$ is
         the plane of $F$.
     (b) If $X'$ contains a point in $F$, return it.
  5. Return null.
```

5.4. Topology recovery algorithm. Algorithm SampleTopology samples a set of points S on Σ so that $\mathrm{Tri}\, S|_\Sigma$ is homeomorphic to Σ. It begins with initializing S to contain the critical points of Σ in the x_3-direction. We denote it by S_0, the *seeds*. Then the algorithm calls a procedure Topology that repeatedly invokes the subroutines presented in the last subsection in case of any violation of GIP or TBP. Upon the return of Topology, $\mathrm{Tri}\, S|_\Sigma$ is homeomorphic to Σ. However, it is possible that some seeds are too close together, which means that the surface triangulation may be denser than necessary around the seeds. We fix this problem by deleting the seeds incrementally. One may observe that we may start the algorithm with a single initial point and then let Silhouette generate more points on each component of Σ instead of computing the seed set S_0. However, to avoid the more expensive computation of critical points of the silhouette in a given direction, we recommend starting with the seed set S_0.

```
SampleTopology(surface $\Sigma$)
  1. Compute $S_0 :=$ SurfaceCritical($\Sigma$).
  2. Compute $S :=$ Topology($S_0$).
  3. While there is a seed $p \in S$, delete $p$ from $S$ and compute
     $S :=$ Topology($S$).
  4. Return $S$.
Topology(sample set $S$)
  1. Perform steps (a)–(e) in order. Terminate the current step as
     soon as the returned $x$ is nonnull, skip the following steps, and
     go to step 2.
     (a) For every edge $e$ of Vor $S$, compute $x :=$ VorEdge($e$).
     (b) For every facet $F$ of Vor $S$, compute $x :=$ FacetContact($F$).
     (c) For every $p \in S$, compute $x :=$ TopoDisk($p$).
     (d) For every facet $F$ of Vor $S$, compute $x :=$ FacetCycle($F$).
     (e) For every $p \in S$, compute $x :=$ Silhouette($p$).
  2. If $x$ is nonnull, insert $x$ into $S$, update Vor $S$, and go to step 1.
     Otherwise, return $S$.
```

5.5. Analysis. We first prove that any point x inserted in step 2 of Topology is at distance $\lambda f(p)$ or more from its closest sample p in S. This shows that Topology terminates after introducing only finitely many points on Σ.

Lemma 5.1. *Let x be a point inserted by* Topology *into S. The distance from x to its closest sample p in the current S is at least $\lambda f(p)$.*

Proof. If x is returned by VorEdge(e) for some edge e of a Voronoi cell V_p, Lemma 4.7 implies that $\|p - x\| \geq \lambda f(p)$.

If x is returned by FacetContact(F) (resp., FacetCycle(F)) for some facet F of a Voronoi cell V_p, then $\|p - x\| \geq \lambda f(p)$ by Lemma 4.8 (resp., Lemma 4.9).

If x is returned by TopoDisk(p) for some sample $p \in S$, the set T_p of triangles in Tri $S|_\Sigma$ incident to p do not form a topological disk. There are three possibilities.

- *Case* 1. Some triangle edge in T_p is not shared by another triangle in T_p. Let $F \subset V_p$ be the dual Voronoi facet of this triangle edge. This case happens because Σ intersects only one boundary edge e of F. Since Σ has no boundary, the endpoint(s) of $F \cap \Sigma$ must lie on e. Thus, e intersects Σ in more than one point or e intersects Σ tangentially. But this is impossible because VorEdge(e) did not return any point.
- *Case* 2. Three or more triangles in T_p share an edge. Let $F \subset V_p$ be the dual Voronoi facet of this common edge. This case happens because Σ intersects three or more distinct boundary edges of F. Since VorEdge did not return any point for all edges of Vor S, every edge of F intersects Σ in at most one point (nontangentially). Also, Σ meets F transversally as FacetContact did not return any point. It follows that $F \cap \Sigma$ contains two or more topological intervals. Thus, Lemma 4.10 applies, implying that $\|p - x\| \geq \lambda f(p)$.
- *Case* 3. T_p contains two or more cycles of triangles around p. It implies that $V_p \cap \Sigma$ has at least two boundary cycles C_1 and C_2, and that if C_i is nondegenerate, C_i is not contained in a facet of V_p. Thus, Lemma 4.12 applies, implying that $\|p - x\| \geq \lambda f(p)$.

If x is returned by Silhouette(p) for some $p \in S$, then $\|p - x\| \geq \lambda f(p)$ by Lemma 4.14. □

The next result shows a useful property of the Voronoi diagram enforced by step 1 of SampleTopology.

LEMMA 5.2. *If a sample set S includes all critical points of Σ in the x_3-direction, then no connected component of Σ is contained in a Voronoi cell in* $\operatorname{Vor} S$.

Proof. The critical points of any connected component M in the x_3-direction belong to S by assumption. Each connected component has at least two critical points. Thus, if M is contained inside a Voronoi cell V_p, V_p must contain two seed points. This is a contradiction to the emptiness of V_p even though p may be one of the seed points. □

We are ready to show that at the end of step 2 of SAMPLETOPOLOGY, GIP and TBP are satisfied.

LEMMA 5.3. *Let S_0 be a sample set such that no connected component of Σ is contained in a Voronoi cell in* $\operatorname{Vor} S_0$. *Let Σ' be the surface obtained by perturbing Σ as presented in section* 5.1. *Then* TOPOLOGY(S_0) *returns a set S such that* $\operatorname{Vor} S$ *satisfies GIP and TBP with respect to Σ'.*

Proof. By Lemma 5.1 TOPOLOGY(S_0) maintains a positive lower bound on interpoint distances for the points it inserts. This means it can insert only finitely many points on a compact surface Σ. Therefore, it must terminate and return a sample set S. VOREDGE and FACETCONTACT make sure that Σ intersects all edges and facets of $\operatorname{Vor} S$ transversally. Any vertex of $\operatorname{Vor} S$ on Σ is perturbed by the method presented in section 5.1. Thus, $\operatorname{Vor} S$ satisfies GIP for Σ'.

VOREDGE guarantees that Σ does not intersect any Voronoi edge in $\operatorname{Vor} S$ in more than one point at the end of TOPOLOGY. Neither does Σ'.

Consider a Voronoi facet F in $\operatorname{Vor} S$. The intersection $F \cap \Sigma'$ cannot contain more than two or more topological intervals. (Because of the perturbation, any topological interval in $F \cap \Sigma'$ is nondegenerate.) Otherwise, Σ' must intersect at least four boundary edges of F because Σ' intersects any edge of F in at most one point. This means that the dual Delaunay edge of F is incident to more than two triangles in $\operatorname{Tri} S|_\Sigma$. But then TOPODISK should have detected this, a contradiction. The intersection $F \cap \Sigma'$ cannot contain any cycle. Otherwise, $F \cap \Sigma$ would contain the same cycle and TOPOLOGY would not have terminated because FACETCYCLE(F) would have returned a point. In all, if $F \cap \Sigma'$ is nonempty, it is a single topological interval.

Consider a Voronoi cell V_p in $\operatorname{Vor} S$. The intersection $V_p \cap \Sigma'$ is a manifold possibly with boundary. By assumption, no Voronoi cell of $\operatorname{Vor} S_0$ contains a connected component of Σ. Hence, no Voronoi cell of $\operatorname{Vor} S_0$ contains a connected component of Σ' too. Since $S_0 \subseteq S$, no connected component of Σ' is contained in any Voronoi cell in $\operatorname{Vor} S$ either.

Can $V_p \cap \Sigma'$ have two or more boundary cycles? If so, $V_p \cap \Sigma$ also has two or more boundary cycles. (Some may degenerate to a vertex of V_p.) VOREDGE guarantees that the boundary of $V_p \cap \Sigma$ does not intersect the same edge of V_p twice. FACETCYCLE guarantees that no nondegenerate boundary cycle of $V_p \cap \Sigma$ is contained in a facet of V_p. Therefore, the boundary cycles of $V_p \cap \Sigma$ must induce at least two cycles of triangles in $\operatorname{Tri} S|_\Sigma$ around p. But then TOPODISK(p) should have detected this, a contradiction.

We conclude that $V_p \cap \Sigma'$ must be connected and have a single boundary cycle. Assume to the contrary that $V_p \cap \Sigma'$ is not a topological disk; i.e., its genus is positive. The perturbation scheme in section 5.1 does not create any component inside V_p with positive genus. Thus, $V_p \cap \Sigma$ has a connected component M with positive genus. So M is not a single vertex of V_p; i.e., the boundary cycle of M is nondegenerate. Lemma 4.13 implies that M must intersect the silhouette of Σ with respect to direction n_p. But then TOPOLOGY would not have terminated because SILHOUETTE(p) would have returned a point, a contradiction. □

Corollary 5.4. *Let S_0 be a sample set such that no connected component of Σ is contained in any Voronoi cell in* $\operatorname{Vor} S_0$. *Then* Topology(S_0) *returns a set S where the underlying space of* $\operatorname{Tri} S|_\Sigma$ *is homeomorphic to Σ.*

Proof. Let Σ' be the surface obtained by perturbing Σ as presented in section 5.1. By Lemma 5.3, $\operatorname{Tri} S|_\Sigma = \operatorname{Del} S|_{\Sigma'}$ is homeomorphic to Σ' and hence to Σ. ∎

The lower bound in Lemma 5.1 on the distances from any new point inserted by Topology to existing samples is instrumental to the analysis of the size of the final triangulation. However, the seeds inserted in step 1 of SampleTopology can be arbitrarily close together (unlikely in practice though). Therefore, we delete the seeds one by one in step 3 of SampleTopology. The GIP or TBP may become invalid after the deletion of a seed. Thanks to the next lemma, we can restore it by running Topology.

Lemma 5.5. *Suppose that no Voronoi cell in* $\operatorname{Vor} S$ *contains a connected component of Σ and no Voronoi facet in* $\operatorname{Vor} S$ *intersects Σ in a cycle. For any $p \in S$, no connected component of Σ is contained in a Voronoi cell in* $\operatorname{Vor}(S \setminus \{p\})$.

Proof. Assume to the contrary that a connected component M of Σ is contained in the Voronoi cell of a sample q in $\operatorname{Vor}(S \setminus \{p\})$. Consider the insertion of p into $S \setminus \{p\}$ and the corresponding update of the Voronoi diagram. Let H be the bisecting plane of p and q. If H does not intersect M, the Voronoi cell of p or q in $\operatorname{Vor} S$ would contain M, contradicting our assumption. If H intersects M, then $H \cap M$ would contain a cycle in the facet between the Voronoi cells of p and q in $\operatorname{Vor} S$, a contradiction again. ∎

Lemma 5.5 enables us to invoke Corollary 5.4 after the deletion of one seed. This shows that $\operatorname{Tri} S|_\Sigma$ will be homeomorphic to Σ after running Topology.

Corollary 5.6. *Following each deletion of a seed in step* 3 *of* SampleTopology, *the invocation of* Topology *guarantees that the underlying space of* $\operatorname{Tri} S|_\Sigma$ *is homeomorphic to Σ afterward.*

6. Meshing algorithm.

Our meshing algorithm DelMesh first invokes the algorithm SampleTopology to capture the topology of Σ. Topological guarantee alone is not sufficient for many applications. In finite element methods, it is important that the surface triangles have bounded aspect ratio. Also, the output approximation should be smooth enough as it approximates a smooth surface. We introduce two procedures to address these issues in section 6.1. Notice that these two procedures help in capturing the geometry of Σ but cannot guarantee a nontrivial upper bound on the Hausdorff distances between input and output relative to the local feature sizes. It seems that such a guarantee would require computing the local feature sizes explicitly, which we want to avoid. In section 6.2, we give the complete description of DelMesh and its analysis.

6.1. Geometry sampling.

Given a triangle t, define $\rho(t)$ to be the ratio of the circumradius of t to the shortest side length of t. It is well known that t has bounded aspect ratio if $\rho(t)$ is bounded from above by some constant. Following Chew [15], if there is a triangle t in $\operatorname{Tri} S|_\Sigma$ with $\rho(t) > 1+\lambda$, the procedure Quality below inserts into S the intersection point between Σ and the dual Voronoi edge of t.

```
Quality(sample set $S$)
  1. While there is a triangle $t$ in $\operatorname{Tri} S|_\Sigma$ with $\rho(t) > 1 + \lambda$, do the
     following:
     (a) Compute an intersection point $x$ between $\Sigma$ and the dual
         Voronoi edge of $t$. (Arbitrarily choose one if there are more
         than one.)
```

```
        (b) Insert $x$ into $S$ and update Vor $S$.
    2. Return $S$.
```

Assuming that $\mathrm{Tri}\, S|_\Sigma$ is an orientable 2-manifold without boundary, we measure its smoothness using the dihedral angles at the edges. Specifically, for each edge e in $\mathrm{Tri}\, S|_\Sigma$, we define the *roughness* of e, denoted by $g(e)$, to be π minus the dihedral angle at e. The procedure SMOOTH below samples a point from Σ if the roughness of some edge exceeds $2\beta(\lambda)$.

```
SMOOTH(sample set $S$)
    1. If there is an edge $pq$ in $\mathrm{Tri}\, S|_\Sigma$ such that $g(pq) > 2\beta(\lambda)$, do the
       following:
        (a) Compute the intersections between $\Sigma$ and the dual Voronoi
            edges of the triangles in $\mathrm{Tri}\, S|_\Sigma$ incident to $pq$.
        (b) Pick the furthest intersection point $x$ from $p$.
        (c) Insert $x$ into $S$ and update Vor $S$.
    2. Return $S$.
```

QUALITY enforces that the angles of every triangle are no less than $\arcsin(\frac{1}{2+2\lambda})$. SMOOTH enforces that the dihedral angles are no less than $\pi - O(\lambda)$. Thus, we can improve the triangle shape and smoothness by decreasing λ. However, as explained in Theorem 6.2, the mesh size increases linearly in $\frac{1}{\lambda^2}$.

6.2. Finale. We give the pseudocode of DELMESH below. DELMESH maintains the sample set S and $\mathrm{Tri}\, S|_\Sigma$ throughout its execution. The final triangulation $\mathrm{Tri}\, S|_\Sigma$ is the output surface mesh desired.

```
DELMESH(surface $\Sigma$)
    1. Compute $S :=$ SAMPLETOPOLOGY($\Sigma$).
    2. Compute $S :=$ QUALITY($S$). If QUALITY inserted some point(s)
       into $S$, compute $S :=$ TOPOLOGY($S$) and repeat step 2.
    3. Compute $S :=$ SMOOTH($S$). If SMOOTH inserted a point into
       $S$, compute $S :=$ TOPOLOGY($S$) and go to step 2.
    4. Output $\mathrm{Tri}\, S|_\Sigma$.
```

Notice that after QUALITY or SMOOTH inserts new sample point(s), we call TOPOLOGY again because the new sample point(s) may disturb the topology of $\mathrm{Tri}\, S|_\Sigma$. It is worthwhile to note that one does not need to search the entire Vor S for possible topology violation. Instead, since the insertion of a new point changes Vor S locally, a local search suffices.

We need the following technical result to analyze DELMESH.

LEMMA 6.1. *Let x be a point whose distance to the nearest sample $p \in S$ is at least $\lambda f(p)$. For any point $q \in S$,* (i) $\|q - x\| \geq \lambda f(x)/(1+\lambda)$ *and* (ii) $\|q - x\| \geq \lambda f(q)/(1+2\lambda)$.

Proof. By the Lipschitz condition, $f(x) \leq f(p) + \|p - x\| \leq (1+\lambda)\, \|p - x\|/\lambda$. Since p is the nearest sample to x, for any $q \in S$, $\|q - x\| \geq \|p - x\| \geq \lambda f(x)/(1+\lambda)$. By the Lipschitz condition again, $f(q) \leq f(x) + \|q - x\| \leq (1+2\lambda)\, \|q - x\|/\lambda$. □

We are ready to prove the performance guarantees of DELMESH.

THEOREM 6.2. *Let $\lambda \leq 0.06$ be a constant chosen a priori. Given a smooth closed surface Σ,* DELMESH(Σ) *outputs a mesh* $\mathrm{Tri}\, S|_\Sigma$ *consisting of Delaunay triangles such that the following hold*:

(i) *The underlying space of* $\mathrm{Tri}\, S|_\Sigma$ *is homeomorphic to* Σ.

(ii) *The radius-edge ratio of every triangle in* $\mathrm{Tri}\, S|_\Sigma$ *is at most* $1+\lambda$, *and the roughness of every edge of* $\mathrm{Tri}\, S|_\Sigma$ *is at most* $2\beta(\lambda)$.

(iii) *For any $\varepsilon < 1/5$, the size of S is within a factor $O(\frac{\varepsilon^2}{\lambda^2})$ of the size of an ε-sample of Σ.*

Proof. Take a nonseed point x inserted by DelMesh. We prove by induction that $\|p - x\| \geq \lambda f(p)$, where p is a nearest sample in S to x at the time of insertion.

If x is a nonseed point inserted by SampleTopology, then $\|p - x\| \geq \lambda f(p)$ by Lemma 5.1.

Suppose that x is inserted by Quality. So x is an intersection point between Σ and the dual Voronoi edge of some triangle t in $\mathrm{Tri}\, S|_\Sigma$. Let pq be the shortest edge of t. Since S does not contain any seed point at the end of SampleTopology, p and q are nonseed points inserted some time in the past. Without loss of generality, assume that p was inserted after q. Then $\|p - q\| \geq \lambda f(p)/(1 + \lambda)$ by induction assumption and Lemma 6.1(i). Observe that p and the other vertices of t are the nearest samples to x when x is inserted. Since $\rho(t) > 1 + \lambda$, $\|p - x\| > (1 + \lambda)\, \|p - q\| > \lambda f(p)$.

Suppose that x is inserted by Smooth, triggered by an edge pq in $\mathrm{Tri}\, S|_\Sigma$ whose roughness is greater than $2\beta(\lambda)$. Let pqr and pqs be the two triangles in $\mathrm{Tri}\, S|_\Sigma$ incident to pq. So among the intersection points between Σ and the dual Voronoi edges of pqr and pqs, x is the furthest one from p. Observe that p is a nearest sample in S to x. Assume to the contrary that $\|p - x\| < \lambda f(p)$. This implies that the circumradii of pqr and pqs are less than $\lambda f(p)$. Let d and d' be the outward normals of pqr and pqs, respectively. By Lemma 2.3, $\angle d, n_p \leq \beta(\lambda)$ and $\angle d', n_p \leq \beta(\lambda)$. Therefore, $\angle d, d' \leq 2\beta(\lambda)$, which means that the roughness of pq is at most $2\beta(\lambda)$, contradicting the insertion of x. Hence, $\|p - x\| \geq \lambda f(p)$.

In all, when x is inserted by DelMesh, the distance from x to its nearest sample p in S is at least $\lambda f(p)$. Therefore, DelMesh terminates by a packing argument.

By Lemma 5.2, Corollary 5.4, and Corollary 5.6, $\mathrm{Tri}\, S|_\Sigma$ is homeomorphic to Σ, and no Voronoi cell in $\mathrm{Vor}\, S$ contains a connected component of Σ at the end of the first call to SampleTopology. We claim that the subsequent insertion of new point(s) to repair the geometry preserves the property that no connected component of Σ is contained in any Voronoi cell. This can be seen as follows. Suppose that we add a new point p to repair the geometry after SampleTopology. Any existing Voronoi cell can only shrink, and the shrunken Voronoi cell cannot contain a component. If the new Voronoi cell V_p contains a component M, either p is the only vertex sampled from M or no point is sampled from M at all. But this is impossible because SampleTopology guarantees that $\mathrm{Tri}\, S|_\Sigma$ is homeomorphic to Σ, which means $\mathrm{Tri}\, S|_\Sigma$ has a vertex from every connected component of Σ before we insert any point to repair the geometry. By our claim, one can argue inductively that no Voronoi cell in $\mathrm{Vor}\, S$ contains a connected component of Σ throughout the execution of DelMesh. Then, (i) follows from Corollary 5.4. The correctness of (ii) follows from the termination of DelMesh.

For the rest of the proof, S denotes the final sample set obtained by DelMesh. Erickson [25] proved that for any $\varepsilon < 1/5$, the size of an ε-sampling of Σ is

$$\Omega\left(\int_\Sigma \frac{1}{\varepsilon^2 f(x)^2}\, dx\right).$$

Therefore, to prove (iii), it suffices to show that the size of S is $O(\frac{1}{\lambda^2}) \cdot \int_\Sigma \frac{1}{f(x)^2}\, dx)$.

Let q and r be two points in S. Irrespective of whether DelMesh inserted q or r first, Lemma 6.1(i) and (ii) imply that $\|q - r\| \geq \lambda f(q)/(1 + 2\lambda)$. Therefore, if we put a ball B_q centered at q with radius $\lambda f(q)/(2 + 4\lambda)$ for each $q \in S$, the balls have disjoint interior. It follows that

$$\int_\Sigma \frac{1}{f(x)^2}\,dx \;\geq\; \sum_{q\in S}\int_{B_q\cap\Sigma}\frac{1}{f(x)^2}\,dx.$$

For each point $x \in B_q \cap \Sigma$, the Lipschitz condition implies that $f(x) \leq f(q) + \|q - x\| \leq (2+5\lambda)f(q)/(2+4\lambda)$. Hence,

$$\int_\Sigma \frac{1}{f(x)^2}\,dx \;\geq\; \sum_{q\in S}\int_{B_q\cap\Sigma}\frac{4(1+2\lambda)^2}{(2+5\lambda)^2 f(q)^2}\,dx.$$

By Lemma 2.4, $\angle qx, n_q \geq \arccos(\lambda/(4+8\lambda))$. Let x' be the orthogonal projection of x onto the tangent plane of Σ at q. It follows that $\|q - x'\| \geq \|q - x\| \cdot \cos(\arcsin(\lambda/(4+8\lambda))$, which is greater than $\|q-x\|/\sqrt{2}$ for $\lambda \leq \lambda_0 = 0.06$. By a result of Boissonnat and Cazals [5], $B_q \cap \Sigma$ is a topological disk as radius$(B_q) < f(q)$. Moreover, it can be verified using Lemmas 2.4 and 2.2 that $B_q \cap \Sigma$ is monotone in direction n_q. We conclude that the orthogonal projection of $B_q \cap \Sigma$ onto the tangent plane of Σ at q covers a disk centered at q with radius $\frac{\lambda}{2\sqrt{2}(1+2\lambda)} f(q)$. Hence, area$(B_q \cap \Sigma) \geq \frac{\pi\lambda^2}{8(1+2\lambda)^2}\, f(q)^2$. Therefore,

$$\int_\Sigma \frac{1}{f(x)^2}\,dx \;\geq\; \sum_{q\in S}\int_{B_q\cap\Sigma}\frac{\pi\lambda^2}{2(2+5\lambda)^2\,\mathrm{area}(B_q\cap\Sigma)}\,dx \;=\; \frac{\pi\lambda^2\,|S|}{2(2+5\lambda)^2}. \qquad \square$$

7. Discussions. We presented a provable algorithm for sampling and meshing a smooth surface without boundary. Implicit surfaces can be meshed with this algorithm, which offers guarantees on the topology, triangular shape, smoothness, and size of the output triangulation. The mesh is Delaunay. It is worthwhile to note that we also obtain a Delaunay meshing of the volume bounded by the output surface mesh.

We implemented a simplified version of DelMesh using CGAL [10]. We did not implement the FacetContact, FacetCycle, and Silhouette subroutines. Figure 12 shows the results of this implementation for some simple smooth surfaces. Although the theory applies to smooth surfaces, we experimented with some triangulated surfaces obtained by a surface reconstruction software called Tight Cocone [21]. Although these surfaces already have sample points, we disregarded all these sample points for our experiments and considered the piecewise linear surface as input. For each surface, DelMesh generated a new set of sample points and the corresponding restricted Delaunay triangulation. Figure 13 shows these triangulations.

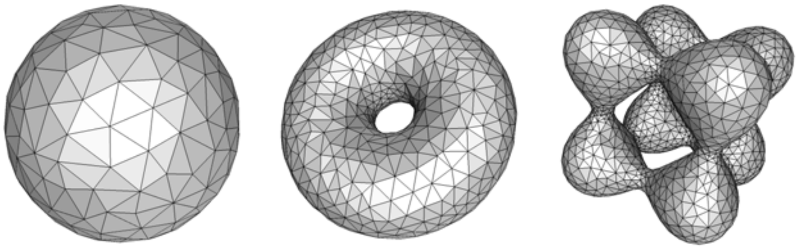


Fig. 12. *Meshing of a smooth sphere, torus, and a metaball, each of which is input with an implicit equation.*

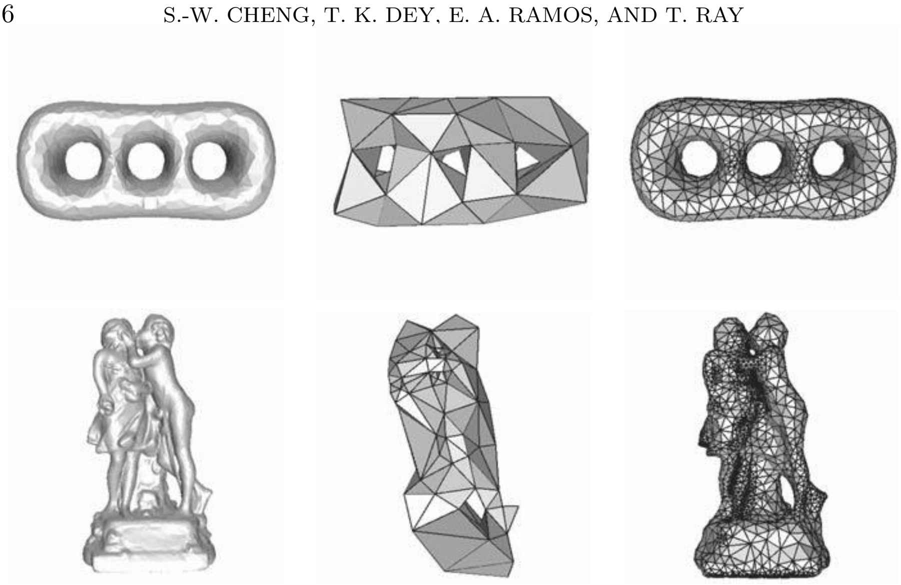


Fig. 13. *The first column shows the surfaces to be sampled. The second column shows the triangulations after capturing the topology and deleting the seeds. The third column shows the results after improving the triangular shape and smoothing.*

These examples show that DelMesh can be used for remeshing triangulated surfaces while guaranteeing bounded aspect ratio. An open question remains if the method or its variant can be proved to mesh nonsmooth surfaces with guarantees. This question is partially addressed by Dey, Li, and Ray [20] and Boissonnat and Oudot [9].

The critical point computations are the most costly computations in DelMesh. Can we avoid them and under what circumstances?

Acknowledgments. We thank Günter Rote immensely for detecting an error in an earlier draft which led us to introduce the concept of silhouette. We also thank the anonymous referees whose detailed and thoughtful comments were very helpful in improving the paper.

REFERENCES

[1] N. Amenta and M. Bern, *Surface reconstruction by Voronoi filtering*, Discrete Comput. Geom., 22 (1999), pp. 481–504.

[2] N. Amenta, S. Choi, T. K. Dey, and N. Leekha, *A simple algorithm for homeomorphic surface reconstruction*, Internat. J. Comput. Geom. Appl., 12 (2002), pp. 125–141.

[3] C. L. Bajaj, *Surface fitting using implicit algebraic surface patches*, in Topics in Surface Modeling, H. Hagen, ed., SIAM, Philadelphia, 1992, pp. 23–52.

[4] J. Bloomenthal, *Polygonization of implicit surfaces*, Comput. Aided Geom. Design, 5 (1988), pp. 341–355.

[5] J.-D. Boissonnat and F. Cazals, *Natural neighbor coordinates of points on a surface*, Comput. Geom., 19 (2001), pp. 155–173.

[6] J.-D. Boissonnat, D. Cohen-Steiner, B. Mourrain, G. Rote, and G. Vegter, *Meshing of surfaces*, in Effective Computational Geometry for Curves and Surfaces, J.-D. Boissonnat and M. Teillaud, eds., Springer-Verlag, Berlin, 2007, pp. 181–229.

[7] J.-D. Boissonnat, D. Cohen-Steiner, and G. Vegter, *Isotopic implicit surface meshing*, in Proceedings of the 36th Annual ACM Symposium on Theory of Computing, 2004, pp. 301–309.

[8] J.-D. Boissonnat and S. Oudot, *Provably good sampling and meshing of surfaces*, Graph. Models, 67 (2005), pp. 405–451.

[9] J.-D. Boissonnat and S. Oudot, *Provably good sampling and meshing of Lipschitz surfaces*, in Proceedings of the 22nd Annual ACM Symposium on Computational Geometry, 2006, pp. 337–346.

[10] *CGAL-Computational Geometry Algorithms Library*, http://www.cgal.org.

[11] H.-L. Cheng, T. K. Dey, H. Edelsbrunner, and J. Sullivan, *Dynamic skin triangulation*, Discrete Comput. Geom., 25 (2001), pp. 525–568.

[12] S.-W. Cheng and T. K. Dey, *Quality meshing with weighted Delaunay refinement*, SIAM J. Comput., 33 (2003), pp. 69–93.

[13] S.-W. Cheng, T. K. Dey, E. A. Ramos, and T. Ray, *Quality meshing for polyhedra with small angles*, Internat. J. Comput. Geom. Appl., 15 (2005), pp. 421–461.

[14] S.-W. Cheng and S.-H. Poon, *Three-dimensional Delaunay mesh generation*, Discrete Comput. Geom., 36 (3006), pp. 419–456.

[15] L. P. Chew, *Guaranteed-quality mesh generation for curved surfaces*, in Proceedings of the Ninth Annual ACM Symposium on Computational Geometry, 1993, pp. 274–280.

[16] D. Cohen-Steiner, É. Colin de Verdière, and M. Yvinec, *Conforming Delaunay triangulations in 3D*, in Proceedings of the 18th Annual ACM Symposium on Computational Geometry, 2002, pp. 199–208.

[17] J.C. Cuillière, *An adaptive method for the automatic triangulation of 3D parametric surfaces*, Computer-Aided Design, 30 (1998), pp. 139–149.

[18] T. K. Dey and W. Zhao, *Approximating the medial axis from the Voronoi diagram with a convergence guarantee*, Algorithmica, 38 (2004), pp. 179–200.

[19] T. K. Dey, *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*, Cambridge University Press, New York, 2006.

[20] T. K. Dey, G. Li, and T. Ray, *Polygonal surface meshing with Delaunay refinement*, in Proceedings of the 14th International Meshing Roundtable, 2005, pp. 343–351.

[21] T. Dey, J. Giesen, S. Goswami, J. Hudson, and W. Zhao, *COCONE Software for Surface Reconstruction and Medial Axis*, http://www.cse.ohio-state.edu/~tamaldey/cocone.html.

[22] H. Edelsbrunner, *Deformable smooth surface design*, Discrete Comput. Geom., 21 (1999), pp. 87–115.

[23] H. Edelsbrunner and E. Mücke, *Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms*, ACM Trans. Graph., 9 (1990), pp. 66–104.

[24] H. Edelsbrunner and N. Shah, *Triangulating topological spaces*, Internat. J. Comput. Geom. Appl., 7 (1997), pp. 365–378.

[25] J. Erickson, *Nice point sets can have nasty Delaunay triangulations*, Discrete Comput. Geom., 30 (2003), pp. 109–132.

[26] V. Guillemin and A. Pollack, *Differential Topology*, Prentice–Hall, Englewood Cliffs, NJ, 1974.

[27] T. G. Kolda, R. M. Lewis, and V. Torczon, *Optimization by direct search: New perspectives on some classical and modern methods*, SIAM Rev., 45 (2003), pp. 385–482.

[28] T. S. Lau and S. H. Lo, *Finite element mesh generation over analytical curved surfaces*, Comput. & Structures, 59 (1996), pp. 301–309.

[29] W. E. Lorensen and H. E. Cline, *Marching cubes: A high resolution 3D surface construction algorithm*, Comput. Graphics, 21 (1987), pp. 163–169.

[30] B. Mourrain and J.-P. Técourt, *Isotopic meshing of a real algebraic surface*, Technical report RR-5508, INRIA, Sophia Antipolis, France, 2005.

[31] S. Plantinga and G. Vegter, *Isotopic approximation of implicit curves and surfaces*, in Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, 2004, pp. 245–254.

[32] J. R. Shewchuk, *Tetrahedral mesh generation by Delaunay refinement*, in Proceedings of the 14th Annual ACM Symposium on Computational Geometry, 1998, pp. 86–95.

[33] J. R. Shewchuk, *What is a good linear element? Interpolation, conditioning, and quality measures*, in Proceedings of the 11th International Meshing Roundtable, 2002, pp. 115–126.

[34] B. T. Stander and J. C. Hart, *Guaranteeing the topology of an implicit surface polygonalization for interactive modeling*, in Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, ACM Press, Addison–Wesley, New York, 1997, pp. 279–286.

[35] J. R. Tristano, S. J. Owen, and S. A. Canann, *Advancing front surface mesh generation in parametric space using a Riemannian surface definition*, in Proceedings of the 7th International Meshing Roundtable, 1998.

[36] A. P. Witkin and P. S. Heckbert, *Using particles to sample and control implicit surfaces*, in Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, ACM Press, New York, 1994, pp. 269–277.

SIAM J. COMPUT.
Vol. 37, No. 4, pp. 1228–1258

AN ISOMORPHISM BETWEEN SUBEXPONENTIAL AND PARAMETERIZED COMPLEXITY THEORY[*]

YIJIA CHEN[†] AND MARTIN GROHE[‡]

Abstract. We establish a close connection between (sub)exponential time complexity and parameterized complexity by proving that the so-called miniaturization mapping is a reduction preserving isomorphism between the two theories.

Key words. parameterized complexity, exponential time complexity, exponential time hypothesis, subexponential time

AMS subject classifications. 03D15, 03D30, 68Q15, 68Q17

DOI. 10.1137/070687153

1. Introduction.

The area of exact algorithms for hard algorithmic problems has received considerable attention in recent years (see, e.g., [18, 19]). The goal is to design exact, as opposed to approximative, algorithms for hard (usually NP-complete) problems that are better than the trivial brute force algorithms but may still be exponential. For example, the currently best deterministic algorithm for the 3-satisfiability problem (3-SAT) due to Brueggemann and Kern [3] has a running time of $O(1.473^n)$, and the best randomized algorithm due to Rolf [17] has a running time of $O(1.3222^n)$. Here n denotes the number of variables of the input formula.

Exponential complexity theory. A more qualitative question that has been recognized as central for the emerging theory of "exponential time complexity" is whether 3-SAT can be solved in time $2^{o(n)}$. The hypothesis that this is not possible is known as the *exponential time hypothesis* (ETH). It was first studied systematically by Impagliazzo, Paturi, and Zane [16], who proved that the hypothesis is very robust and equivalent to analogous hypotheses for many other NP-complete problems. For example, ETH is equivalent to the question of whether the INDEPENDENT-SET problem can be solved in time $2^{o(n)}$, where n denotes the number of vertices of the input graph. The reader may wonder why the running time is measured in terms of the number of variables and number of vertices, respectively, and not in the actual input size. The main contribution of Impagliazzo, Paturi, and Zane was to prove that the ETH is independent of the "size measure" (number of variables or actual size), that is, that 3-SAT is solvable in time $2^{o(n)}$ if and only if it is solvable in time $2^{o(m)}$ for the input size m.[1] It is easy to see that this implies the corresponding result for INDEPENDENT-SET and a number of further problems. However, in general, subexponential solvability is very sensitive with respect to the size measure, as the trivial example of the CLIQUE problem shows: Clearly, CLIQUE is solvable in time

[*] Received by the editors April 2, 2007; accepted for publication August 9, 2007; published electronically November 30, 2007.
http://www.siam.org/journals/sicomp/37-4/68715.html

[†] BASICS, Department of Computer Science, Shanghai Jiaotong University, Shanghai 200030, China (yijia.chen@cs.sjtu.edu.cn).

[‡] Institut für Informatik, Humboldt-Universität, Unter den Linden 6, 10099 Berlin, Germany (grohe@informatik.hu-berlin.de).

[1] Let us remark that here we assume that 3-CNF formulas have no repeated clauses and hence that $m = O(n^3)$. Otherwise the input size would not be bounded in terms of n, and hence the problem would not be solvable in time $f(n)$ for any function f. Later, this problem will disappear because we will always admit a polynomial of the input size as a factor of the running time.

$2^{o(n)}$ if and only if Independent-Set is, but Clique is trivially solvable in time $n^{O(\sqrt{m})} = 2^{o(m)}$, where m denotes the size of the input graph.

The example of the Clique problem illustrates that it is reasonable for a theory of exponential time complexity to view problems as pairs (P, ν) consisting of a decision problem $P \subseteq \Sigma^*$ over some finite alphabet Σ and a mapping $\nu : \Sigma^* \to \mathbb{N}$, which may be viewed as the *size measure*. For technical reasons, the mapping ν is required to be polynomial time computable. Typical size measures are the number of vertices or the number of edges for graph problems or the number of variables for satisfiability problems. Of course there is always the trivial size measure $\nu(x) = |x|$. At first sight, it seems reasonable to also require a size measure ν to be polynomially related to the input length, that is, $|x|^c \leq \nu(x) \leq |x|^d$ for some $c, d > 0$ and all $x \in \Sigma^*$. However, we prefer *not* to make this additional requirement, because there are problems where very natural size measures do not fulfill it. The most important example is the satisfiability problem Sat for arbitrary CNF formulas. The number of variables still seems to be one of the most natural complexity parameters, but it may be exponentially smaller than the input length. Hypergraph problems provide further natural examples; here the number of vertices is a natural parameter. Of course for such examples it is no longer appropriate to call ν a size measure, but other than that the role of ν remains the same.

But what would it mean for Sat to be subexponential with respect to "number of variables"? Clearly, Sat is not solvable in time $2^{o(n)}$, because the size m of the input formula may be $2^{\Omega(n)}$. The natural question is whether Sat is solvable in time $2^{o(n)} \cdot m^{O(1)}$. More generally, we say that a problem (P, ν) is *subexponential* if it is solvable in time

$$2^{o(\nu(x))} \cdot |x|^{O(1)} \tag{1.1}$$

for every instance x. Of course, if ν is polynomially related to the input length, then the term $|x|^{O(1)}$ is dominated by $2^{o(\nu(x))}$ and can hence be omitted. We denote the class of all subexponential problems by SUBEPT. We are interested in the dividing line between exponential and subexponential solvability; the problems we consider are usually (trivially) solvable in time $2^{O(\nu(x))} \cdot |x|^{O(1)}$. Let us denote the class of all of these problems by EPT.[2] The main advantage of our "parameterized" approach to exponential complexity is that it makes it easier to compare problems that otherwise would not be by "rescaling" them. For example, Planar-Independent-Set is solvable in time $2^{O(\sqrt{n})}$, and the question of whether it is solvable in time $2^{o(\sqrt{n})}$ is equivalent to the question of whether Independent-Set is solvable in time $2^{o(n)}$. If we rescale Planar-Independent-Set by letting $\nu(n) = \sqrt{n}$, we actually obtain a problem that is equivalent to Independent-Set with the size measure number of variables under suitable reductions. Thus by specifying size measures, we get a more robust theory.

To develop a complexity theory, we need suitable reductions. Impagliazzo, Paturi, and Zane [16] introduced so-called *subexponential reduction families*, which are a form of Turing reductions that preserve subexponential solvability. We essentially work with these reductions and also with a corresponding notion of many-one reductions.

The goals of *exponential complexity theory* may now be stated as classifying concrete problems within this framework and investigating the structure of the resulting complexity classes.

[2] This terminology has been introduced in [15] in the context of parameterized complexity theory; it will be explained in section 2.

Parameterized complexity theory. The reader familiar with parameterized complexity theory will have noticed that we are dealing with exactly the type of problems as considered there. A *parameterized problem* is a pair (Q, κ), where $Q \subseteq \Sigma^*$ for some finite alphabet Σ and $\kappa : \Sigma^* \to \mathbb{N}$, the *parameterization*, is polynomial time computable. We usually denote parameterized problems in the "exponential world" by (P, ν) and problems in the "parameterized world" by (Q, κ) because it will be important for us to separate the two worlds, but formally both are the same. However, the questions asked in parameterized complexity theory are different; parameterized complexity theory's main intention is to address complexity issues in situations where the parameter is expected to be small compared to the input size, whereas in the exponential theory the parameter was introduced as a size measure. A problem (Q, κ) is *fixed-parameter tractable* if it can be solved in time $f(\kappa(x)) \cdot |x|^{O(1)}$, where f is an arbitrary computable function. The class of all fixed-parameter tractable problems is denoted by FPT. There are suitable notions of *FPT (many-one) reduction* and *FPT Turing reduction.*

As opposed to exponential complexity theory, parameterized complexity theory has been developed in depth over the past fifteen years. A rich and perhaps unwieldy structure has emerged. The most important parameterized complexity classes are those of the *W-hierarchy.* Most natural parameterized problems are complete for one of these classes. Parameterized complexity theory almost exclusively studies problems which are in the class XP of all problems that can be solved in polynomial time for every fixed parameter value (see also Definition 13).

Our results. *Our main result is that exponential and parameterized complexity theory are isomorphic.* Let us explain what we mean by this: On the exponential side, we consider the partial order induced by subexponential reduction families on the *degrees*, that is, equivalence classes under subexponential reduction families. On the parameterized side, we consider the partial order induced by FPT reductions on the corresponding degrees. We define a mapping, the so-called *miniaturization mapping*, that associates a problem (Q, κ) with every problem (P, ν) and prove that this mapping is an isomorphism between the partial order of degrees inside EPT under subexponential reduction families and the partial order of degrees inside XP under FPT reductions. This result holds for both many-one and Turing reductions. In particular, miniaturization maps the class SUBEPT (the lowest "exponential degree") to FPT (the lowest "parameterized degree") and EPT to XP.

If we look at degrees outside of EPT, then the miniaturization mapping is still an embedding, but we prove that it is no longer onto. That is, there are parameterized degrees outside XP that are not in the image of the miniaturization mapping. Technically, this is our most difficult result.

The precise technical statement of our results requires some additional uniformity conditions. Essentially, the "little oh" in (1.1) has to be interpreted "effectively" (see section 2 for a precise statement). Alternatively, the uniformity condition in the definition of fixed-parameter tractability, requiring f to be computable, can be relaxed.

The miniaturization mapping is a fairly natural mapping between parameterized problems that has been studied before [4, 7, 9], albeit not as an abstract mapping between parameterized problems but just as a transformation between concrete problems such as vertex cover. As a matter of fact, there already is a body of work in parameterized complexity theory, starting with Abrahamson, Downey, and Fellows [1], that studies the relation between fixed-parameter tractability and exponential time complexity [4, 5, 6, 7, 9, 13]. Notably, it follows from this work that the miniaturization mapping maps the degree of 3-Sat with the number of variables size measure to a class M[1] between FPT and the first level W[1] of the W-hierarchy. The degree of Sat is mapped to a class M[2] between W[1] and W[2], and the degree of the satisfiability problem for Boolean circuits under the "number of input nodes" size measure is mapped to the class W[P]. This shows that the miniaturization isomorphism is not just an abstract mapping between partial orders but actually is a meaningful mapping between concrete problems and complexity classes. In the last section of this paper, we analyze the correspondence between the W-hierarchy and a natural hierarchy in the exponential theory, and we determine the preimage of the W-hierarchy under the miniaturization mapping.

2. Preliminaries.

In this section we give the necessary background of exponential and parameterized complexity. For more comprehensive details the reader is referred to [12, 14, 16].

We use $\mathbb{N}$ to denote the set of natural numbers (positive integers). For $m \leq n \in \mathbb{N}$ we let $[m, n] := \{m, m+1, \dots, n\}$ and $[n] := [1, n]$. A *classical decision problem* is a set $P \subseteq \Sigma^*$, where Σ is a finite alphabet. A *parameterized problem* is a pair (Q, κ), where $Q \subseteq \Sigma^*$ for some finite alphabet Σ and $\kappa : \Sigma^* \to \mathbb{N}$ is polynomial time computable. The mapping κ is called the *parameterization*. We occasionally write $x \in (Q, \kappa)$ instead of $x \in P$.

Exponential complexity. Recall that in the exponential theory we study parameterized problems (P, ν) and view the parameterization ν as a *size measure*. The most obvious size measure for a problem $P \subseteq \Sigma^*$ is the *length of the input* $\nu(x) = |x|$. Unfortunately, for most natural problems, the length of the input is not exactly what we think of as its "size." For example, we are used to thinking of the size of a graph G with n vertices and m edges as being $(m + n)$ rather than $\Theta(n + m \cdot \log n)$, which would be the length of a reasonable encoding of G over a finite alphabet. To abstract from such a heavy dependence on coding issues, we define the *size* $||G||$ of a graph G with n vertices and m edges to $m + n$. Hence we distinguish between size and "length (of an encoding)" of a graph. Similarly, we define the size $||C||$ of a Boolean circuit to be the number of gates plus the number of lines. We view Boolean formulas as special circuits, so they inherit the size measures for circuits. For example, for a CNF formula $\alpha = \bigwedge_{i=1}^{m} \bigvee_{j=1}^{k_i} \lambda_{ij}$ this means that $||\alpha|| = \Theta(\sum_{i=1}^{m} k_i)$. We denote the parameterization of a graph problem or satisfiability problem P by the input size by s-P. For example, we let

s-Sat
Instance: A CNF formula α.
Parameter: $||\alpha||$.
Problem: Decide whether α is satisfiable.

Other natural size measures are the "number of vertices" for graph problems and the number of variables for satisfiability problems. As examples, consider the following problems:

s-vert-Independent-Set
Instance: A graph $G = (V, E)$ and a $\ell \in \mathbb{N}$.
Parameter: $|V|$.
Problem: Decides whether G has an independent set of cardinality ℓ.

s-var-3-Sat	
Instance:	A 3-CNF formula α.
Parameter:	Number of variables of α.
Problem:	Decide whether α is satisfiable.

Similarly, we can define the graph problems *s-vert*-Clique, *s-vert*-Vertex-Cover, and *s-vert*-Dominating-Set and the satisfiability problems *s-var*-Sat (satisfiability of CNF formulas) and *s-var*-Circuit-Sat (satisfiability of Boolean circuits, parameterized by the number of input gates).

For two functions f, g with range $\mathbb{N}$, we say f is *effectively little oh* of g and write $f \in o^{\text{eff}}(g)$, if there is a computable function ι that is *nondecreasing* and *unbounded* such that

$$f = O\left(\frac{g}{\iota}\right).$$

Definition 1. *A parameterized problem (P, ν) is* subexponentially solvable *if there is an algorithm $\mathbb{A}$ that for every instance x decides whether $x \in P$ in time*

$$2^{o^{\text{eff}}(\nu(x))} \cdot |x|^{O(1)}.$$

By $2^{o^{\text{eff}}(\nu(x))}$ we mean $2^{f(x)}$ for some function $f \in o^{\text{eff}}(\nu)$. SUBEPT *denotes the class of all subexponentially solvable problems.*

As mentioned in the introduction, an example of a problem in SUBEPT is s-Clique, the clique problem parameterized by the input size, which is solvable in time $2^{O(\sqrt{m} \cdot \log m)}$, where m denotes the size of the input graph. Other, less trivial examples of problems in SUBEPT are the planar restrictions of many standard optimization problems parameterized by the number of vertices (or the size, which is equivalent for planar graphs). For example, the problems *s-vert*-Planar-Vertex-Cover, *s-vert*-Planar-Independent-Set, and *s-vert*-Planar-Dominating-Set are all solvable in time $2^{O(\sqrt{n})}$ and hence in SUBEPT [2]. However, most natural NP-complete problems do not seem to be in SUBEPT. To establish a completeness theory giving evidence to claims of problems not being in SUBEPT, we need an appropriate notion of reduction. The following lemma offers an alternative characterization of SUBEPT, which is the basis of the reductions we shall introduce afterwards.

Lemma 2. *Let (P, ν) be a parameterized problem over the alphabet Σ. The following are equivalent:*

(1) $(P, \nu) \in$ SUBEPT.

(2) *There is an algorithm $\mathbb{A}$ expecting inputs from $\Sigma^* \times \mathbb{N}$ and a computable function f such that for all $(x, \ell) \in \Sigma^* \times \mathbb{N}$ the algorithm $\mathbb{A}$ decides if $x \in P$ in time $f(\ell) \cdot 2^{\nu(x)/\ell} \cdot |x|^{O(1)}$.*

(3) *There is an algorithm $\mathbb{A}$ expecting inputs from $\Sigma^* \times \mathbb{N}$, a computable function f, and a constant $c \in \mathbb{N}$ such that for all $(x, \ell) \in \Sigma^* \times \mathbb{N}$ the algorithm $\mathbb{A}$ decides if $x \in P$ in time $f(\ell) \cdot 2^{c \cdot \nu(x)/\ell} \cdot |x|^{O(1)}$.*

Proof. (1) $\Rightarrow$ (2): Assume $(P, \nu) \in$ SUBEPT. Let ι be a computable function that is nondecreasing and unbounded, and let $\mathbb{A}$ be an algorithm deciding $x \in P$ in time $2^{c \cdot \nu(x)/\iota(\nu(x))} \cdot |x|^{O(1)}$ for some constant $c \in \mathbb{N}$. For $\ell \in \mathbb{N}$, let $n(\ell) := \max(\{n \mid \iota(n) < c \cdot \ell\} \cup \{1\})$ and $f(\ell) := 2^{c \cdot n(\ell)}$. Then for all $(x, \ell) \in \Sigma^* \times \mathbb{N}$ we have $2^{c \cdot \nu(x)/\iota(\nu(x))} \cdot |x|^{O(1)} \leq f(\ell) \cdot 2^{\nu(x)/\ell} \cdot |x|^{O(1)}$. Let $\mathbb{A}'$ be the algorithm that, given $(x, \ell) \in \Sigma^* \times \mathbb{N}$, simply ignores ℓ and simulates $\mathbb{A}$ on input x. Then $\mathbb{A}'$ and f satisfy the conditions of (2).

The direction from (2) to (3) is trivial. We turn to (3) ⇒ (1): Let $f : \mathbb{N} \to \mathbb{N}$ be a computable function, $c \in \mathbb{N}$ a constant, and $\mathbb{A}$ an algorithm that, given $(x, \ell) \in \Sigma^* \times \mathbb{N}$, decides if $x \in P$ in time $f(\ell) \cdot 2^{c \cdot \nu(x)/\ell} \cdot |x|^{O(1)}$. Without loss of generality, we may assume f is increasing and time-constructible. Let $\iota : \mathbb{N} \to \mathbb{N}$ be the following computable function: $\iota(n) := \max(\{\ell \mid f(\ell) < n\} \cup \{1\})$, which is clearly nondecreasing, unbounded, and computable in time polynomial in n. Let $\mathbb{A}'$ be the following algorithm for deciding P: Given $x \in \Sigma^*$, first compute $n := \nu(x)$ and $\ell := \iota(n)$, and then simulate $\mathbb{A}$ on (x, ℓ). The running time of $\mathbb{A}$ is bounded by

$$\begin{aligned} &|x|^{O(1)} + n^{O(1)} + f(\iota(n)) \cdot 2^{c \cdot n/\iota(n)} \cdot |x|^{O(1)} \\ \leq\ &|x|^{O(1)} + n^{O(1)} + O(n) \cdot 2^{o^{\mathrm{eff}}(n)} \cdot |x|^{O(1)} \\ =\ &2^{o^{\mathrm{eff}}(n)} \cdot |x|^{O(1)}. \qquad \square \end{aligned}$$

Our notion of reduction is essentially that of *subexponential reduction families*, as introduced in [16]. The reduction families in [16] are Turing reductions; we also introduce a many-one version.

DEFINITION 3. *Let (P, ν) and (P', ν') be parameterized problems over the alphabets Σ and Σ', respectively.*

(1) *A* subexponential reduction family*, or simply* serf reduction*, from (P, ν) to (P', ν') is a mapping $R : \Sigma^* \times \mathbb{N} \to (\Sigma')^*$ such that:*
  (a) *for all $(x, \ell) \in \Sigma^* \times \mathbb{N}$ we have $(x \in P \iff R(x, \ell) \in P')$;*
  (b) *given a pair $(x, \ell) \in \Sigma^* \times \mathbb{N}$, the image $R(x, \ell)$ is computable in time*

$$f(\ell) \cdot 2^{\nu(x)/\ell} \cdot |x|^{O(1)}$$

  for some computable function $f : \mathbb{N} \to \mathbb{N}$;
  (c) *There is a computable function $g : \mathbb{N} \to \mathbb{N}$ such that*

$$\nu'(R(x, \ell)) \leq g(\ell) \cdot (\nu(x) + \log |x|)$$

  for all $(x, \ell) \in \Sigma^* \times \mathbb{N}$.

(2) *A* subexponential Turing reduction family*, or* serf Turing reduction*, from (P, ν) to (P', ν') is an algorithm $\mathbb{A}$ with an oracle to P' such that there are computable functions $f, g : \mathbb{N} \to \mathbb{N}$ with the following:*
  (a) *Given a pair $(x, \ell) \in \Sigma^* \times \mathbb{N}$, the algorithm $\mathbb{A}$ decides if $x \in P$ in time*

$$f(\ell) \cdot 2^{\nu(x)/\ell} \cdot |x|^{O(1)}.$$

  (b) *For all oracle queries "$y \in P'$?" posed by $\mathbb{A}$ on input $(x, \ell) \in \Sigma^* \times \mathbb{N}$ it holds that*

$$\nu'(y) \leq g(\ell) \cdot (\nu(x) + \log |x|).$$

We write $(P, \nu) \leq^{\mathrm{serf}} (P', \nu')$ (or $(P, \nu) \leq^{\text{serf-T}} (P', \nu')$) if there is a serf reduction (serf Turing reduction, respectively) from (P, ν) to (P', ν'). We write $(P, \nu) \equiv^{\mathrm{serf}} (P', \nu')$ if $(P, \nu) \leq^{\mathrm{serf}} (P', \nu')$ and $(P', \nu') \leq^{\mathrm{serf}} (P, \nu)$.

It is clear that $(P, \nu) \leq^{\mathrm{serf}} (P', \nu')$ implies $(P, \nu) \leq^{\text{serf-T}} (P', \nu')$. It is not completely trivial that serf reductions preserve subexponential solvability.

PROPOSITION 4. *Let (P, ν) and (P', ν') be parameterized problems. If $(P, \nu) \leq^{\text{serf-T}} (P', \nu')$ and $(P', \nu') \in$ SUBEPT, then $(P, \nu) \in$ SUBEPT.*

Proof. Let Σ, Σ' be the alphabets of (P, ν), (P', ν'), respectively. Let $\mathbb{A}$ be a serf Turing reduction from (P, ν) to (P', ν'), and let f, g be the functions bounding the running time and the parameter. Let $f' : \mathbb{N} \to \mathbb{N}$ be a computable function and $\mathbb{A}'$ an algorithm that, given $(x', \ell') \in (\Sigma')^* \times \mathbb{N}$, decides if $x' \in P'$ in time $f'(\ell') \cdot 2^{\nu'(x')/\ell'} \cdot |x'|^{O(1)}$. Such $f', \mathbb{A}'$ exist by Lemma 2.

Let $\mathbb{B}$ be the algorithm that, given $(x, \ell) \in \Sigma^* \times \mathbb{N}$, simulates $\mathbb{A}$ and answers the oracle queries with instance x' by simulating $\mathbb{A}'$ on input (x', ℓ'), where $\ell' := g(\ell) \cdot \ell$. Observe that, since the running time of $\mathbb{A}$ is bounded by $f(\ell) \cdot 2^{\nu(x)/\ell} \cdot |x|^{O(1)}$, for each oracle query with instance x' we have $|x'| \leq f(\ell) \cdot 2^{\nu(x)/\ell} \cdot |x|^{O(1)}$. Also recall that, by the definition of subexponential reduction families, we have $\nu'(x') \leq g(\ell) \cdot (\nu(x) + \log |x|)$. Then $\mathbb{B}$ decides P, and the running time on input (x, ℓ) is bounded by

$$\begin{aligned} & f(\ell) \cdot 2^{\nu(x)/\ell} \cdot |x|^{O(1)} \cdot f'(\ell') \cdot 2^{\nu'(x')/\ell'} \cdot |x'|^{O(1)} \\ \leq\ & f(\ell) \cdot 2^{\nu(x)/\ell} \cdot |x|^{O(1)} \cdot f'(g(\ell) \cdot \ell) \\ & \qquad \cdot 2^{(\nu(x)+\log |x|)/\ell} \cdot f(\ell)^{O(1)} \cdot 2^{O(\nu(x)/\ell)} \cdot |x|^{O(1)} \\ \leq\ & f(\ell)^{O(1)} \cdot f'(g(\ell) \cdot \ell) \cdot 2^{O(\nu(x)/\ell)} \cdot 2^{\log |x|/\ell} \cdot |x|^{O(1)} \\ \leq\ & h(\ell) \cdot 2^{O(\nu(x)/\ell)} \cdot |x|^{O(1)} \end{aligned}$$

for a suitable computable function h. Thus by Lemma 2, $(P, \nu) \in$ SUBEPT. ◻

Example 5. For every graph problem P we trivially have *s-P* $\leq^{\mathrm{serf}}$ *s-vert-P*, and as a reduction family we can simply use the mapping $R(x, \ell) = x$. Similarly, for every satisfiability problem P we have *s-P* $\leq^{\mathrm{serf}}$ *s-var-P*.

It is a highly nontrivial result due to Impagliazzo, Paturi, and Zane [16] that, for many natural graph and satisfiability problems, the converse also holds. As a matter of fact, Impagliazzo, Paturi, and Zane proved that, the following problems are all equivalent with respect to serf Turing reductions: *s*-3-Sat, *s-var*-3-Sat, *s*-Independent-Set, *s-vert*-Independent-Set, *s*-Vertex-Cover, *s-vert*-Vertex-Cover, *s*-Dominating-Set, and *s-vert*-Clique.

Note that *s*-Clique does not appear in this list of problems. Indeed, it seems unlikely that *s*-Clique is equivalent to the problems above because it is in SUBEPT, whereas the other problems are not in SUBEPT unless the exponential time hypothesis (mentioned in the introduction) fails.

While unlikely to belong to SUBEPT, all problems considered in the previous example belong to the class EPT:

Definition 6. *A parameterized problem (P, ν) is in* EPT *if there is an algorithm $\mathbb{A}$ that, for every instance x, decides if $x \in P$ in time $2^{O(\nu(x))} \cdot |x|^{O(1)}$.*

It follows from the *time hierarchy theorem* that SUBEPT is a *proper* subclass of EPT. The next proposition shows that EPT is closed under serf Turing reductions.

Proposition 7. *Let (P, ν) and (P', ν') be parameterized problems. If (P, ν) $\leq^{\text{serf-T}}$ (P', ν') and $(P', \nu') \in$ EPT, then $(P, \nu) \in$ EPT.*

Proof. It is not hard to see that, from the serf Turing reduction from (P, ν) to (P', ν'), we can construct an algorithm $\mathbb{A}$ with an oracle to P' satisfying the following:

(R1) $\mathbb{A}$ decides if $x \in P$ in time $O(2^{\nu(x)} \cdot |x|^{O(1)})$ for any instance x.

(R2) For all oracle queries "$y \in P'$?" posed by $\mathbb{A}$ on input x we have $\nu'(y) = O(\nu(x) + \log |x|)$.

Let $\mathbb{A}'$ be an algorithm that, given x', decides $x' \in P'$ in time $2^{O(\nu'(x'))} \cdot |x'|^{O(1)}$.

Now we define the following algorithm $\mathbb{B}$ that, given an instance x, simulates $\mathbb{A}$ and answers the oracle queries with instance x' by simulating $\mathbb{A}'$ on x'. By (R1), for each oracle query with instance x' we have $|x'| = O(2^{\nu(x)} \cdot |x|^{O(1)})$. In addition, (R2) implies that $\nu'(x') = O(\nu(x) + \log|x|)$. It is clear that $\mathbb{B}$ decides P, and its running time on input x is bounded by

$$\begin{aligned} & O(2^{\nu(x)} \cdot |x|^{O(1)}) \cdot 2^{O(\nu'(x'))} \cdot |x'|^{O(1)} \\ = \; & O(2^{\nu(x)} \cdot |x|^{O(1)}) \cdot 2^{O(\nu(x)+\log|x|)} \cdot O(2^{O(\nu(x))} \cdot |x|^{O(1)}) \\ = \; & 2^{O(\nu(x))} \cdot |x|^{O(1)}. \qquad \square \end{aligned}$$

The following example introduces another problem that will turn out to be complete for the class EPT under serf reductions.

Example 8. Consider the halting problem for alternating Turing machines with a binary alphabet parameterized by the amount of space a computation uses (halting problems parameterized by space are referred to as “compact” halting problems in the parameterized complexity literature):

p-Compact-Bin-ATM-Halt
Instance: An alternating Turing machine M with binary alphabet, $k \in \mathbb{N}$.
Parameter: k.
Problem: Decide whether M accepts the empty input using at most k tape cells.

It is easy to see that p-Compact-Bin-ATM-Halt $\in$ EPT; just observe that, for a given instance (M, k), there are at most $N = 2^{O(k)} \cdot |M|^{O(1)}$ many relevant configurations. So we can first compute the *configuration graph* of M of size N and then test the accepting condition in time polynomial in N by computing the *alternating reachability problem* on that graph.

Parameterized complexity. As mentioned in the introduction, in parameterized complexity we are dealing with the same type of problems as in exponential complexity, namely, parameterized problems (Q, κ), where $Q \subseteq \Sigma^*$ and $\kappa : \Sigma^* \to \mathbb{N}$ is polynomial time computable. However, we study the problems from a different perspective; in parameterized complexity theory, we usually assume the parameter to be small, whereas in the exponential theory, the parameter is supposed to be a size measure and hence close to the size of the instance.

Definition 9. *A parameterized problem* (Q, κ) *is* fixed-parameter tractable *if there is an algorithm* $\mathbb{A}$ *and a computable function* f *such that for every instance* x $\mathbb{A}$ *decides if* $x \in Q$ *in time*

$$f(\kappa(x)) \cdot |x|^{O(1)}.$$

FPT *denotes the class of all fixed-parameter tractable problems.*

Observe that

$$\mathrm{EPT} \subseteq \mathrm{FPT}.$$

Example 10. The following parameterized vertex cover problem is maybe the best-studied problem in parameterized complexity theory.

p-Vertex-Cover
Instance: A graph G and a nonnegative integer k.
Parameter: k.
Problem: Decide if G has a vertex cover of cardinality k.

It is easy to see that p-Vertex-Cover is fixed-parameter tractable. Actually, the problem is even in EPT.

Example 11. The problem *s-var*-Sat is in EPT and hence fixed-parameter tractable.

Definition 12. *Let (Q, κ) and (Q', κ') be two parameterized problems over the alphabets Σ and Σ', respectively.*

(1) *A* (many-one) FPT reduction *from (Q, κ) to (Q', κ') is a mapping $R : \Sigma^* \to (\Sigma')^*$ such that:*
   (a) *for all $x \in \Sigma^*$ we have $(x \in Q \iff R(x) \in Q')$;*
   (b) *for all $x \in \Sigma^*$, the image $R(x)$ is computable in time*
$$f(\kappa(x)) \cdot |x|^{O(1)}$$
   for a computable $f : \mathbb{N} \to \mathbb{N}$;
   (c) *there is a computable function $g : \mathbb{N} \to \mathbb{N}$ such that $\kappa'(R(x)) \leq g(\kappa(x))$ for all $x \in \Sigma^*$.*

(2) *An* FPT Turing reduction *from (Q, κ) to (Q', κ') is an algorithm $\mathbb{A}$ with an oracle to Q' such that there are computable functions $f, g : \mathbb{N} \to \mathbb{N}$ with the following:*
   (a) *Given an instance $x \in \Sigma^*$, the algorithm $\mathbb{A}$ decides if $x \in Q$ in time*
$$f(\kappa(x)) \cdot |x|^{O(1)}.$$
   (b) *For all oracle queries "$y \in Q'$?" posed by $\mathbb{A}$ on input $x \in \Sigma^*$ it holds that $\kappa'(y) \leq g(\kappa(x))$.*

We write $(Q, \kappa) \leq^{\text{FPT}} (Q', \kappa')$ (or $(Q, \kappa) \leq^{\text{FPT-T}} (Q', \kappa')$) if there is an FPT reduction (FPT Turing reduction, respectively) from (Q, κ) to (Q', κ'), and we write $(Q, \kappa) \equiv^{\text{FPT}} (Q', \kappa')$ (and similarly $(Q, \kappa) \equiv^{\text{FPT-T}} (Q', \kappa')$) if $(Q, \kappa) \leq^{\text{FPT}} (Q', \kappa')$ and $(Q', \kappa') \leq^{\text{FPT}} (Q, \kappa)$. It is easy to see that a many-one FPT reduction is also an FPT Turing reduction, and FPT is closed under FPT Turing reductions.

Most parameterized problems that are studied in parameterized complexity theory have the property that, for every fixed-parameter value, the instances with this value are decidable in polynomial time (albeit by an algorithm whose running time is bounded by a polynomial that may depend on the parameter k). Essentially, XP is the class of all such problems but with a uniformity condition added.

Definition 13. *A parameterized problem (Q, κ) is in* XP *if there is an algorithm $\mathbb{A}$ and a computable function $f : \mathbb{N} \to \mathbb{N}$ such that $\mathbb{A}$ decides Q, and the running time of $\mathbb{A}$ on input x is bounded by*
$$O(|x|^{f(\kappa(x))}).$$

Clearly, XP is closed under FPT and FPT Turing reductions.

Example 14. The *parameterized compact halting problem for alternating Turing machines* is the following parameterized problem.

p-Compact-ATM-Halt
Instance: An *alternating* Turing machine M with *arbitrary* alphabet, $k \in \mathbb{N}$.
Parameter: k.
Problem: Decide whether M accepts the empty input using space k.

It has been proved by Demri, Laroussinie, and Schnoebelen [8] that p-Compact-ATM-Halt is complete for XP under FPT reductions.

3. The miniaturization mapping.

Definition 15. *Let (P, ν) be a parameterized problem over the alphabet Σ. We define the* miniaturization $\mathcal{M}(P, \nu)$ *of* (P, ν) *as the following parameterized problem:*

$\mathcal{M}(P, \nu)$
Instance: $x \in \Sigma^*$ and $m \in \mathbb{N}$ in *unary.*
Parameter: $\lceil \frac{\nu(x)}{\log m} \rceil$ if $m \geq 2$ and $\nu(x)$ otherwise.
Problem: Decide whether $x \in P$.

In other words, $\mathcal{M}(P, \nu)$ is the parameterization of the classical problem

Instance: $x \in \Sigma^*$ and $m \in \mathbb{N}$ in unary.
Problem: Decide whether $x \in P$.

with $\kappa(x, m) = \lceil \nu(x)/ \log m \rceil$.

The ideas underlying the following result go back to [4, 9]. The theorem also follows from Theorem 19 below, but nevertheless we find it worthwhile to state and prove it separately first.

Theorem 16. *Let (P, ν) be a parameterized problem. Then* $(P, \nu) \in$ SUBEPT $\iff$ $\mathcal{M}(P, \nu) \in$ FPT.

Proof. Let Σ be the alphabet of P. Assume (P, ν) is in SUBEPT. By Lemma 2, there is an algorithm $\mathbb{A}$ expecting inputs from $\Sigma^* \times \mathbb{N}$ and a computable function f such that, for all $(x, \ell) \in \Sigma^* \times \mathbb{N}$, the algorithm $\mathbb{A}$ decides if $x \in P$ in time $f(\ell) \cdot 2^{\nu(x)/\ell} \cdot |x|^{O(1)}$.

Let $\mathbb{B}$ be the following algorithm: Given an instance $(x, m) \in \Sigma^* \times \mathbb{N}$, first $\mathbb{B}$ computes the parameter

$$k := \begin{cases} \left\lceil \frac{\nu(x)}{\log m} \right\rceil & \text{if } m \geq 2, \\ \nu(x) & \text{otherwise.} \end{cases}$$

It is easy to verify that

$$2^{\nu(x)/k} \leq 2m. \tag{3.1}$$

Then $\mathbb{B}$ simulates $\mathbb{A}$ on (x, k). The time taken by the simulation is bounded by

$$\begin{aligned} f(k) \cdot 2^{\nu(x)/k} \cdot |x|^{O(1)} &\leq f(k) \cdot 2m \cdot |x|^{O(1)} && \text{(by (3.1))} \\ &\leq 2f(k) \cdot (|x| + m)^{O(1)}. \end{aligned}$$

Therefore $\mathcal{M}(P, \nu)$ is fixed-parameter tractable.

For the converse direction, assume there is an algorithm $\mathbb{A}$ and a computable function $f : \mathbb{N} \to \mathbb{N}$ that for every (x, m), with $m \geq 2$, decides if (x, m) is a "yes" instance of $\mathcal{M}(P, \nu)$ in time

$$f\left(\left\lceil \frac{\nu(x)}{\log m} \right\rceil \right) \cdot (|x| + m)^{O(1)}.$$

Without loss of generality we assume that f is nondecreasing. Now let $\mathbb{B}$ be the algorithm that, for any given $(x, \ell) \in \Sigma^* \times \mathbb{N}$, first computes

$$m := \left\lceil 2^{\nu(x)/\ell} \right\rceil,$$

which can be done in time $2^{O(\nu(x)/\ell)} \cdot |x|^{O(1)}$. Note $m \geq 2$. Then $\mathbb{B}$ simulates $\mathbb{A}$ on (x, m), which requires time

$$f\left(\left\lceil \frac{\nu(x)}{\log m} \right\rceil\right) \cdot (|x| + m)^{O(1)} \leq f(\ell) \cdot 2^{O(\nu(x)/\ell)} \cdot |x|^{O(1)}.$$

So again by Lemma 2, (P, ν) is in SUBEPT. □

As the following two examples show, there are a number of problems that have natural parameterized problems as their miniaturizations. The *Hamming weight* of a finite Boolean-valued function is the number of arguments that are mapped to TRUE.

Example 17. The miniaturization of *s-var*-CIRCUIT-SAT is FPT-equivalent to the following *parameterized weighted circuit satisfiability problem*:

p-W-CIRCUIT-SAT	
Instance:	A Boolean circuit C and a $k \in \mathbb{N}$.
Parameter:	k.
Problem:	Decide whether C has a satisfying assignment of Hamming weight k.

Essentially, this result goes back to Abrahamson, Downey, and Fellows [1] (see [14] for a proof). It derives its significance from the fact that p-W-CIRCUIT-SAT is complete for the important parameterized complexity class W[P].

Example 18. The miniaturization of p-COMPACT-BIN-ATM-HALT (see Example 8) is FPT-equivalent to the parameterized problem p-COMPACT-ATM-HALT (see Example 14).

Proof. First it is easy to see the miniaturization of p-COMPACT-BIN-ATM-HALT is FPT-equivalent to the problem:

p-$k \cdot \log n$-COMPACT-BIN-ATM-HALT	
Instance:	An alternating Turing machine M with binary alphabet, $n \in \mathbb{N}$ in *unary*, $k \in \mathbb{N}$.
Parameter:	k.
Problem:	Decide whether M accepts the empty input using space $k \cdot \lceil \log n \rceil$.

Therefore it suffices to show that p-COMPACT-ATM-HALT $\equiv^{\text{FPT}}$ p-$k \cdot \log n$-COMPACT-BIN-ATM-HALT.

p-COMPACT-ATM-HALT $\leq^{\text{FPT}}$ p-$k \cdot \log n$-COMPACT-BIN-ATM-HALT: Given an alternating machine M of arbitrary alphabet, it is not very hard, albeit tedious, to construct another alternating machine M_{bin} with a binary alphabet by encoding each symbol in M by a binary string of length $\lceil \log |M| \rceil$. Thus M accepts the empty input using space k if and only if M_{bin} accepts the empty input using space $k \cdot \lceil \log |M| \rceil$. So

$$(M, k) \mapsto (M_{\text{bin}}, |M|, k)$$

is an appropriate FPT reduction.

p-$k \cdot \log n$-COMPACT-BIN-ATM-HALT $\leq^{\text{FPT}}$ p-COMPACT-ATM-HALT: For an alternating machine M with a binary alphabet and a natural number n, we can construct an alternating machine $M_{\lceil \log n \rceil}$ whose alphabet is of size n, each corresponding to a $\lceil \log n \rceil$-bit binary. Thus $M_{\lceil \log n \rceil}$ can simulate a computation of M which uses space $k \cdot \lceil \log n \rceil$ by a computation using space k. Hence

$$(M, n, k) \mapsto (M_{\lceil \log n \rceil}, k)$$

gives the required FPT reduction. □

The previous example can be generalized to the halting problems for *deterministic* and *nondeterministic* Turing machines parameterized by space. However for the moment we do not have a similar exact correspondence between "time" halting problems.

As we have seen in Theorem 16, miniaturization maps subexponential solvability of a problem exactly into the fixed-parameter tractability of its miniaturization. Indeed it can be viewed as a consequence of the following theorem.

THEOREM 19. *Let (P, ν) and (P', ν') be parameterized problems. Then*

(1) $(P, \nu) \leq^{\text{serf}} (P', \nu') \iff \mathcal{M}(P, \nu) \leq^{\text{FPT}} \mathcal{M}(P', \nu')$*;*

(2) $(P, \nu) \leq^{\text{serf-T}} (P', \nu') \iff \mathcal{M}(P, \nu) \leq^{\text{FPT-T}} \mathcal{M}(P', \nu')$*.*

Proof. We show (2), and the easier (1) is left to the reader. Let Σ and Σ' be the alphabets of P and P', respectively.

For the forward direction, let $\mathbb{A}$ be a serf Turing reduction from (P, ν) to (P', ν') such that for every input $(x, \ell) \in \Sigma^* \times \mathbb{N}$ the running time of $\mathbb{A}$ is bounded by $f(\ell) \cdot 2^{\nu(x)/\ell} \cdot |x|^{O(1)}$ and for any oracle query "$x' \in P'$?" posed by $\mathbb{A}$ it holds that

$$\nu'(x') \leq g(\ell) \cdot (\nu(x) + \log |x|) \tag{3.2}$$

for some computable function $g : \mathbb{N} \to \mathbb{N}$. Without loss of generality we assume that f is nondecreasing.

Let $\mathbb{B}$ be an algorithm that, given an instance (x, m) of $\mathcal{M}(P, \kappa)$, simulates $\mathbb{A}$ on (x, k), where

$$k := \begin{cases} \left\lceil \frac{\nu(x)}{\log m} \right\rceil & \text{if } m \geq 2, \\ \nu(x) & \text{otherwise} \end{cases}$$

is the parameter of (x, m) and replaces each oracle query "$x' \in P'$?" posed by $\mathbb{A}$ by "$(x', m') \in \mathcal{M}(P', \nu')$?", where $m' := \max\{|x'|, m, 2\}$.

The overall running time of $\mathbb{B}$ is bounded by

$$f(k) \cdot 2^{\nu(x)/k} \cdot |x|^{O(1)} \leq f(k) \cdot 2m \cdot |x|^{O(1)}.$$

Furthermore, for each oracle query "$(x', m') \in \mathcal{M}(P', \nu')$?" posed by $\mathbb{B}$, we have $\nu'(x') \leq g(k) \cdot (\nu(x) + \log |x|)$ by (3.2). Since $m' \geq 2$, the parameter of (x', m') is $\lceil \frac{\nu'(x')}{\log m'} \rceil$.

- If $m \geq 2$, then $k = \lceil \frac{\nu(x)}{\log m} \rceil$, and we have

$$\begin{aligned} \left\lceil \frac{\nu'(x')}{\log m'} \right\rceil &\leq g(k) \cdot \left\lceil \frac{\nu(x) + \log |x|}{\log m'} \right\rceil \\ &\leq g(k) \cdot \left(\left\lceil \frac{\nu(x)}{\log m} \right\rceil + 1 \right) \qquad (\text{by } m' \geq |x| \text{ and } m' \geq m) \\ &= g(k) \cdot (k + 1). \end{aligned}$$

- Otherwise $m = 1$ and $k = \nu(x)$. It follows that

$$\begin{aligned}\left\lceil \frac{\nu'(x')}{\log m'} \right\rceil &\le g(k) \cdot \left\lceil \frac{\nu(x) + \log |x|}{\log m'} \right\rceil \\ &\le g(k) \cdot \left(\left\lceil \frac{\nu(x)}{\log m'} \right\rceil + 1 \right) && \text{(by } m' \ge |x|\text{)} \\ &\le g(k) \cdot (k+1) && \text{(since } m' \ge 2 \text{ and } k = \nu(x)\text{).}\end{aligned}$$

Thus $\mathbb{B}$ is an FPT Turing reduction from $\mathcal{M}(P, \nu)$ to $\mathcal{M}(P', \nu')$.

For the backward direction, let $\mathbb{A}$ be an algorithm with an oracle to $\mathcal{M}(P', \nu')$ such that there are nondecreasing computable functions $f, g : \mathbb{N} \to \mathbb{N}$ and a constant $d \in \mathbb{N}$ with the following:

(F1) Given an instance $(x, m) \in \Sigma^* \times \mathbb{N}$, with $m \ge 2$, the algorithm $\mathbb{A}$ decides if $(x, m) \in \mathcal{M}(P, \nu)$ in time

$$f\left(\left\lceil \frac{\nu(x)}{\log m} \right\rceil \right) \cdot (|x| + m)^d.$$

(F2) For all oracle queries "$(x', m') \in \mathcal{M}(P', \nu')$?" posed by $\mathbb{A}$ on input (x, m), with $m \ge 2$, we have

$$\left\lceil \frac{\nu'(x')}{\log m'} \right\rceil \le g\left(\left\lceil \frac{\nu(x)}{\log m} \right\rceil \right),$$

if $m' \ge 2$, and

$$\nu'(x') \le g\left(\left\lceil \frac{\nu(x)}{\log m} \right\rceil \right),$$

if $m' = 1$.

To show $(P, \nu) \le^{\text{serf-T}} (P', \nu')$, let $\mathbb{B}$ be an algorithm with an oracle to (P', ν') such that, given a pair $(x, \ell) \in \Sigma^* \times \mathbb{N}$, the algorithm $\mathbb{B}$ simulates $\mathbb{A}$ on (x, m), where

$$m := \max\left\{ |x|, \left\lceil 2^{\nu(x)/(d \cdot \ell)} \right\rceil \right\},$$

and replaces oracle queries "$(x', m') \in \mathcal{M}(P', \nu')$?" posed by $\mathbb{A}$ by "$x' \in P'$?".

Observe that $m \ge 2$ and

$$\frac{\nu(x)}{\log m} \le d \cdot \ell. \tag{3.3}$$

So by (F1) the running time of $\mathbb{A}$ on (x, m), and hence $\mathbb{B}$ on (x, ℓ), is bounded by

$$f\left(\left\lceil \frac{\nu(x)}{\log m} \right\rceil \right) \cdot (|x| + m)^d \le f(d \cdot \ell) \cdot 2^d \cdot 2^{\nu(x)/\ell} \cdot |x|^d.$$

Here we assume without loss of generality that $|x| > 0$. Note that it follows that, for any oracle query "$(x', m') \in \mathcal{M}(P', \nu')$?" posed by $\mathbb{A}$, we have

$$m' \le f(d \cdot \ell) \cdot 2^d \cdot 2^{\nu(x)/\ell} \cdot |x|^d, \tag{3.4}$$

since it is represented in unary.

Now for any oracle query "$x' \in P'$?" posed by $\mathbb{B}$, which comes from the simulation of $\mathbb{A}$ over an oracle query "$(x', m') \in \mathcal{M}(P', \nu')$?", if $m' \geq 2$, then (F2) implies

$$\left\lceil \frac{\nu'(x')}{\log m'} \right\rceil \leq g\left(\left\lceil \frac{\nu(x)}{\log m} \right\rceil \right).$$

Therefore by (3.3) and (3.4)

$$\nu'(x') \leq g(d \cdot \ell) \cdot \log(f(d \cdot \ell) \cdot 2^d \cdot 2^{\nu(x)/\ell} \cdot |x|^d) \leq h(\ell) \cdot (\nu(x) + \log|x|)$$

for some suitable computable function h. Otherwise $m' = 1$. Again by (F2), we have

$$\nu'(x') \leq g\left(\left\lceil \frac{\nu(x)}{\log m} \right\rceil \right).$$

It follows that

$$\nu'(x') \leq g(d \cdot \ell) \leq g(d \cdot \ell) \cdot (\nu(x) + \log|x|).$$

So $\mathbb{B}$ is a serf Turing reduction from (P, ν) to (P', ν'). □

The main results of this and the following section can most elegantly be formulated in the language of *degrees* from classical recursion theory. Suppose we have some reducibility relation $\leq$ on parameterized problems, for example, $\leq^{\text{FPT}}$. In general, $\leq$ only needs to be a reflexive and transitive relation. Let us denote the corresponding equivalence relation by $\equiv$. Then the $\leq$-*degree* of a problem (Q, ν), denoted by $[\![(Q, \nu)]\!]^{\leq}$, is the $\equiv$-equivalence class of (Q, ν). For example, the $\leq^{\text{FPT}}$-degree of p-COMPACT-ATM-HALT is the class of all XP-complete problems. The class of all $\leq$-degrees is denoted by $\mathbf{D}_{\leq}$, and for a class C of parameterized problems that is downward closed under $\leq$, the class of all degrees in C is denoted by $\mathbf{C}_{\leq}$. The reduction $\leq$ induces a partial order on $\mathbf{D}_{\leq}$. If $\leq = \leq^{\text{FPT}}$, then to simplify the notation we speak of FPT-degrees instead of $\leq^{\text{FPT}}$-degrees and write $[\![(Q, \nu)]\!]^{\text{FPT}}$, $\mathbf{D}_{\text{FPT}}$, etc. The same notational convention applies to reductions $\leq^{\text{FPT-T}}$, $\leq^{\text{serf}}$, and $\leq^{\text{serf-T}}$.

Note that, by Theorem 19 (1), the miniaturization mapping induces a well-defined mapping $\boldsymbol{\mathcal{M}} : \mathbf{D}_{\text{serf}} \to \mathbf{D}_{\text{FPT}}$, defined by $\boldsymbol{\mathcal{M}}\big([\![(P, \nu)]\!]^{\text{serf}}\big) := \big[\!\big[\mathcal{M}(P, \nu)\big]\!\big]^{\text{FPT}}$, on the serf-degrees. By Theorem 19(2), it also induces a mapping on the serf Turing degrees. The main results of this section can be summarized in the following theorem.

THEOREM 20 (embedding theorem). *The miniaturization mapping induces an embedding of the partially ordered set* $(\mathbf{D}_{\text{serf}}, \leq^{\text{serf}})$ *into the partially ordered set* $(\mathbf{D}_{\text{FPT}}, \leq^{\text{FPT}})$ *and also an embedding of the partially ordered set* $(\mathbf{D}_{\text{serf-T}}, \leq^{\text{serf-T}})$ *into the partially ordered set* $(\mathbf{D}_{\text{FPT-T}}, \leq^{\text{FPT-T}})$.

4. An isomorphism between EPT and XP.

LEMMA 21. *Let* $(Q, \kappa) \in \text{XP}$. *Then there exists a problem* $(P, \nu) \in \text{EPT}$ *such that* $(Q, \kappa) \equiv^{\text{FPT}} \mathcal{M}(P, \nu)$.

Proof. Let Σ be the alphabet of Q. Without loss of generality we may assume that $Q \neq \emptyset$ and $Q \neq \Sigma^*$. In a first step we construct a problem (Q', κ') that is FPT-equivalent to (Q, κ) and decidable in time $O(|x|^{\sqrt{\kappa'(x)}})$.

Suppose that $x \in Q$ is decidable in time $O\left(|x|^{f(\kappa(x))}\right)$, where without loss of generality f is increasing and time constructible. Let (Q', κ') be the following parameterized problem:

Input: $x \in \Sigma^*$, and $\ell \in \mathbb{N}$ in unary such that $\ell \geq f(\kappa(x))^2$.
Parameter: ℓ.
Problem: Decide whether $x \in Q$.

It is easy to see that indeed $(Q', \kappa') \equiv^{\text{FPT}} (Q, \kappa)$ and that Q' is decidable in time $O(|x|^{\sqrt{\kappa'(x)}})$.

In the second step, we construct the desired problem (P, ν). Let Σ' be the alphabet of Q'. We let (P, ν) be the following problem:

Input: $x \in (\Sigma')^*$.
Parameter: $\kappa'(x) \cdot \lceil \log |x| \rceil$ if $|x| \geq 2$ and $\kappa'(x)$ otherwise.
Problem: Decide whether $x \in Q'$.

Then $P = Q'$; that is, (P, ν) is just a reparameterization of (Q', κ'). Recall that Q' is decidable in time

$$O\left(|x|^{\sqrt{\kappa'(x)}}\right) = O\left(2^{\sqrt{\kappa'(x)} \cdot \log |x|}\right) \leq O\left(2^{\nu(x)}\right).$$

It follows that $(P, \nu) \in \text{EPT}$. Now we claim that $\mathcal{M}(P, \nu) \equiv^{\text{FPT}} (Q', \kappa')$.

Let $x_+ \in Q'$ and $x_- \in (\Sigma')^* \setminus Q'$. To prove that $\mathcal{M}(P, \nu) \leq^{\text{FPT}} (Q', \kappa')$, we define a reduction R by letting

$$R(x, m) := \begin{cases} x_+ & \text{if } m \geq |x|^{\sqrt{\kappa'(x)}} \text{ and } x \in Q', \\ x_- & \text{if } m \geq |x|^{\sqrt{\kappa'(x)}} \text{ and } x \notin Q', \\ x & \text{otherwise.} \end{cases}$$

Then clearly for all $(x, m) \in \Sigma^* \times \mathbb{N}$ we have $(x, m) \in \mathcal{M}(P, \nu) \iff R(x, m) \in Q'$. Moreover, $R(x, m)$ is computable in polynomial time, because $x \in Q'$ is decidable in time $O(|x|^{\sqrt{\kappa'(x)}})$, which is $O(m)$ if $m \geq |x|^{\sqrt{\kappa'(x)}}$.

It remains to prove that the parameter $\kappa'(R(x, m))$ of the image is effectively bounded in terms of the parameter

$$k := \begin{cases} \lceil \nu(x)/\log m \rceil & \text{if } m \geq 2, \\ \nu(x) & \text{otherwise} \end{cases}$$

of the argument. Let (x, m) be an instance of $\mathcal{M}(P, \nu)$. If $m \geq |x|^{\sqrt{\kappa'(x)}}$, then $\kappa(R(x, m)) \leq \max\{\kappa(x_+), \kappa(x_-)\}$, which is a constant. So let us assume that $m < |x|^{\sqrt{\kappa'(x)}}$. Then $\kappa'(R(x, m)) = \kappa'(x)$.

- If $|x| \geq 2$, then $\log m < \sqrt{\kappa'(x)} \cdot \lceil \log |x| \rceil = \nu(x)/\sqrt{\kappa'(x)}$, because $\nu(x) = \kappa'(x) \cdot \lceil \log |x| \rceil$. Thus in the case $m \geq 2$ we have $\kappa'(x) = \kappa'(x) \cdot \log m / \log m < \sqrt{\kappa'(x)} \cdot \nu(x)/\log m$ and therefore $\kappa'(x) \leq (\nu(x)/\log m)^2 \leq k^2$. Otherwise, if $m = 1$, we have $k = \nu(x) = \kappa'(x) \cdot \lceil \log |x| \rceil \geq \kappa'(x)$.
- If $|x| < 2$, then $\kappa'(x)$ is bounded by a constant, since there are only finitely many such x.

This shows that indeed R is an FPT reduction and proves $\mathcal{M}(P, \nu) \leq^{\text{FPT}} (Q', \kappa')$.

For the other direction, $(Q', \kappa') \leq^{\text{FPT}} \mathcal{M}(P, \nu)$, we define a reduction $R : (\Sigma')^* \to (\Sigma')^* \times \mathbb{N}$ by $R(x) = (x, 2|x| + 2))$. It is easy to see that $\nu(x) \leq \kappa'(x) \cdot \lceil \log(|x| + 2) \rceil$

for every $x \in (\Sigma')^*$. Thus

$$\left\lceil \frac{\nu(x)}{\log(2|x|+2))} \right\rceil \leq \left\lceil \frac{\kappa'(x) \cdot \lceil \log(|x|+2) \rceil}{\log(2|x|+2)} \right\rceil \leq \kappa'(x),$$

and R is an FPT reduction from (Q', κ') to $\mathcal{M}(P, \nu)$. □

Now we can establish a similar correspondence as Theorem 16 with respect to XP and EPT.

THEOREM 22. *Let (P, ν) be a parameterized problem. Then $(P, \nu) \in$ EPT $\iff$ $\mathcal{M}(P, \nu) \in$ XP.*

Proof. Let $(P, \nu) \in$ EPT; in other words, (P, ν) is decidable in time $2^{O(\nu(x))} \cdot |x|^O(1)$. Let (x, m) be an instance of $\mathcal{M}(P, \nu)$. If $m \geq 2$, then we let

$$k := \lceil \nu(x)/\log m \rceil .$$

Then the instance is decidable in time $2^{O(\nu(x))} \cdot |x|^{O(1)} \leq m^{O(\nu(x)/\log m)} \cdot |x|^{O(1)} \leq (m + |x|)^{O(k)}$. If $m = 1$, then the parameter k of (x, m) is $\nu(x)$. It follows that we can decide whether $(x, m) \in \mathcal{M}(P, \nu)$ in time $2^{O(\nu(x))} \cdot |x|^{O(1)} = 2^{O(k)} \cdot |x|^{O(1)}$. Hence, $\mathcal{M}(P, \nu)$ is in XP.

For another direction, assume $\mathcal{M}(P, \nu) \in$ XP. By Lemma 21, there is a parameterized problem $(P', \nu') \in$ EPT such that $\mathcal{M}(P, \nu) \equiv^{\text{FPT}} \mathcal{M}(P', \nu')$. Now Theorem 19 implies $(P, \nu) \equiv^{\text{serf}} (P', \nu')$, and by Proposition 7, $(P, \nu) \in$ EPT. □

COROLLARY 23. *Let (P, ν) be a parameterized problem. (P, ν) is* EPT*-complete (*EPT*-hard) under* serf *reductions if and only $\mathcal{M}(P, \nu)$ is* XP*-complete (*XP*-hard, respectively) under* FPT *reductions.*

Example 24. p-COMPACT-BIN-ATM-HALT is complete for EPT under serf reductions.

Proof. p-COMPACT-ATM-HALT is complete for XP under FPT reductions [8], and the miniaturization of p-COMPACT-ATM-HALT is FPT-equivalent to p-COMPACT-BIN-ATM-HALT (see Example 18). So Corollary 23 implies p-COMPACT-BIN-ATM-HALT is complete for EPT under serf reductions. □

Rephrasing the results of this section in the language of degrees introduced at the end of the previous section, we obtain the following theorem.

THEOREM 25 (isomorphism theorem). *The miniaturization mapping induces an isomorphism between $(\mathbf{EPT}_{\text{serf}}, \leq^{\text{serf}})$ and $(\mathbf{XP}_{\text{FPT}}, \leq^{\text{FPT}})$ and also an isomorphism between $(\mathbf{EPT}_{\text{serf-T}}, \leq^{\text{serf-T}})$ and $(\mathbf{XP}_{\text{FPT-T}}, \leq^{\text{FPT-T}})$.*

Outside EPT and XP. The following theorem shows that the isomorphism theorem cannot be extended from the degrees in EPT and XP to all degrees, because outside of XP the mapping induced by the miniaturization mapping is not onto.

THEOREM 26. *There is a parameterized problem (Q, κ) that is not* FPT-T*-equivalent to $\mathcal{M}(P, \nu)$ for any (P, ν).*

We need some preparation before we prove the theorem.

DEFINITION 27. *Let Q and Q' be two classical problems. An algorithm $\mathbb{A}$ with an oracle to Q' is a* 2-exptime Turing reduction *from Q to Q', if for any instance x of Q, $\mathbb{A}$ decides if $x \in Q$ in time*

$$2^{2^{|x|^{O(1)}}}.$$

2-exptime Turing reductions are slightly at odds with all of the usual reductions (including those introduced in this paper so far); namely, they are not *transitive.* However, they are closed under the composition with polynomial time Turing reductions. More precisely, we have the following.

LEMMA 28. *Let Q, Q', Q'' be classical problems. There is a 2-exptime Turing reduction from Q to Q'':*

- *if there is a 2-exptime Turing reduction from Q to Q' and a polynomial time Turing reduction from Q' to Q'';*
- *or if there is a polynomial time Turing reduction from Q to Q' and a 2-exptime Turing reduction from Q' to Q''.*

We omit the routine proof.

LEMMA 29. *There is a sequence of problems $(Q_i)_{i\in\mathbb{N}}$ such that*

- *$\big\{\{i\} \times Q_i \mid i \in \mathbb{N}\big\}$ is decidable, and*
- *for all $i \in \mathbb{N}$, Q_i is not 2-exptime Turing reducible to*

$$L_i := \big\{(j, x) \mid j \neq i \text{ and } x \in Q_j\big\}.$$

Proof. Fix an alphabet Σ. Let $\mathbb{A}_1, \mathbb{A}_2, \dots$ be an effective enumeration of all of the 2-exptime Turing reductions from problems over the alphabet Σ to problems that are subsets of $\mathbb{N} \times \Sigma^*$.

For $S \subseteq \mathbb{N} \times \Sigma^*$, $e \in \mathbb{N}$, and $x \in \Sigma^*$, we define

$$\mathbb{A}_e^S(x) := \begin{cases} 1 & \mathbb{A}_e \text{ accepts } x \text{ with an oracle to } S, \\ 0 & \text{otherwise} \end{cases}$$

and let

$$u(S, e, x) := \max\big\{|y| \;\big|\; \text{in the computation of } \mathbb{A}_e^S(x) \\ \text{there is an oracle query ``} y \in S\text{?''}\big\}.$$

Clearly, for a computable S the sets $\mathbb{A}_e^S(x)$ and $u(S, e, x)$ are both computable in e and x.

Note that, for given $S_1, S_2 \subseteq \mathbb{N} \times \Sigma^*$, $e \in \mathbb{N}$, and $x \in \Sigma^*$, if for all $y \in \mathbb{N} \times \Sigma^*$, with $|y| \leq u(S_1, e, x)$, $(y \in S_1 \iff y \in S_2)$, then the computation of $\mathbb{A}_e^{S_1}(x)$ exactly coincides with that of $\mathbb{A}_e^{S_2}(x)$; in particular, $\mathbb{A}_e^{S_1}(x) = \mathbb{A}_e^{S_2}(x)$.

Let $\langle\,\cdot\,,\,\cdot\,\rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be a reasonable bijective encoding function. We shall construct a computable sequence of pairwise distinct elements $(a_j)_{j\in\mathbb{N}} \in \Sigma^*$ such that, for $j = \langle i, e\rangle$, a_j witnesses the fact that

$$\mathbb{A}_e \text{ is not a reduction from } Q_i \text{ to } L_i.$$

Simultaneously, we will define a sequence $0 = \ell_0 < \ell_1 < \ell_2 \cdots$ of nonnegative integers and a sequence $\emptyset = P_0 \subseteq P_1 \subseteq P_2 \subseteq \cdots \subseteq \mathbb{N} \times \Sigma^*$ of sets such that for all $j \in \mathbb{N}$, with $j = \langle i, e\rangle$,

$$\ell_{j-1} < |(i, a_j)| \leq \ell_j, \tag{4.1}$$

$$\text{and} \quad |P_j \backslash P_{j-1}| \subseteq \{(i, a_j)\}. \tag{4.2}$$

Then we set, for each $i \in \mathbb{N}$,

$$Q_i := \Big\{a \in \Sigma^* \mid (i, a) \in \bigcup_{j\in\mathbb{N}} P_j\Big\}. \tag{4.3}$$

Recall that $\ell_0 = 0$ and $P_0 = \emptyset$. Now for $j \in \mathbb{N}$, with $j = \langle i, e\rangle$, assume ℓ_{j-1} and P_{j-1} are already defined. Let $a_j \in \Sigma^*$ be the minimal element in the lexicographical order such that

$$|(i, a_j)| > \ell_{j-1}$$

and that a_j is distinct from all $a_{j'}$ for $1 \le j' < j$. Now let

$$T_j := \big\{(i', a) \in P_{j-1} \mid i' \neq i\big\} = P_{j-1} \backslash \big(\{i\} \times \Sigma^*\big) \tag{4.4}$$

and

$$\ell_j := \max\big\{u(T_j, e, a_j),\ |(i, a_j)|\big\}. \tag{4.5}$$

(i) If $\mathbb{A}_e^{T_j}(a_j) = 0$, let $P_j := P_{j-1} \cup \{(i, a_j)\}$.
(ii) Otherwise, if $\mathbb{A}_e^{T_j}(a_j) = 1$, let $P_j := P_{j-1}$.

This finishes the construction.

Now we show that, for each $i, e \in \mathbb{N}$, $\mathbb{A}_e$ is not a reduction from Q_i to

$$\begin{aligned} L_i &= \big\{(i', a) \mid i' \neq i \text{ and } a \in Q_{i'}\big\} \\ &= \big\{(i', a) \mid i' \neq i, \text{ and there exists a } j \in \mathbb{N} \text{ such that } (i', a) \in P_j\big\}. \end{aligned} \tag{4.6}$$

Let $j := \langle i, e\rangle$. Note that it suffices to prove that it is *not* the case that

$$a_j \in Q_i \iff \mathbb{A}_e^{L_i}(a_j) = 1.$$

First observe that (4.4) and (4.6) imply $T_j \subseteq L_i$. Also by (4.1), (4.2), and (4.5), for any $(i', a') \in L_i \backslash T_j$, we have $|(i', a')| > u(T_j, e, a_j)$. It follows that

$$\mathbb{A}_e^{L_i}(a_j) = \mathbb{A}_e^{T_j}(a_j). \tag{4.7}$$

Recall during the construction that we have two cases for $\mathbb{A}_e^{T_j}(a_j)$:

(i) If $\mathbb{A}_e^{T_j}(a_j) = 0$, then $\mathbb{A}_e^{L_i}(a_j) = 0$ by (4.7). Also by our construction $(i, a_j) \in P_j$, and therefore $a_j \in Q_i$ by (4.3).
(ii) If $\mathbb{A}_e^{T_j}(a_j) = 1$, then $\mathbb{A}_e^{L_i}(a_j) = 1$. Hence $(i, a_j) \notin P_j \backslash P_{j-1}$. It follows that $(i, a_j) \notin P_{j'}$ for all $j' \in \mathbb{N}$ by (4.2). Consequently $a_j \notin Q_i$.

To see that $\{\{i\} \times Q_i \mid i \in \mathbb{N}\}$ is decidable, let $(i, a) \in \mathbb{N} \times \Sigma^*$ be an instance. Clearly

$$a \in Q_i \iff (i, a) \in \bigcup_{j \in \mathbb{N}} P_j.$$

We compute the minimal $j \in \mathbb{N}$ such that

$$|(i, a)| \le \ell_j$$

and then decide if (i, a) is in the finite set P_j. □

Proof of Theorem 26. Let $(Q_i)_{i \in \mathbb{N}}$ be as stated in Lemma 29. Also for each $e \in \mathbb{N}$, let ϕ_e denote the eth partially recursive function.

For any $e \in \mathbb{N}$ and $n \in \mathbb{N} \cup \{0\}$, let

$$C(e, n) := \begin{cases} k & k \text{ is maximum such that } \phi_e(1), \ldots, \phi_e(k) \text{ are defined,} \\ & \quad \text{together can be computed in at most } n \text{ steps,} \\ & \quad \text{and each is smaller than } n, \\ & \\ 1 & \text{if no such } k \text{ exists.} \end{cases}$$

Clearly for any fixed e, $C(e,n)$ can be computed in time polynomial in n, and $C(e,n) \leq n$. Moreover if $\phi_e(1)$ is defined, then

$$\phi_e(C(e,n)) \leq \max\{\phi_e(1), n\}. \tag{4.8}$$

Now let

$$Q := \big\{(e,a,k) \mid e \in \mathbb{N},\ a \in Q_e, \text{ and } k = C(e,|a|)\big\}.$$

Define the parameterization κ by $\kappa(e,a,k) := k$.

Suppose for contradiction that

$$(Q,\kappa) \equiv^{\text{FPT-T}} \mathcal{M}(P,\nu)$$

for a parameterized problem (P,ν) over some alphabet Σ'. Let

$$\mu(x,m) = \left\lceil \frac{\nu(x)}{\log m} \right\rceil$$

be the parameterization of $\mathcal{M}(P,\nu)$.

Let $\mathbb{A}$ be an FPT-T reduction from (Q,κ) to $\mathcal{M}(P,\nu)$, and let $f : \mathbb{N} \to \mathbb{N}$ be a computable function such that, for all instances x, the algorithm $\mathbb{A}$ decides if $x \in Q$ in time

$$f(\kappa(x)) \cdot |x|^{O(1)}.$$

Let $e \in \mathbb{N}$ be such that $\phi_e = f$. We leave e fixed for the rest of the proof.

Claim 1. *There is a polynomial time Turing reduction from Q_e to $\mathcal{M}(P,\nu)$.*

Proof of the claim. Let $\mathbb{B}$ be the following Turing reduction from Q_e to Mini(P,ν): Given an instance $a \in \Sigma^*$, the algorithm $\mathbb{B}$ computes in polynomial time $k := C(e,|a|)$. Then it simulates the FPT-T reduction $\mathbb{A}$ on (e,a,k). Since $a \in Q_e \iff (e,a,k) \in Q$, this algorithm $\mathbb{B}$ correctly decides if $a \in Q_e$.

Moreover (4.8) implies that

$$\phi_e(\kappa(e,a,k)) = \phi_e(k) = O(|a|)$$

for ϕ_e is total. Hence the running time of $\mathbb{A}$ (and hence of $\mathbb{B}$) on (e,a,k) is

$$\begin{aligned}\phi_e(\kappa(e,a,k)) \cdot |(e,a,k)|^{O(1)} &= O(|a|) \cdot |(e,a,k)|^{O(1)} \\ &= O(|a|) \cdot |a|^{O(1)} \\ &= O(|a|^{O(1)}).\end{aligned}$$

The second equality follows from the fact that $k = C(e,|a|) \leq |a|$, and e is a constant. Thus $\mathbb{B}$ is a desired polynomial time Turing reduction, which proves Claim 1. □

Since ν is polynomial time computable, without loss of generality we assume

$$\nu(x) \leq 2^{|x|} \tag{4.9}$$

for any $x \in (\Sigma')^*$. Let

$$P' := \left\{(x, 2^{2^{|x|}}) \mid x \in P\right\}.[3]$$

[3] Here $2^{2^{|x|}}$ is represented in unary.

It follows that for all $(x, m) \in P'$

$$m = 2^{2^{|x|}}. \tag{4.10}$$

CLAIM 2. *$\mathcal{M}(P, \nu)$ is 2-exptime Turing reducible to P'.*

Proof of the claim. Given an instance $(x, m) \in (\Sigma')^* \times \mathbb{N}$, we have

$$(x, m) \in \mathcal{M}(P, \nu) \iff x \in P \iff (x, 2^{2^{|x|}}) \in P'.$$

This can be easily turned into a 2-exptime Turing reduction, and thus Claim 2 is proved. □

Note that for all $(x, m) \in P'$ we have

$$\begin{aligned} \mu(x, m) &= \left\lceil \frac{\nu(x)}{\log m} \right\rceil \\ &\le \left\lceil \frac{2^{|x|}}{\log 2^{2^{|x|}}} \right\rceil \qquad \text{(by (4.9) and (4.10))} \\ &= 1. \end{aligned} \tag{4.11}$$

CLAIM 3. *There is a polynomial time Turing reduction $\mathbb{B}$ from P' to Q. Moreover the set*

$$\big\{(e, a, k) \in Q \mid \textit{for some } (x, m) \in (\Sigma')^* \times \mathbb{N}, \textit{ there is} \\ \textit{an oracle query "}(e, a, k) \in Q?\textit{" in the computation of } \mathbb{B} \textit{ on } (x, m)\big\}$$

is finite.

Proof of the claim. Recall that we assume $\mathcal{M}(P, \nu) \equiv^{\text{FPT-T}} (Q, \kappa)$. So there is an algorithm $\mathbb{A}'$ with an oracle to (Q, κ) and a computable function $g : \mathbb{N} \to \mathbb{N}$ such that, given an instance $(x, m) \in (\Sigma')^* \times \mathbb{N}$:

(E1) $\mathbb{A}'$ decides if $(x, m) \in \mathcal{M}(P, \nu)$ in time

$$g(\mu(x, m)) \cdot |(x, m)|^{O(1)};$$

(E2) for each oracle query "$(e', a', k') \in Q$?" posed by $\mathbb{A}'$,

$$\kappa(e', a', k') = k' \le g'(\mu(x, m))$$

for a computable function $g' : \mathbb{N} \to \mathbb{N}$. For simplicity, we assume $g' = g$.

Given an instance (x, m) of P', the algorithm $\mathbb{B}$ first checks in polynomial time if $m \neq 2^{2^{|x|}}$ or $\mu(x, m) > 1$. If so, then (x, m) is a "no" instance by (4.10) and (4.11). Otherwise $m = 2^{2^{|x|}} \ge |x|$, and $\mu(x, m) = 1$. It follows that

$$(x, m) \in P' \iff x \in P \iff (x, m) \in \mathcal{M}(P, \nu).$$

Therefore $\mathbb{B}$ decides if $(x, m) \in P'$ by simulating $\mathbb{A}'$ on (x, m), which by (E1) requires time

$$g(\mu(x, m)) \cdot |(x, m)|^{O(1)} = g(1) \cdot |(x, m)|^{O(1)},$$

and is therefore polynomial in $|(x, m)|$.

Now for any oracle query "$(e', a', k') \in Q$?" that occurs in the simulation of $\mathbb{A}'$ on (x, m), by (E2) we have

$$k' = \kappa(e', a', k') \leq g(\mu(x, m)) = g(1).$$

If $(e', a', k') \in Q$ and $e' = e$, then $k' = C(e, |a'|)$ is the maximum such that $\varphi_e(1), \dots, \varphi_e(k')$ can be computed in at most $|a'|$ steps and all bounded by $|a'|$. As k' is bounded by the fixed constant $g(1)$ independent of x and m, there are only finitely many such a' with $(e, a', k') \in Q$ for all possible instances (x, m); otherwise, ϕ_e is not total. Thus Claim 3 is proved. □

CLAIM 4. *There is a polynomial time Turing reduction from P' to*

$$L_e = \bigl\{(e', a) \mid e' \neq e \text{ and } a \in Q_{e'}\bigr\}.$$

Proof of the claim. By Claim 3 it suffices to give a polynomial time Turing reduction from

$$\{(e', a, k) \in Q \mid e' \neq e\}$$

to L_e: For any instance (e', a, k), if $k \neq C(e', |a|)$ or $e' = e$, then it is a no instance. Otherwise

$$(e', a, k) \in \bigl\{(e', a, k) \in Q \mid e' \neq e\bigr\} \iff (e', a) \in L_e.$$

This proves Claim 4. □

Combining Claims 1, 2, and 4, we have a 2-exptime Turing reduction from Q_e to L_e by Lemma 28, which contradicts our construction. □

5. The S-hierarchy and the W-hierarchy. The purpose of this last section of the paper is to gather further evidence that the miniaturization mapping is not only an abstract isomorphism between exponential and parameterized complexity theory but actually establishes a relation between interesting and relevant complexity classes on both sides. Specifically, we shall lay out a relation between a natural hierarchy of the exponential theory and the W-hierarchy of parameterized complexity theory. The basic ideas underlying this section go back to Abrahamson, Downey, and Fellows [1] and have been refined in [5, 6, 13].

The W-hierarchy is defined in terms of *weighted satisfiability problems* for classes Γ of Boolean formulas or circuits:

p-WSAT(Γ)
Instance: $\gamma \in \Gamma$ and $k \in \mathbb{N}$.
Parameter: k.
Problem: Decide whether γ has a satisfying assignment of Hamming weight k.

We denote the class of Boolean circuits by CIRC and the class of formulas by FORM. Recall that FORM is viewed as a subclass of CIRC. Note that p-WSAT(CIRC) is exactly the parameterized problem p-W-CIRCUIT-SAT introduced in Example 17.

For $t \geq 0$ and $d \geq 1$ we inductively define the following classes $\Gamma_{t,d}$ and $\Delta_{t,d}$ of Boolean formulas:

$$\begin{aligned}
\Gamma_{0,d} &:= \{\lambda_1 \wedge \cdots \wedge \lambda_c \mid c \leq d, \lambda_1, \ldots, \lambda_c \text{ literals}\},\\
\Delta_{0,d} &:= \{\lambda_1 \vee \cdots \vee \lambda_c \mid c \leq d, \lambda_1, \ldots, \lambda_c \text{ literals}\},\\
\Gamma_{t+1,d} &:= \Big\{ \bigwedge_{i \in I} \delta_i \mid I \text{ finite}, \delta_i \in \Delta_{t,d} \text{ for all } i \in I \Big\},\\
\Delta_{t+1,d} &:= \Big\{ \bigvee_{i \in I} \gamma_i \mid I \text{ finite}, \gamma_i \in \Gamma_{t,d} \text{ for all } i \in I \Big\}.
\end{aligned}$$

The *W-hierarchy* of parameterized complexity theory consists of the following classes.

DEFINITION 30. (1) *For* $t \geq 1$, $\mathrm{W}[t] := \bigcup_{d \geq 1} \{(Q,\kappa) \mid (Q,\kappa) \leq^{\mathrm{FPT}} p\text{-WSAT}(\Gamma_{t,d})\}$.

(2) $\mathrm{W}[\mathrm{SAT}] := \{(Q,\kappa) \mid (Q,\kappa) \leq^{\mathrm{FPT}} p\text{-WSAT}(\mathrm{FORM})\}$.

(3) $\mathrm{W}[\mathrm{P}] := \{(Q,\kappa) \mid (Q,\kappa) \leq^{\mathrm{FPT}} p\text{-WSAT}(\mathrm{CIRC})\}$.

This definition of the W-hierarchy establishes the role of weighted satisfiability problems as the "generic" (hard) problems of parameterized complexity theory. We propose that plain (unweighted) satisfiability problems with the number of variables size measure can play a similar role in exponential complexity theory. For every class Γ of Boolean formulas or circuits, we let:

s-var-SAT(Γ)
Instance: $\gamma \in \Gamma$.
Parameter: Number of variables of γ.
Problem: Decide whether γ is satisfiable.

The *S-hierarchy* consists of the following classes.

DEFINITION 31. (1) *For* $t \geq 1$, $\mathrm{S}[t] := \bigcup_{d \geq 1} \{(P,\nu) \mid (P,\nu) \leq^{\mathrm{serf}} s\text{-}var\text{-SAT}(\Gamma_{t,d})\}$.

(2) $\mathrm{S}[\mathrm{SAT}] := \{(P,\nu) \mid (P,\nu) \leq^{\mathrm{serf}} s\text{-}var\text{-SAT}(\mathrm{FORM})\}$.

(3) $\mathrm{S}[\mathrm{P}] := \{(P,\nu) \mid (P,\nu) \leq^{\mathrm{serf}} s\text{-}var\text{-SAT}(\mathrm{CIRC})\}$. So far, mainly the first level S[1] of the S-hierarchy has been studied, and some highly nontrivial completeness results are known. Most importantly, Impagliazzo, Paturi, and Zane [16] have proved that *s-var*-SAT($\Gamma_{1,3}$) (that is, 3-satisfiability) is complete for S[1] under serf Turing reductions. Thus the exponential hypothesis (discussed in the introduction) is equivalent to $\mathrm{S}[1] \neq \mathrm{SUBEPT}$.

The image of the S-hierarchy.

DEFINITION 32. (1) *For a class* C *of parameterized problems that is downward closed under serf reducibility, the* image of C under the miniaturization mapping *is the class*

$$\mathcal{M}(\mathrm{C}) := \{(Q,\kappa) \mid (Q,\kappa) \leq^{\mathrm{FPT}} \mathcal{M}(P,\nu) \text{ for some } (P,\nu) \in \mathrm{C}\big\}.$$

(2) *For a class* C *of parameterized problems that is downward closed under FPT reducibility, the* preimage of C under the miniaturization mapping *is the class*

$$\mathcal{M}^{-1}(\mathrm{C}) := \{(P,\nu) \mid \mathcal{M}(P,\nu) \in \mathrm{C}\}.$$

To relate the S-hierarchy and the W-hierarchy, we consider the image of the S-hierarchy under the miniaturization mapping (the so-called *M-hierarchy*).

Definition 33. *For $t \geq 1$, we let* $\mathrm{M}[t] := \mathcal{M}(\mathrm{S}[t])$. *Furthermore, we let* $\mathrm{M}[\textsc{Sat}] := \mathcal{M}(\mathrm{S}[\textsc{Sat}])$ *and* $\mathrm{M}[\mathrm{P}] := \mathcal{M}(\mathrm{S}[\mathrm{P}])$.

The M-hierarchy is a natural hierarchy within the realm of parameterized complexity theory. It is populated not only by the miniaturizations of the satisfiability problems but also by the following reparameterizations. For every class Γ of Boolean formulas or circuits, we let:

p-$\log^{-1}$-$\textsc{Sat}(\Gamma)$
Instance: A circuit $\gamma \in \Gamma$ of size m with n inputs (variables).
Parameter: $\lceil n/\log m \rceil$.
Problem: Decide whether γ is satisfiable.

The motivation to study these problems can be explained as follows: For simplicity, let us consider $\Gamma = \textsc{Circ}$, and let m denote the size and n the number of input gates. If we parameterize $\textsc{Sat}(\textsc{Circ})$ by n, then we obtain the familiar problem s-*var*-$\textsc{Sat}(\textsc{Circ})$, which is fixed-parameter tractable; it even belongs to the class EPT. A parameterized problem gets "harder" if we decrease the parameter. For the satisfiability problem, we may consider the parameterizations $(\textsc{Sat}(\textsc{Circ}), \kappa_h)$ for functions $h : \mathbb{N} \to \mathbb{N}$, where for every circuit γ of size m with n inputs we let

$$\kappa_h(\gamma) = \left\lceil \frac{n}{h(m)} \right\rceil .$$

For constant $h = 1$, κ_h is just our old parameterization by the number of inputs, and therefore the problem $(\textsc{Sat}(\textsc{Circ}), \kappa_h) = s$-$\textsc{Sat}(\textsc{Circ})$ is fixed-parameter tractable. At the other end of the scale, for $h(m) \geq m \geq n$ we have $\kappa_h(\gamma) = 1$, and essentially $(\textsc{Sat}(\textsc{Circ}), \kappa_h)$ is just the NP-complete unparameterized problem $\textsc{Sat}(\textsc{Circ})$. More formally, the problem is complete for the parameterized complexity class para-NP (cf. [14]). Thus by increasing the function h we can shift the fixed-parameter tractable problem s-$\textsc{Sat}(\textsc{Circ})$ to a highly intractable problem that is not even contained in the class XP (unless PTIME = NP). We leave it as an easy exercise for the reader to prove that for $h(m) \in o^{\mathrm{eff}}(\log m)$ the problem $(\textsc{Sat}(\textsc{Circ}), \kappa_h)$ is still fixed-parameter tractable. For $h(m) \in \omega(\log m)$, it seems unlikely that the problem $(\textsc{Sat}(\textsc{Circ}), \kappa_h)$ is in XP, and therefore it is not so interesting from the point of view of parameterized complexity. However, for $h(m) \in \Theta(\log m)$ the problem $(\textsc{Sat}(\textsc{Circ}), \kappa_h)$ is right on the "boundary" of XP. Note that p-$\log^{-1}$-$\textsc{Sat}(\textsc{Circ})$ is $(\textsc{Sat}(\textsc{Circ}), \kappa_h)$ for $h(m) = \log m$.

While the problem p-$\log^{-1}$-$\textsc{Sat}(\Gamma)$ bears some similarity with $\mathcal{M}(s\text{-}\mathit{var}\text{-}\textsc{Sat}(\Gamma))$, the two problems are not the same. Nevertheless, it can be shown that for "well-behaved" classes Γ of circuits, which include $\textsc{Circ}$, $\textsc{Form}$, and all classes $\Gamma_{t,d}$, the two problems are FPT-equivalent [13]. Hence the M-hierarchy can be characterized directly in terms of the log-parameterizations:

$$\begin{aligned} \mathrm{M}[t] &= \bigcup_{d \geq 1} \left\{ (Q,\kappa) \;\middle|\; (Q,\kappa) \leq^{\mathrm{FPT}} p\text{-}\log^{-1}\text{-}\textsc{Sat}(\Gamma_{t,d}) \right\} \qquad \text{for } t \geq 1, \\ \mathrm{M}[\textsc{Sat}] &= \left\{ (Q,\kappa) \;\middle|\; (Q,\kappa) \leq^{\mathrm{FPT}} p\text{-}\log^{-1}\text{-}\textsc{Sat}(\textsc{Form}) \right\}, \\ \mathrm{M}[\mathrm{P}] &= \left\{ (Q,\kappa) \;\middle|\; (Q,\kappa) \leq^{\mathrm{FPT}} p\text{-}\log^{-1}\text{-}\textsc{Sat}(\textsc{Circ}) \right\}. \end{aligned}$$

From the fact that s-$\textsc{Sat}(\Gamma_{1,3})$ (i.e., 3-SAT) is complete for S[1] under serf Turing reductions, it follows that p-$\log^{-1}$-$\textsc{Sat}(\Gamma_{1,3})$ is complete for M[1] under FPT Turing

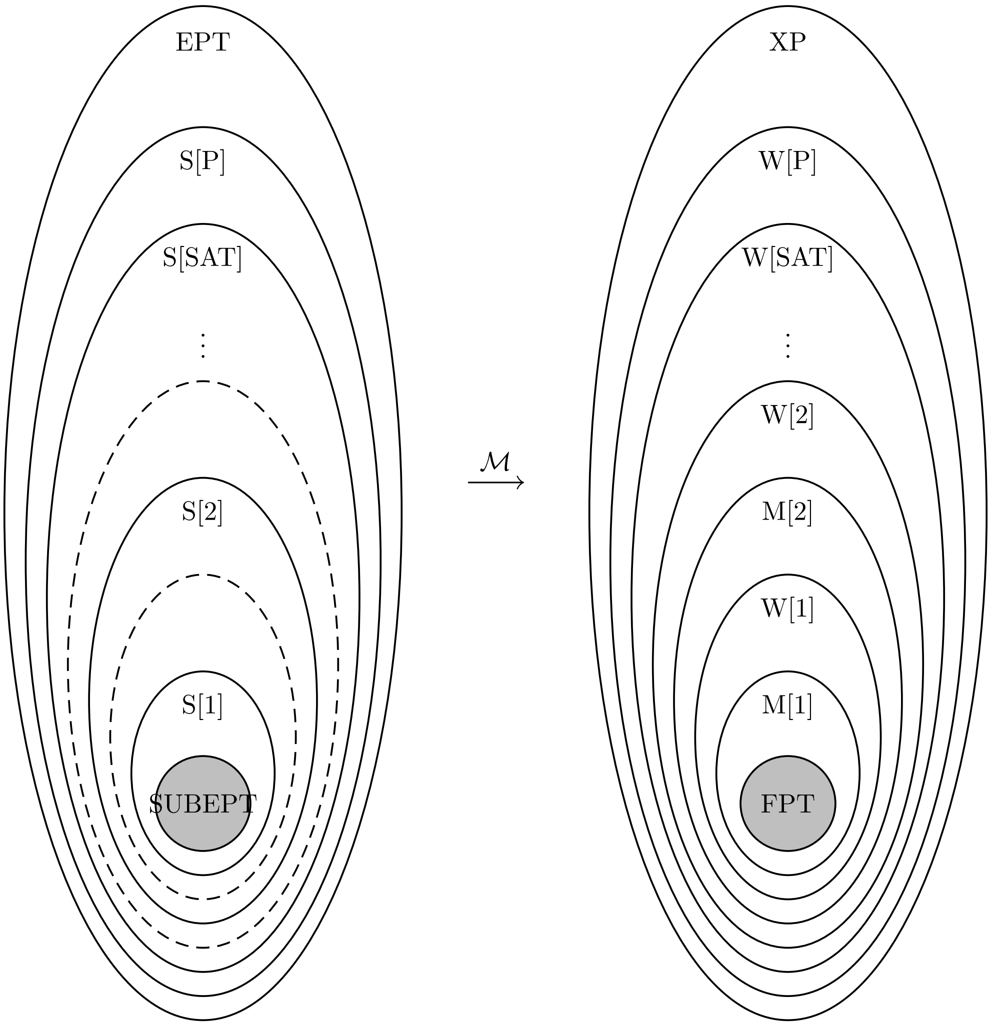


Fig. 5.1. *Overview over the complexity classes and hierarchies.*

reductions. Another natural problem complete for M[1] is the following reparameterized vertex cover problem; this result is due to Cai and Juedes [4]:

p-$\log^{-1}$-Vertex-Cover
Instance: A graph G of size m and a nonnegative integer k.
Parameter: $\lceil k/\log m \rceil$.
Problem: Decide whether G has a vertex cover of cardinality k.

The following theorem shows how the M-hierarchy relates to the W-hierarchy.

Theorem 34 (Abrahamson, Downey, and Fellows [1]). *For every* $t \geq 1$*,* $\mathrm{M}[t] \subseteq \mathrm{W}[t] \subseteq \mathrm{M}[t+1]$. *Moreover* M[Sat] = W[Sat] *and* M[P] = W[P].

For a proof, we refer the reader to [13] or [14].

A schematic image illustrating the relations between the complexity classes considered so far can be found in Figure 5.1.

The preimage of the W-hierarchy. The discussion above shows that the M-hierarchy, that is, the image of the S-hierarchy under the miniaturization mapping, is reasonably well understood and is interesting also for "intrinsically parameterized complexity" reasons. What about the preimage of the W-hierarchy in the world of exponential complexity (shown as dashed ovals in Figure 5.1)? The remainder of the section is devoted to this question.

When we studied the M-hierarchy and, more generally, the image of problems in EPT under the miniaturization mapping, we reparameterized the problems by dividing the parameter by the logarithm of the instance size, hence making the parameter smaller and the problems harder. It turned out that this increased the parameterized complexity by just the right amount, shifting problems from EPT to XP. The natural idea for the converse direction is to multiply the parameter by the logarithm of the instance size, hence making the parameter larger and the problems "easier." Again, we will see that this simple idea works.

It will be convenient to first prove that the reparameterization of problems by multiplying the parameter by the logarithm of the instance size gives an inverse for the miniaturization mapping for all problems satisfying certain technical conditions. After that, we shall prove that the W-hierarchy can be characterized in terms of problems satisfying these technical conditions.

Definition 35. *Let (Q, κ) be a parameterized problem over the alphabet Σ.*

The log reparameterization *of (Q, κ) is the mapping $\mathcal{L}$ between parameterized problems defined by $\mathcal{L}(Q, \kappa) := (Q, \lambda)$, where $\lambda : \Sigma^* \to \mathbb{N}$, with*

$$\lambda(x) = \begin{cases} \lceil \kappa(x) \cdot \log |x| \rceil & \textit{if } |x| \geq 2, \\ 1 & \textit{otherwise} \end{cases}$$

for all $x \in \Sigma^*$.

Then obviously the lemma below follows.

Lemma 36. *For all parameterized problems (Q, κ) it holds that*

$$(Q, \kappa) \leq^{\mathrm{FPT}} \mathcal{M}(\mathcal{L}(Q, \kappa)).$$

Proof. The mapping $x \mapsto (x, |x|)$ is an FPT reduction. □

We can prove a partial converse for problems that have the following property.

Definition 37. *A parameterized problem (Q, κ) over the alphabet Σ is* scalable *if there is a mapping $F : \Sigma^* \times \mathbb{N}$ such that for all $x \in \Sigma^*$ and $s \geq \max\{|x|, 2\}$ it holds that:*

(1) *$(x \in Q \iff F(x, s) \in Q)$;*

(2) *$F(x, s)$ is computable in time polynomial in $(|x| + s)$;*

(3) *$\kappa(F(x, s)) \leq \lceil \kappa(x) \cdot \frac{\log |x|}{\log s} \rceil$.*

Example 38. The parameterized independent set problem

p-Independent-Set
Instance: A graph G and $k \geq 1$.
Parameter: k.
Problem: Decide whether G has an independent set of cardinality k.

is scalable.

To see this, let $G = (V, E)$ be a graph and $k \in \mathbb{N}$. Let m be the size of the instance (G, k) and $s \geq 2$. Clearly $m \geq 2$. Let $\ell := \lceil \log s / \log m \rceil$. Then $m^{\ell - 1} <$

$s \leq m^\ell$. Without loss of generality we assume that k is a multiple of ℓ (in the proof of Lemma 42, we shall see how to avoid such an assumption). Let G' be the graph whose vertices are all independent sets of G of size ℓ, with an edge between two such independent sets if they are disjoint and their union is an independent set of size 2ℓ. Then G has an independent set of size k if and only if G' has an independent set of size

$$\frac{k}{\ell} \leq k \cdot \frac{\log m}{\log s} \leq \left\lceil k \cdot \frac{\log m}{\log s} \right\rceil .$$

We let $(G', k/\ell)$ be the scaled instance of p-INDEPENDENT-SET.

The *slices* of a parameterized problem (Q,κ) are the classical problems $Q_i = \{x \in Q \mid \kappa(x) = i\}$ for $i \geq 1$. Observe that, if a problem is scalable and its first slice is polynomial time decidable, then the problem is in XP. To see this, let (Q,κ) be a scalable problem such that Q_1 is polynomial time decidable. Let x be an instance. Scale x with $s = |x|^{\kappa(x)}$. Then the resulting instance x' has parameter value 1. Hence $x' \in Q$ can be decided in polynomial time. Note that this actually gives an algorithm deciding $x \in Q$ in time $|x|^{O(\kappa(x))}$.

LEMMA 39. *Let (Q,κ) be a scalable problem. Then*

$$\mathcal{M}\left(\mathcal{L}\left(Q,\kappa\right)\right) \equiv^{\mathrm{FPT}} (Q,\kappa).$$

Proof. By Lemma 36, we have $(Q,\kappa) \leq^{\mathrm{FPT}} \mathcal{M}\left(\mathcal{L}\left(Q,\kappa\right)\right)$. Thus we have only to prove the converse. Let Σ be the alphabet of Q. Let $F : \Sigma^* \times \mathbb{N} \to \Sigma^*$ be a function witnessing that (Q,κ) is scalable. We define a reduction $R : \Sigma^* \times \mathbb{N} \to \Sigma^*$ by:

$$R(x,m) := \begin{cases} F(x,m) & \text{if } m \geq |x| \geq 2, \\ x & \text{otherwise.} \end{cases}$$

Obviously, $(x,m) \in \mathcal{M}\left(\mathcal{L}\left(Q,\kappa\right)\right)$ if and only if $R(x,m) \in Q$. Furthermore, R is polynomial time computable because F is. To see that the parameter of the R-image of an instance can be bounded in terms of the parameter of the instance, let (x,m) be an instance of $\mathcal{M}\left(\mathcal{L}\left(Q,\kappa\right)\right)$ and $x' := R(x,m)$. If $|x| < 2$, then $x' = x$, and $\kappa(x') = \kappa(x)$ is bounded by a constant, since there are only finitely many x with $|x| < 2$. So we assume $|x| \geq 2$ and let

$$k := \begin{cases} \left\lceil \frac{\lceil \kappa(x) \cdot \log|x| \rceil}{\log m} \right\rceil & \text{if } m \geq 2, \\ \lceil \kappa(x) \cdot \log|x| \rceil & \text{otherwise} \end{cases}$$

be the parameter value of (x,m). If $m \geq |x|$, then $x' = F(x,m)$ and

$$\kappa(x') \leq \left\lceil \frac{\kappa(x) \cdot \log|x|}{\log m} \right\rceil \leq k,$$

where the first inequality holds by the definition of scalability. Otherwise $m < |x|$. It follows that $x' = x$ and $\kappa(x') = \kappa(x)$.

- If $m \geq 2$, then

$$\kappa(x) = \frac{\kappa(x) \cdot \log m}{\log m} < \frac{\kappa(x) \cdot \log|x|}{\log m} \leq \left\lceil \frac{\lceil \kappa(x) \cdot \log|x| \rceil}{\log m} \right\rceil = k.$$

- Otherwise, $m = 1$, and we have

$$k = \lceil \kappa(x) \cdot \log|x| \rceil \geq \kappa(x). \qquad \square$$

The notation in the following example requires explanation: For functions $f, g : \mathbb{N}^2 \to \mathbb{N}$ we say that $g \in o^{\text{eff}}(f)$ if there is a computable function $\iota : \mathbb{N} \to \mathbb{N}$ that is nondecreasing and unbounded such that for all m, k it holds that $g(m,k) \leq f(m,k)/\iota(m+k)$. We say that $g \in 2^{o^{\text{eff}}(f(m,k))}$ if there is a function $g' \in o^{\text{eff}}(f)$ such that $g(m,k) \leq 2^{g'(m,k)}$ for all m, k. Some care needs to be taken with this notation. For example,

$$2^{o^{\text{eff}}(k \cdot \log n)} \subset o^{\text{eff}}(n^k).$$

An example of a function in $o^{\text{eff}}(n^k) \setminus 2^{o^{\text{eff}}(k\cdot \log n)}$ is $n^{k/2}$.

Example 40. The log reparameterization of p-Independent-Set is the problem:

s-log-Independent-Set	
Instance:	A graph G of size m and $k \geq 1$.
Parameter:	$\lceil k \cdot \log m \rceil$.
Problem:	Decide whether G has an independent set of cardinality k.

Hence the miniaturization mapping maps s-log-Independent-Set to a parameterized problem FPT-equivalent to p-Independent-Set or, equivalently, maps the serf-degree $[\![s\text{-log-Independent-Set}]\!]^{\text{serf}}$ to the FPT-degree $[\![p\text{-Independent-Set}]\!]^{\text{FPT}}$. As p-Independent-Set is complete for W[1] under FPT reductions [11], it follows that s-log-Independent-Set is complete for $\mathcal{M}^{-1}(\text{W}[1])$ under serf reductions.

An interesting consequence of this result is that FPT $=$ W[1] if and only if there is an algorithm deciding whether a graph of size m has an independent set of size k in time $2^{o^{\text{eff}}(k\cdot\log m)} \cdot m^{O(1)}$ or, equivalently, an algorithm deciding whether an n-vertex graph has an independent set of size k in time $2^{o^{\text{eff}}(k\cdot\log n)} \cdot n^{O(1)}$.

We cannot apply the same technique as in the previous example to the defining problems p-WSat$(\Gamma_{t,d})$ of the W-hierarchy directly, because they are not (obviously) scalable. We take a detour through the monotone and antimonotone versions of these problems. Let us call a Boolean formula *monotone* if it contains no negations and *antimonotone* if all variables are negated and no other negations occur. For a class Φ of formulas, we use Φ^+ to denote the class of monotone formulas in Φ and similarly Φ^- for the antimonotone formulas.

Lemma 41 (Downey and Fellows [10, 11]). *Let $t, d \in \mathbb{N}$ such that $t + d \geq 3$.*

(1) *If t is even, then p-*WSat$(\Gamma^+_{t,d})$ *is complete for* W[t] *under* FPT *reductions.*

(2) *If t is odd, then p-*WSat$(\Gamma^-_{t,d})$ *is complete for* W[t] *under* FPT *reductions.*

Lemma 42. *Let $t, d \in \mathbb{N}$ such that $t + d \geq 3$.*

(1) *If t is even, then p-*WSat$(\Gamma^+_{t,d})$ *is scalable.*

(2) *If t is odd, then p-*WSat$(\Gamma^-_{t,d})$ *is scalable.*

*Furthermore, the problems p-*WSat(Form) *and p-*WSat(Circ) *are scalable.*

Proof. Assume first that t is odd. Let $\gamma \in \Gamma^-_{t,d}$ be a formula of size m with n variables. Without loss of generality we assume $m \geq n \geq 2$ and let $k, s \in \mathbb{N}$ such that $n \geq k \geq 1$, $s \geq m$, and $\ell := \lceil \log s / \log n \rceil$. Let us assume first that k is a multiple of ℓ.

Let V denote the set of variables of γ. For each ℓ-element subset $S \subseteq V$ we introduce a new variable $Y_S \notin V$. Now we replace each negative literal $\neg X$ in γ by the conjunction

$$\bigwedge_{\substack{S \subseteq V \text{ with} \\ |X| = \ell \text{ and } X \in S}} \neg Y_S.$$

The resulting formula can be transformed into an equivalent $\Gamma^-_{t,d}$ formula γ' in time $O(m^{d\cdot\ell})$. We let

$$\gamma'' := \gamma' \wedge \bigwedge_{\substack{S,S' \subseteq V \text{ with} \\ |S|=|S'|=\ell \text{ and } S\cap S' \neq \emptyset}} (\neg Y_S \vee \neg Y_{S'}).$$

As $t + d \geq 3$, the resulting formula is still in $\Gamma^-_{t,d}$. It is easy to see that γ'' has a satisfying assignment of Hamming weight k/ℓ if and only if γ has a satisfying assignment of Hamming weight k. Furthermore,

$$\frac{k}{\ell} \leq k \cdot \frac{\log n}{\log s} \leq k \cdot \frac{\log m}{\log s} \leq \left\lceil k \cdot \frac{\log m}{\log s} \right\rceil.$$

It remains to explain what we do if k is a multiple not of ℓ. We let $\tilde{k}$ be the least multiple of ℓ greater than k. Then we choose $\tilde{s} \geq s$ such that

$$\left\lceil \log \tilde{s} / \log(n + 1 + \tilde{k} - k) \right\rceil = \ell.$$

We let $\tilde{\gamma}$ be the $\Gamma^+_{t,d}$ formula

$$\gamma \wedge (\neg Y_1 \vee \neg Z) \wedge \cdots \wedge (\neg Y_{\tilde{k}-k} \vee \neg Z),$$

where the $Y_1, \ldots, Y_{\tilde{k}-k}$, Z are new variables not in γ. Then $\tilde{\gamma}$ has a satisfying assignment of Hamming weight $\tilde{k}$ if and only if γ has a satisfying assignment of Hamming weight k. We apply the construction above to the instance $(\tilde{\gamma}, \tilde{k})$ and $\tilde{s}$ to obtain a $\Gamma^+_{t,d}$ formula $\tilde{\gamma}''$ that has a satisfying assigment of Hamming weight $\tilde{k}/\ell$ if and only if γ has a satisfying assignment of weight k. As $\lceil \tilde{k}/\ell \rceil = \lceil k/\ell \rceil$ by the choice of $\tilde{k}$, this is good enough.

Assume now that t is even. Let $\gamma \in \Gamma^+_{t,d}$ be a formula of size m with n variables, and let $k, s \in \mathbb{N}$ such that $n \geq k \geq 1, s \geq m$. Let $\ell := \lceil \log s / \log n \rceil$. Without loss of generality, we may assume that k is a multiple of ℓ. Let V be the set of variables of γ. For each ℓ-element subset $S \subseteq V$ we introduce a new variable Y_S. Now we replace each variable X in γ by the disjunction $\bigvee_{S \subseteq V \text{ with } X \in S} Y_S$. The resulting formula can be transformed into an equivalent $\Gamma^+_{t,d}$ formula γ' in time $O(m^{d\cdot\ell})$. It is easy to see that γ' has a satisfying assignment of Hamming weight k/ℓ if γ has a satisfying assignment of Hamming weight k. Conversely, if γ' has a satisfying assignment of Hamming weight k/ℓ, then γ has a satisfying assignment of Hamming weight at most k and hence, by monotonicity, a satisfying assignment of weight exactly k. Moreover, we have $k/\ell \leq \lceil k \cdot \log m / \log s \rceil$ (as above).

Using similar ideas, it is easy to prove that p-WSAT(FORM) and p-WSAT(CIRC) are scalable. □

For every class Γ of Boolean formulas or circuits, we let:

s-log-WSat(Γ)
Instance: $\gamma \in \Gamma$ of size m and $k \in \mathbb{N}$.
Parameter: $\lceil k \cdot \log m \rceil$.
Problem: Decide whether γ has a satisfying assignment of Hamming weight k.

Corollary 43. *Let $t, d \in \mathbb{N}$ such that $t + d \geq 3$.*

(1) *If t is even, then s-log-WSat($\Gamma^+_{t,d}$) is complete for $\mathcal{M}^{-1}(\mathrm{W}[t])$ under* serf *reductions.*

(2) *If t is odd, then s-log-WSat($\Gamma^-_{t,d}$) is complete for $\mathcal{M}^{-1}(\mathrm{W}[t])$ under* serf *reductions.*

As $\mathcal{M}^{-1}(\mathrm{W[Sat]}) = \mathrm{S[Sat]}$ and $\mathcal{M}^{-1}(\mathrm{W[P]}) = \mathrm{S[P]}$, we also get the following.

Corollary 44. (1) *s-log-WSat(Form) is complete for* S[Sat] *under* serf *reductions.*

(2) *s-log-WSat(Circ) is complete for* S[P] *under* serf *reductions.*

Note that, combined with our earlier results, (2) implies the following corollary. (Statement (1) has a similar consequence for formulas, which we do not state explicitly.)

Corollary 45. *The following statements are equivalent:*

(1) *There is an algorithm deciding if a circuit of size m with n inputs is satisfiable in time $2^{o^{\mathrm{eff}}(n)} \cdot m^{O(1)}$.*

(2) *There is an algorithm deciding if a circuit of size m with n inputs has a satisfying assignment of Hamming weight k in time $2^{o^{\mathrm{eff}}(k \cdot \log m)} \cdot m^{O(1)}$.*

Corollary 43 already yields a characterization of the preimage of the W-hierarchy under the miniaturization mapping, but we find it not yet completely satisfactory because it involves a restriction of the satisfiability problems not used so far. Fortunately, we can easily get rid of this restriction.

Lemma 46. *Let $t, d \in \mathbb{N}$ such that $t + d \geq 3$.*

(1) *If t is even, then s-log-WSat($\Gamma_{t,d}$) $\equiv^{\mathrm{serf}}$ s-log-WSat($\Gamma^-_{t,d}$).*

(2) *If t is odd, then s-log-WSat($\Gamma_{t,d}$) $\equiv^{\mathrm{serf}}$ s-log-WSat($\Gamma^+_{t,d}$).*

Proof. In [15], the authors give polynomial time reductions from WSat($\Gamma_{t,d}$) to WSat($\Gamma^+_{t,d}$) for even t and from WSat($\Gamma_{t,d}$) to WSat($\Gamma^+_{t,d}$) for odd t that are linear in the parameter. These reductions can easily be turned into the desired serf reductions. $\square$

Finally, we are ready to state and prove the main result of this section.

Theorem 47. *For every $t \geq 1$,*

$$\mathcal{M}^{-1}\big(\mathrm{W}[t]\big) = \bigcup_{d \geq 1} \big\{(P, \nu) \mid (P, \nu) \leq^{\mathrm{serf}} s\text{-log-WSat}(\Gamma_{t,d})\big\}.$$

6. Concluding remarks.

This paper is a contribution to a (not yet clearly established) exponential complexity theory. A long-term goal of such a theory might be to prove that the exponential time hypothesis is equivalent to $\mathrm{P} \neq \mathrm{NP}$ and thus obtain a solid basis for exponential lower bounds such as the one for 3-Sat stated by the exponential time hypothesis. However, with current methods this goal seems out of reach, and maybe such an equivalence cannot be established without actually proving that the exponential time hypothesis and hence P $\neq$ NP holds.[4]

What we can establish now is a close connection between exponential and parameterized complexity theory. The obvious open question is whether the exponential time hypothesis is equivalent to FPT $\neq$ W[1] or, more or less equivalently, whether M[1] = W[1]. Despite serious efforts, so far researchers in parameterized complexity have not been able to prove that M[1] = W[1], even though this still seems quite plausible. However, it would also be compatible with our current knowledge that M[2] = W[1].

The higher levels of the S-hierarchy (or, equivalently, the M-hierarchy) have not yet received much attention. Specifically, no completeness results for S[2] are known, even though the class contains natural problems such as *s-var*-SAT that are not believed to be in S[1]. It may be worthwhile to study the class S[2] and develop a completeness theory for this class similar to the S[1]-completeness theory, which is based on Impagliazzo, Paturi, and Zane's sparsification lemma [16].

REFERENCES

[1] K.A. Abrahamson, R.G. Downey, and M.R. Fellows, *Fixed-parameter tractability and completeness* IV: *On completeness for W[P] and PSPACE analogs*, Ann. Pure Appl. Logic, 73 (1995), pp. 235–276.

[2] J. Alber, H. Fernau, and R. Niedermeier, *Parameterized complexity: Exponential speed-up for planar graph problems*, in Proceedings of the 28th International Colloquium on Automata, Languages and Programming, F. Orejas, P.G. Spirakis, and J. van Leeuwen, eds., Lecture Notes in Comput. Sci. 2076, Springer-Verlag, Berlin, 2001, pp. 261–272.

[3] T. Brueggemann and W. Kern, *An improved deterministic local search algorithm for* 3*-SAT*, Theoretical Computer Science, 329 (2004), pp. 303–313.

[4] L. Cai and D. Juedes, *On the existence of subexponential parameterized algorithms*, J. Comput. System Sci., 67 (2003), pp. 789–807.

[5] J. Chen, B. Chor, M. Fellows, X. Huang, D. Juedes, I. Kanj, and G. Xia, *Tight lower bounds for certain parameterized NP-hard problems*, Inform Comput., 201 (2005), pp. 216–231.

[6] J. Chen, X. Huang, I. Kanj, and G. Xia, *Strong computational lower bounds via parameterized complexity*, J. Comput. System Sci., 72 (2006), pp. 1346–1367.

[7] Y. Chen and J. Flum, *On miniaturized problems in parameterized complexity*, Theoretical Computer Science, 351 (2006), pp. 314–336.

[8] S. Demri, F. Laroussinie, and Ph. Schnoebelen, *A parametric analysis of the state explosion problem in model checking*, in Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science, H. Alt and A. Ferreira, eds., Lecture Notes in Comput. Sci. 2285, Springer-Verlag, Berlin, 2002, pp. 620–631.

[9] R.G. Downey, V. Estivill-Castro, M. Fellows, E. Prieto-Rodriguez, and F. Rosamond, *Cutting up is hard to do: The parameterized complexity of k-cut and related problems*, in Proceedings of the Australian Theory Symposium, J. Harland, ed., Electron. Notes Theor. Comput. Sci. 78, Elsevier, New York, 2003.

[10] R.G. Downey and M.R. Fellows, *Fixed-parameter tractability and completeness* I: *Basic results*, SIAM J. Comput., 24 (1995), pp. 873–921.

[11] R.G. Downey and M.R. Fellows, *Fixed-parameter tractability and completeness* II: *On completeness for W*[1], Theoretical Computer Science, 141 (1995), pp. 109–131.

[12] R.G. Downey and M.R. Fellows, *Parameterized Complexity*, Springer-Verlag, Berlin, 1999.

[13] J. Flum and M. Grohe, *Parameterized complexity and subexponential time*, Bull. Eur. Assoc. Theor. Comput. Sci. EATCS, 84 (2004), pp. 71–100.

[14] J. Flum and M. Grohe, *Parameterized Complexity Theory*, Springer-Verlag, Berlin, 2006.

[15] J. Flum, M. Grohe, and M. Weyer, *Bounded fixed-parameter tractability and $\log^2 n$ nondeterministic bits*, J. Comput. System Sci., 72 (2006), pp. 34–71.

[4] However, 20 years ago an equivalence between the inapproximability of, say, MAX-SAT and P $\neq$ NP also must have seemed far out of reach.

[16] R. Impagliazzo, R. Paturi, and F. Zane, *Which problems have strongly exponential complexity?* J. Comput. System Sci., 63 (2001), pp. 512–530.

[17] D. Rolf, *Improved bound for the PPSZ/Schöing-algorithm for* 3-*SAT*, Journal on Satisfiability, Boolean Modeling and Computation, 1 (2006), pp. 111–122.

[18] U. Schöning, *Algorithmics in exponential time*, in Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science, V. Diekert and B. Durand, eds., Lecture Notes in Comput. Sci. 3404, Springer-Verlag, Berlin, 2005, pp. 36–43.

[19] G. Woeginger, *Space and time complexity of exact algorithms: Some open problems*, in Proceedings of the 1st International Workshop on Parameterized and Exact Computation, R.G. Downey, M. Fellows, and F. Dehne, eds., Lecture Notes in Comput. Sci. 3162, Springer-Verlag, Berlin, 2004, pp. 36–43.

SIAM J. COMPUT.
Vol. 37, No. 4, pp. 1259–1279

LOCALIZED CLIENT-SERVER LOAD BALANCING WITHOUT GLOBAL INFORMATION*

BARUCH AWERBUCH†, MOHAMMAD T. HAJIAGHAYI‡, ROBERT KLEINBERG§, AND TOM LEIGHTON¶

Abstract. We consider distributed algorithms for maximizing throughput in a network of clients and servers, modeled as a bipartite graph. We seek algorithms and lower bounds for decentralized algorithms in which each participant has only local knowledge about the state of itself and its neighbors. Our problem is analogous to recent work on oblivious routing [M. Bienkowski, M. Korzeniowski, and H. Räcke, *Proceedings of the* 15*th Annual ACM Symposium on Parallel Algorithms and Architectures*, 2003, pp. 24–33, C. Harrelson, K. Hildrum, and S. Rao, *Proceedings of the* 15*th Annual ACM Symposium on Parallel Algorithms and Architectures*, 2003, pp. 34–43, H. Räcke, *Proceedings of the* 43*rd Annual IEEE Symposium on Foundations of Computer Science*, 2002, pp. 43–52] but with the objective of maximizing throughput rather than minimizing congestion. In contrast to that work, we prove a strong lower bound (polynomial in n, the size of the graph) on the competitive ratio of any oblivious algorithm. This is accompanied by simple algorithms achieving upper bounds which are tight in terms of k, the maximum throughput achievable by an omniscient algorithm, and are also tight in terms of m, the number of servers. Finally, we investigate an online version of the problem, in a restricted model which requires that clients, upon becoming active, must remain so for at least $\log(n)$ time steps. In contrast to our primarily negative results in the oblivious case, here we present an algorithm which is constant-competitive. Our lower bounds justify the intuition, implicit in earlier work on the subject [B. Awerbuch and Y. Azar, *Proceedings of the* 35*th Annual IEEE Symposium on Foundations of Computer Science*, 1994, pp. 240–249], that some such restriction (i.e., requiring some stability in the demand pattern over time) is necessary in order to achieve a constant—or even polylogarithmic—competitive ratio.

Key words. distributed algorithms, maximum matching, multicommodity flow, oblivious routing

AMS subject classifications. 68W15, 68W20

DOI. 10.1137/S009753970444661X

1. Introduction.

We consider distributed algorithms for maximizing throughput in a network of clients and servers, modeled as a bipartite graph $G = (V_L, V_R, E)$ with V_L representing the clients, V_R representing the servers, and E representing the client-server assignments which are considered admissible, e.g., because of proximity constraints. Motivated by Internet load-balancing applications, such as load-balancing HTTP connections in a content delivery network, we consider the case in which client-server connections are extremely short-lived (lasting for only one unit of time) and it

*Received by the editors December 3, 2004; accepted for publication (in revised form) April 30, 2007; published electronically December 19, 2007. A preliminary version of this paper appeared in *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2005.

http://www.siam.org/journals/sicomp/37-4/44661.html

†Center for Networking and Distributed Systems, Johns Hopkins University, Baltimore, MD, 21218 (baruch@cs.jhu.edu).

‡Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 32 Vassar St., Cambridge, MA 02139 (hajiagha@theory.lcs.mit.edu).

§Department of Computer Science, Cornell University, Ithaca, NY 14850 (rdk@cs.cornell.edu). This author was supported by a Fannie and John Hertz Foundation Fellowship and a National Science Foundation Mathematical Sciences Postdoctoral Research Fellowship. Portions of this work were completed while this author was a Ph.D. student in the Department of Mathematics at MIT.

¶Department of Mathematics and Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 32 Vassar St., Cambridge, MA 02139, and Akamai Technologies, 8 Cambridge Center, Cambridge, MA 02139 (ftl@theory.lcs.mit.edu).

is impossible to get an instantaneous snapshot of the demand pattern. Our focus is on distributed algorithms in which clients must make decisions knowing nothing about the current demand pattern other than their own demand, and servers must make decisions knowing nothing other than what they learn from their adjacent clients. (We also assume that servers may report their load to the adjacent clients at the end of a round, though this is not necessarily predictive of their load in future rounds.) This emphasis on decentralized algorithms with an extremely limited amount of communication distinguishes the present paper from most of the previous work on distributed load balancing.

Most of the present paper is devoted to analyzing the following game between an adversary and a distributed network of clients and servers, joined together in a bipartite graph with edges connecting clients to servers. First, the adversary specifies a subset of clients as *active*. Next, each active client chooses an adjacent server (possibly using randomization) and sends a request to that server. In choosing a server, a client is not assumed to know any information other than the set of servers to whom it is connected and the fact that the client itself is active. Finally, each server which received at least one request chooses to satisfy one of those requests. In choosing to satisfy a request, the server is not assumed to know any information other than the set of clients from whom it received requests. We say that an algorithm has *competitive ratio* ρ, or is ρ-competitive, if it is the case that no matter what set of clients the adversary designates as active, the expected number of active clients whose requests are satisfied is at least k/ρ, where k is the maximum number of requests that could possibly have been satisfied, i.e., the size of a maximum matching between active clients and servers. While it is possible to achieve competitive ratios significantly better than the trivial $O(n)$ bound for this problem, we show that it is impossible to achieve a polylog(n) competitive ratio. (Here n denotes the total number of clients in the network.)

As a counterpoint to these primarily negative results, we consider an online version of the same problem. In this online problem, the game described above (henceforth, the "one-shot game") is played repeatedly, with the adversary specifying a potentially different set of active clients each time. If no restriction is placed on the adversary's behavior over time, then the repeated game reduces to a sequence of unrelated instances of the one-shot game, and there is no essential difference between the online setting and the one-shot setting: the best competitive ratio achievable by randomized algorithms in the online case is identical to the best competitive ratio achievable by randomized algorithms in the one-shot case. However, the situation becomes quite different when we consider a restricted adversarial model in which a client who becomes active must remain active for at least $r > 0$ rounds thereafter. In this environment, we present an algorithm whose competitive ratio is $O(\Delta^{6/r})$, where Δ is an upper bound (known to all parties) on the degree of any client. In particular, the algorithm achieves a constant competitive ratio when $r = \Omega(\log \Delta)$. Our algorithm is structurally similar to the concurrent routing algorithm of Awerbuch and Azar [2], with two important differences: the latter algorithm assumes that clients are not entering and leaving the system over time, and it requires the clients to gradually increase their flow until eventually reaching the desired level of throughput. Our algorithm permits clients to become active and inactive over time (provided that a client, upon becoming active, remains active for the next r steps), and it permits them to route their full demand in each round in which they are active (though the demand may not be satisfied if it is sent to a congested server).

All of the algorithms presented in this paper are very easy to implement, requiring straightforward decision making and communication protocols on the part of clients and servers. Some of the lower bound proofs, on the other hand, are relatively sophisticated. We show that oblivious algorithms for throughput maximization can be obstructed by the presence of substructures in the bipartite graph which we call γ-focal matchings. The task of proving competitive-ratio lower bounds is thereby reduced to a combinatorial problem of packing as many γ-focal matchings as possible into a bipartite graph of size n. Our construction of such graphs involves an interesting mixture of combinatorial, algebraic, and probabilistic techniques. These lower bound techniques constitute one of the main contributions of this paper, and we believe it may be interesting to consider whether they can be used to obtain lower bounds for other problems.

2. Related work. Recall that this paper considers load balancing for a client-server model which has two essential characteristics. The first one is that our system is fully dynamic and the input can change drastically from one time period to the next. The second one is that there is no central “dispatcher” in the system that could communicate the result of the maximum matching computation to the clients, thus guiding their routing decisions. Indeed, the interplay of these two aspects plays an important role in this paper, since otherwise there are many algorithms in the literature for models possessing only one of these characteristics. Below, we review some of these results.

2.1. Centralized control. Finding a maximum matching, or its generalization to maximum flow, is one of the classical problems in combinatorial optimization. The fastest known sequential algorithm for the problem has running time close to $O(|E||V|)$ [12]. For the more general problem of solving a positive linear program to within a $(1+\epsilon)$ factor of optimality, Plotkin, Shmoys, and Tardos [19] present a sequential algorithm which repeatedly identifies a globally minimum weight path and pushes more flow along that path. The algorithm of Plotkin et al. is further improved by Garg and Könemann [10], who give faster and simpler primal-dual algorithms for multicommodity flow and other fractional packing problems with the same approximation factor $(1+\epsilon)$. In addition, several (deterministic and randomized) parallel algorithms for maximum bipartite matching and maximum flow have been proposed (see, e.g., [9, 12, 15]). Although these algorithms have efficient implementations, they are all centralized algorithms and require global knowledge of the demand pattern and global coordination, which make them unsuitable for fast distributed implementation with local information.

2.2. Distributed control with persistent demands.

Routing and admission control. Assuming that the demand pattern remains stable for at least $\Omega(\log n)$ rounds at a time, a distributed routing and flow control algorithm with a global objective function has been given by Awerbuch and Azar [2]. This work is based on fundamental results from competitive analysis [1, 3] and assumes clients can gradually increase their flow; while the flow is still small it could, for example, be buffered at the client. In this case, under the assumption that there is a small number of routing paths, they provide an $O(\log n)$-competitive algorithm for the routing problem, which takes a polylogarithmic number of rounds to converge. Awerbuch and Leighton [4, 5] have suggested general methods for distributed routing and admission control that use a polynomial amount of buffer space. Our lower bounds demonstrate that at least one of these two assumptions—persistence of demands over time, or the ability to buffer packets—is really required in order to achieve a polylogarithmic competitive ratio.

Distributed admission control alone. For the distributed admission control problem (in which clients do not choose a server or routing path but only their sending rate) Papadimitriou and Yannakakis [18] initiated the study of flow control using distributed routers based only on local information. More precisely, they presented a framework for solving positive linear programs by distributed agents. Luby and Nisan [16], Bartal, Byers, and Raz [7], and Garg and Young [11] obtained $(1 + \epsilon)$-competitive algorithms converging in a polylogarithmic number of rounds.

Even though all of these results are distributed, they converge to their final solution in a polylogarithmic number of rounds, which makes them unsuitable for our client-server model.

2.3. Distributed control without persistence of demands. One possible approach to distributed load-balancing is to use an "oblivious" solution. Such an oblivious algorithm exists for the congestion minimization problem in undirected edge-capacitated graphs (see Räcke's original paper [20] and its subsequent improvement by Harrelson, Hildrum, and Rao [14]) and for directed and node-capacitated graphs [13]. No such solution exists for the throughput problem, though Räcke and Rosén [21] (independently and concurrently with our work) gave a distributed online call control algorithm which is closely related to oblivious throughput maximization in undirected graphs. One of the main results in our paper establishes nearly tight upper and lower bounds on the performance of oblivious routing schemes in *directed bipartite graphs*, in terms of throughput. The performance gap between the optimal and oblivious solution is polynomial; our lower bounds show that this gap is inherent. A comparably strong lower bound for oblivious routing in bipartite directed graphs, with the objective of congestion minimization, was established using a simple construction in [6]. Our lower bound requires a significantly more sophisticated construction because we seek a lower bound on competitive ratio for *throughput* rather than edge congestion. This is the first polynomial lower bound on throughput for oblivious routing.

3. Formal model and statement of results. Our graph terminology is as follows. All the graphs in this paper are directed bipartite graphs without multiple edges. For such a graph $G = (V_L, V_R, E)$, we will refer to elements of V_L as *clients* and elements of V_R as *servers*. The number of clients is denoted by n and the number of servers by m. The edges of E are directed from clients to servers. For a vertex set $S \subseteq V_L \cup V_R$ we denote the set of adjacent vertices by $\Gamma(S)$, the set of outgoing edges by $\delta^+(S)$, and the set of incoming edges by $\delta^-(S)$. When S is a singleton set $\{v\}$, these will be abbreviated to $\Gamma(v), \delta^+(v), \delta^-(v)$. The degree of a vertex v will be denoted by $d(v)$.

The prototypical problem we will analyze is the following throughput maximization problem. An adversary designates a set S of clients, called the *active clients.* Each active client i generates a request and must choose a (possibly random) adjacent server to which it will send this request, without knowing which other clients are active. Each server that receives one or more requests may choose to satisfy any one of them. The goal is to maximize the expected number of satisfied requests, called the *throughput.* The algorithm is judged according to its competitive ratio, i.e., the ratio of its throughput to that of the omniscient algorithm which chooses a throughput-maximizing assignment. We will use the letter k to denote the throughput of the optimal assignment, i.e., the size of a maximum matching from the active clients to

V_R.

The following variants of the problem are also of interest.

Multicast model. In contrast to the *unicast model* described above, we may consider a model in which a client may send its request to any subset of the adjacent servers. A server receiving one or more requests may choose to satisfy any one of them, but it must make this choice without any knowledge about the set of active clients other than the information contained in the requests it received. The throughput is defined as the number of *distinct* clients whose requests are satisfied; i.e., a client whose request is satisfied by two or more servers still contributes only 1 to the throughput. (This definition of throughput is appropriate if we suppose that each client generates at most one job and that when many servers satisfy one client's request they are duplicating each other's work.)

Fractional assignments. Instead of requiring each active client i to choose one of its adjacent servers, it may choose a fractional load distribution among its outgoing edges. In other words, each client chooses a function $f_i : \delta^+(i) \to [0,1]$ satisfying $\sum_{e \in \delta^+(i)} f(e) \leq 1$. As always, client i must specify f_i without knowing which other clients are in S. The load on a server j, denoted by $\ell(j)$, is equal to the total load on all incoming edges. The throughput is defined by $\sum_{j \in V_R} \min\{1, \ell(j)\}$.

Online with restricted adversary. In the online restricted adversary model, we assume that the adversary designates a *sequence* of instances of the throughput maximization problem described above by specifying a different set S_t of active clients in each time period t. The adversary is restricted to choose a sequence of sets satisfying the following constraint: every client, upon becoming active, must remain active for at least r rounds. In other words, if $i \in S_t$, then there exist t_0, t_1 such that $t_0 \leq t \leq t_1$, $t_1 - t_0 \geq r$, and $i \in S_{t'}$ for $t_0 \leq t' \leq t_1$. (We call r the *minimum activity period.*) We also assume that servers may report their load and capacity to the adjacent clients at the end of each round. In distinguishing this online model from the original throughput maximization problem stated at the beginning of this section, we will call the latter problem the *one-shot model.*

In proving lower bounds in this paper, we will assume that the structure of the entire graph G is known to all clients and servers and that they have access to an unlimited supply of shared random bits. In contrast, our upper bounds will be based on algorithms which require much less knowledge on the part of the participants: each vertex needs only know which vertices are adjacent to it. (In section 7 we must also assume that they share a common estimate of the maximum client degree, Δ.)

The following theorems summarize our main results.

THEOREM 1. *In the unicast one-shot model, there is an algorithm whose competitive ratio is $O(\sqrt{k})$, and this bound is tight in terms of k, even if the algorithm is randomized and is allowed to use fractional assignments. In terms of m, the same algorithm has a competitive ratio of $O(\sqrt{m})$, and this is again tight. In terms of n, the competitive ratio of any algorithm is $\Omega(n^{0.103})$.*

THEOREM 2. *In the multicast one-shot model, there is an algorithm whose competitive ratio is $O(k^{1/3})$, provided that the servers know the degree of their adjacent clients or that the clients can communicate this information in their request headers. This bound is tight in terms of k, even if the clients are allowed to put an arbitrary amount of information in the request header. In terms of m, the same algorithm has a competitive ratio of $O(m^{1/3})$, and this is again tight. In terms of n, the competitive ratio of any algorithm is $\Omega(n^{0.069})$.*

THEOREM 3. *In the restricted adversary model with fractional assignments and* *with minimum activity period r, if the clients know the value of r as well as an upper bound Δ on the maximum degree of any client, then there is an algorithm whose competitive ratio is $O(\Delta^{6/r})$. In particular, if $r = \Omega(\log \Delta)$, the competitive ratio is constant.*

4. Lower bounds for the one-shot model. Our lower bounds in the one-shot model depend on finding matchings M between a set of clients M_L and servers M_R, such that removing M from the edge set of G leaves M_L with a very small set of neighbors. We call such structures γ-focal matchings; the structure is illustrated in Figure 1 and defined precisely as follows.

DEFINITION 1. *Let M be a matching in G, and let M_L, M_R denote the sets of left and right endpoints, respectively, of the matching edges. We call M a γ-*focal matching *if $|\Gamma(M_L) \setminus M_R| < |M|/\gamma$ and G contains no edges between M_L and M_R other than those which belong to M.*

Intuitively, the presence of many disjoint γ-focal matchings in G is a barrier to achieving high throughput in an oblivious assignment algorithm for the following reason. When the set of active clients is equal to M_L for some γ-focal matching M, the optimum throughput is $|M|$. Any assignment achieving throughput significantly higher than $|M|/\gamma$ must send many requests along the edges of M, because all other outgoing edges from M_L lead to the set $\Gamma(M_L) \setminus M_R$, whose cardinality is only $|M|/\gamma$. Now suppose that M is chosen at random from among a large set of disjoint γ-focal matchings. If every client i has many outgoing edges, each belonging to a different one of these matchings, then i is unlikely to send its request along the outgoing edge which belongs to the chosen matching M, since it has no information about which matching was chosen other than the fact that it belongs to M_L.

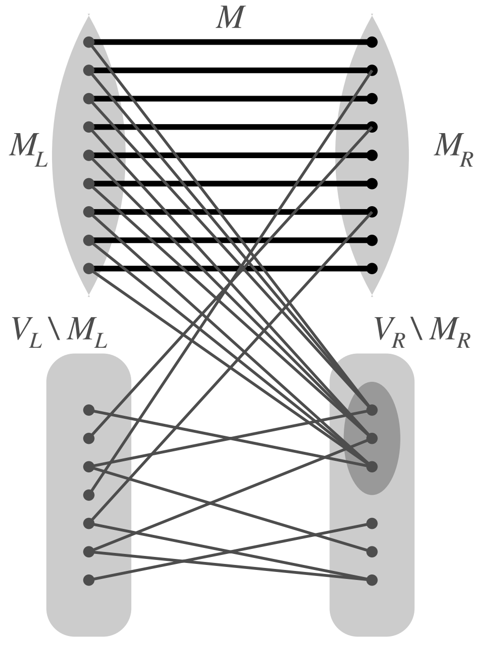


Fig. 1. *A γ-focal matching.*

4.1. Unicast lower bounds. Let $\mathcal{A}$ denote the set of all *fractional assignments* in G, i.e.,

$$\mathcal{A} = \left\{ f \,:\, E \to [0,1] \mid \forall i \in V_L \sum_{j \in \Gamma(i)} f(i,j) = 1 \right\}.$$

A set S of active clients may be represented by a function $D : V_L \to \{0,1\}$ mapping S to 1 and $V_L \setminus S$ to 0; we call this the *demand pattern* associated with S. Given a fractional assignment f, define the load on server j by

$$\ell(j) = \sum_{i \in \Gamma(j)} f(i,j) D(i)$$

and the throughput of f by

$$\theta(f) = \sum_{j \in V_R} \min\{\ell(j), 1\}.$$

We may think of a randomized assignment algorithm in the one-shot model as computing a function $A \,:\, \{0,1\}^{V_L} \times \Omega \to \mathcal{A}$, where Ω is a probability space encapsulating all the random bits (both shared and private) which the parties may use in their decision making. The fact that the assignment is *oblivious* (i.e., that clients must choose their own assignment without knowing which other clients are active) is captured by the following constraint: in the fractional assignment $f = A(D, \omega)$, for any edge (i,j), the value of $f(i,j)$ may depend only on $D(i)$ and ω. In other words, if $f' = A(D', \omega)$ and $D'(i) = D(i)$, then $f'(i,j) = f(i,j)$.

THEOREM 4. *Let G be a bipartite graph which is (d_L, d_R)-biregular; i.e., every $i \in V_L$ has degree d_L and every $j \in V_R$ has degree d_R. Assume all servers have unit capacity. If the edge set of G can be partitioned into γ-focal matchings of equal size, then the competitive ratio of any oblivious randomized fractional assignment algorithm for G is at least $\frac{1}{2}\min\{d_L, \gamma\}$.*

Proof. Let A be any oblivious randomized fractional assignment algorithm, and let $M^{(1)}, \ldots, M^{(s)}$ be a partition of E into γ-focal matchings of equal size k. Note that the number of edges satisfies $sk = |E| = d_L n$, whence

$$\frac{n}{s} = \frac{k}{d_L}.$$

Let the demand pattern $D \,:\, V_L \to \{0,1\}$ be defined by selecting a matching $M = M^{(r)}$ uniformly at random from $\{M^{(1)}, \ldots, M^{(s)}\}$, independently of the algorithm's random seed $\omega \in \Omega$, and setting $D(i) = 1$ if i is the left endpoint of an edge of M, 0 otherwise. As in Definition 1, let M_L, M_R denote the sets of left and right endpoints, respectively, of edges of M. The throughput of the assignment $f = A(D, \omega)$ satisfies

$$\begin{aligned}
\theta(f) &= \sum_{j \in M_R} \min\{\ell(j), 1\} + \sum_{j \in \Gamma(M_L) \setminus M_R} \min\{\ell(j), 1\} \\
&\leq \sum_{j \in M_R} \ell(j) + \sum_{j \in \Gamma(M_L) \setminus M_R} 1 \\
&\leq \sum_{e \in M} f(e) + k/\gamma.
\end{aligned}$$

Let D^* denote the demand pattern in which all clients are active, i.e., $D^*(i) = 1$ for all i, and let $f^* = A(D^*, \omega)$. By the definition of "oblivious," we have that $f(i,j) = f^*(i,j)$ for all $i \in M_L$. Hence

$$\theta(f) \leq k/\gamma + \sum_{e \in M} f^*(e).$$

Now let us take the expectation over the random choice of ω and M:

$$\begin{aligned}
\mathbf{E}[\theta(f)] &\leq \frac{k}{\gamma} + \sum_{e \in E} \Pr(e \in M)\mathbf{E}[f^*(e)] \\
&= \frac{k}{\gamma} + \frac{1}{s}\sum_{e \in E} \mathbf{E}[f^*(e)] \\
&= \frac{k}{\gamma} + \frac{1}{s}\mathbf{E}\left[\sum_{i \in V_L}\sum_{e \in \delta^+(i)} f^*(e)\right] \\
&\leq \frac{k}{\gamma} + \frac{n}{s} \quad = \quad \frac{k}{\gamma} + \frac{k}{d_L} \quad \leq \quad \frac{2k}{\min\{d_L, \gamma\}}.
\end{aligned}$$

The optimal assignment sends the k requests along the edges of M, thus achieving throughput k. Hence the competitive ratio of A is at least $\frac{1}{2}\min\{d_L, \gamma\}$, as claimed. □

Theorem 5. *There exists a bipartite graph G such that the competitive ratio of any oblivious randomized fractional assignment algorithm for G is at least $\sqrt{k}/2$, where k is the maximum throughput achievable in the given problem instance.*

Proof. The graph G is defined as follows. Given a positive integer d, let V_R be the set $\{1, 2, \ldots, d^2\}$, and let V_L be the set of all d-element subsets of V_R. Each such set $i \in V_L$ is joined by an edge to each of its elements $j \in V_R$. G is a biregular bipartite graph, with $d_L = d$ and $d_R = \binom{d^2-1}{d-1}$.

For each $(d-1)$-element subset $S \subseteq V_R$, let $M(S)$ be the matching containing, for each $j \in V_R \setminus S$, an edge from $i = S \cup \{j\}$ to j. Each such matching has size $d^2 - d + 1$, and each edge $(i,j) \in E$ belongs to exactly one such matching $M(S)$. (Namely, $S = i \setminus \{j\}$.)

We claim that each matching $M = M(S)$ is a d-focal matching. We have $M_R = V_R \setminus S$, and each $i \in M_L$ has one edge joining it to M_R (namely, the matching edge) and $d-1$ edges joining it to S. Thus G contains no edges between M_L and M_R other than the matching edges, and

$$|\Gamma(M_L) \setminus M_R| < |M|/d,$$

because the left-hand side is equal to $d-1$ while the right-hand side is equal to $d - 1 + 1/d$.

Applying Theorem 4, we find that the competitive ratio of any oblivious randomized fractional assignment algorithm is at least $d/2$, which is greater than $\sqrt{k}/2$ because $k = d^2 - d + 1$. □

Note that the proof of Theorem 5 also gives a lower bound of $\sqrt{m}/2$, where m is the number of servers. (We will see later that this bound is tight, up to a constant factor, in terms of m.) However, the number of clients in this example, n, is equal to $\binom{d^2}{d}$, and so the competitive ratio lower bound of $d/2$ translates only into a very weak lower bound of $\Omega(\log n/\log\log n)$ in terms of n. The following theorem demonstrates that a much stronger lower bound is possible.

THEOREM 6. *There exists a bipartite graph G such that the competitive ratio of any oblivious randomized fractional assignment algorithm for G is $\Omega(n^{0.103})$.*

Proof. The construction of the graph G in this case is quite complicated. For a positive integer d, let X be the ring $(\mathbb{F}_2)^d$, i.e., the Cartesian product of d copies of the field $\mathbb{F}_2 = \{0,1\}$. Considering X as a vector space over $\mathbb{F}_2$, let Y be a linear subspace of dimension bd. (We will optimize the value of the parameter $b < 1$ later.) Let Z denote the set of all z in X such that zy is nonzero for all nonzero $y \in Y$. (If we identify elements of X with subsets of $\{1, 2, \ldots, d\}$, then the nonzero elements of Y constitute a set system and Z consists of all hitting sets for this set system.) We will want the complement, $X \setminus Z$, to be as small as possible. Here is a calculation which bounds the expected size of $X \setminus Z$ when Y is a *random* linear subspace of dimension bd. For any nonzero $y \in X$, the probability that it belongs to Y is $2^{(b-1)d}$, and the number of z such that $zy = 0$ is $2^{d-wt(y)}$, where $wt(y)$ denotes the Hamming weight of y. This means that an upper bound for the expected size of $X \setminus Z$ is given by

$$\begin{aligned}\sum_{y\in X,\, y\neq 0} 2^{(b-1)d}2^{d-wt(y)} &= 2^{bd}\sum_{y\in X,\, y\neq 0} 2^{-wt(y)}\\ &= 2^{bd}\sum_{j=1}^{d}\binom{d}{j}2^{-j}\\ &= 2^{bd}\left[(3/2)^d - 1\right]\\ &< (3\cdot 2^{b-1})^d.\end{aligned}$$

Henceforth we assume that we have chosen a specific linear subspace Y such that the cardinality of $X \setminus Z$ is at most $(3 \cdot 2^{b-1})^d$. Later, when we specify the value of b, it will be the case that $3 \cdot 2^{b-1} = \sqrt{3} \cdot (1 + o(1))$, and so the fraction of elements of X which are not contained in Z is exponentially small in d.

The bipartite graph G is defined as follows. We put

$$\begin{aligned}V_L &= X \times Z,\\ V_R &= X,\\ E &= \{((x_L, z_L), x_R) \mid x_R - x_L = yz_L \text{ for some } y \in Y\}.\end{aligned}$$

By abuse of notation, we will write an edge e with left endpoint (x_L, z_L) and right endpoint x_R as an ordered triple $e = (x_L, z_L, x_R)$. Note that if $x_R - x_L = yz_L$ for some $y \in Y$, then this y is actually unique. (If $yz_L = y'z_L$, then $(y - y')z_L = 0$. Since $y - y' \in Y$ and $z \in Z$, this implies $y - y' = 0$.) We will refer to this unique value of y as the *type* of edge $e = (x_L, z_L, x_R)$.

We have seen that each $(x_L, z_L) \in V_L$ has exactly $|Y|$ outgoing edges, one of each type $y \in Y$. Similarly, each $x_R \in V_R$ has exactly $|Y \times Z|$ incoming edges. Given $(y, z) \in Y \times Z$, one may easily verify that there is one and only one edge of type y joining $X \times \{z\}$ to x_R, namely the edge $e = (x_R - yz, z, x_R)$. We have thus established that G is $(2^{bd}, 2^{bd} \cdot |Z|)$-biregular.

We must now specify a partition of the edge set into γ-focal matchings of equal size. For each pair (x, y), where $x \in X$, $y \in Y$, let

$$M(x, y) = \{(x + ((1 - y)z), z, x + z) \mid z \in Z\},$$

where "1" denotes the vector $(1, 1, 1, \dots, 1) \in X$. Note that $(x + ((1 - y)z), z, x + z)$ is a valid edge of type y in G, because $x + z = x + 1 \cdot z = x + ((1 - y)z) + yz$. The matchings $M(x, y)$ each have size $|Z|$. To see that every edge belongs to exactly one such matching, observe that if $e = (x_L, z_L, x_R)$ with $x_R - x_L = yz_L$, then e belongs to $M(x_R - z_L, y)$. There can be no other $M(x', y')$ containing e, since y' must equal the type of e and x' must equal $x_R - z_L$ in order for e to belong to $M(x', y')$.

Next, we wish to see that each such matching $M = M(x, y)$ is γ-focal for a reasonably large (i.e., exponential in d) value of γ. To do so, we will first show that every edge between M_L and the set $M_R = x + Z = \{x + z \mid z \in Z\}$ belongs to M. Let $e = (x_L, z_L, x_R)$ be such an edge, with $x_R - x_L = y'z_L$ for some $y' \in Y$. Since $(x_L, z_L) \in M_L$, we have $x_L = x + (1 - y)z_L$, whence $x_R = x + (1 + y' - y)z_L$. If $y' = y$, then $e \in M$. If $y' \neq y$, then we use the fact that every element w of the ring X satisfies $w(1 - w) = 0$. Applying this with $w = y - y'$, we see that $(y - y')(1 + y' - y)z_L = 0$. As $y - y'$ is a nonzero element of Y, we may conclude that $(1 + y' - y)z_L \notin Z$, whence $x_R \notin x + Z$. Finally, observe that

$$|\Gamma(M_L) \setminus M_R| \leq |V_R \setminus M_R| = |X \setminus (x + Z)| \leq (3 \cdot 2^{b-1})^d.$$

Recalling that $|M| = |Z| = (1 - o(1))2^d$, we see that M is γ-focal with

$$\gamma = (1 - o(1)) \left(2^{2-b}/3\right)^d.$$

Applying Theorem 4, we find that no oblivious randomized fractional assignment algorithm achieves a competitive ratio better than

$$\frac{1}{2}\min\{d_L, \gamma\} = \frac{1}{2}\min\left\{2^{bd}, \left(\frac{2^{2-b}}{3}\right)^d (1 - o(1))\right\}.$$

This approximately maximized when $2^b = 2^{2-b}/3$, i.e., when $b = 1 - \frac{1}{2}\log_2(3) = 0.2075\dots$. (Of course, b must be rounded to the nearest multiple of $1/d$, since bd, the dimension of the vector space Y, must be an integer.) Recalling that $n = |X \times Z| < 2^{2d}$, we see that the lower bound of $\Omega(2^{bd})$ on competitive ratio may be expressed as $\Omega(n^{b/2}) = \Omega(n^{0.103})$. □

4.2. Multicast lower bounds. Proving lower bounds in the multicast model is slightly more difficult than in the unicast model, because clients may broadcast their request to every adjacent server if they wish. If the set of active clients is equal to M_L for some γ-focal matching M, and each client chooses to broadcast its request to all adjacent servers, then each server in M_R will receive exactly one request and will satisfy it, leading to a throughput of $|M|$, the optimum throughput for the designated set of active clients. Nevertheless, it is possible to use γ-focal matchings to prove lower bounds in the multicast model by combining them with another device which we call a *smokescreen*. A smokescreen is simply a random set of clients whose size is small relative to the size of the matching and whose purpose is to confuse the servers in M_R by making it difficult for them to distinguish which incoming request is coming from M_L.

We will begin by formalizing the class of protocols which we will be considering. We will assume once again that there is a probability space Ω encapsulating the random bits (both shared and private) available to the parties in their computation. There is also a (not necessarily finite) message space MSG encapsulating all the messages that clients may send to servers. A protocol is specified by a communication function

$$A_i : \{0,1\} \times \Omega \to \mathsf{MSG}^{d(i)}$$

for each client i and a decision function

$$B_j : \mathsf{MSG}^{d(j)} \times \Omega \to \Gamma(j)$$

for each server j. The value of $A_i(D,\omega)$ specifies the $d(i)$-tuple of messages which i will send on its outgoing edges if its demand is D and the random seed is ω. The value of $B_j(m_1, m_2, \ldots, m_{d(j)}, \omega)$ specifies which client's request will be served by j if the random seed is ω and j receives messages $m_1, m_2, \ldots, m_{d(j)}$ on its incoming edges. We will call such a protocol $\{A_i, B_j\}$ an *oblivious assignment protocol for G in the multicast model*.

Without loss of generality we may assume that $\mathsf{MSG} = \{0,1\}$ and that each communication function A_i is simply the function $A_i(D,\omega) = D$. In other words, each client simply informs all adjacent servers whether or not it is active. This assumption is without loss of generality because for any other protocol $\hat{\mathcal{P}} = \{\hat{A}_i, \hat{B}_j\}$, we can construct a protocol $\mathcal{P} = \{A_i, B_j\}$ with A_i defined as above and with B_j defined as follows. For each client $i \in \Gamma(j)$, $B_j(m_1, \ldots, m_{d(j)}, \omega)$ simulates $\hat{A}_i(m_i, \omega)$ to determine what message $\hat{m}_i$ would have been sent from i to j under the protocol $\mathcal{P}$, and it then computes $\hat{B}_j(\hat{m}_1, \ldots, \hat{m}_{d(j)}, \omega)$ to determine what request it would have satisfied. This new protocol $\mathcal{P}$ has precisely the same outcome as $\hat{\mathcal{P}}$.

Based on this reduction, we will assume from now on that each server's decision function is a mapping $B_j : \{0,1\}^{\Gamma(j)} \times \Omega \to \Gamma(j)$ which chooses, for each subset $S \subseteq \Gamma(j)$, a random element $B_j(S,\omega) \in \Gamma(j)$ determined by the random seed ω. The notion that servers have difficulty distinguishing elements of M_L from elements of the smokescreen is captured by the following lemma.

LEMMA 1. *Let Γ be a set of d elements, and consider any function $B : 2^\Gamma \to \Gamma$. Suppose a random element $i \in \Gamma$ is sampled according to the uniform distribution, and a random set $S \subseteq \Gamma \setminus \{i\}$ is sampled by choosing each element independently with probability p. Then* $\Pr(B(S \cup \{i\}) = i) = O\left(\frac{1}{pd}\right)$.

Proof. For any nonempty set $T \subseteq \Gamma$ of cardinality t, we have

$$\Pr(B(T) = i \parallel S \cup \{i\} = T) = \begin{cases} \frac{1}{t} & \text{if } B(T) \in T, \\ 0 & \text{otherwise.} \end{cases}$$

This is obvious if $B(T) \notin T$. Assuming $B(T) \in T$, it holds because for every element $i_0 \in T$,

$$\Pr(i = i_0 \wedge S = T \setminus \{i_0\}) = \frac{1}{d} \cdot p^{t-1} \cdot (1-p)^{d-t}.$$

Denoting this probability by p_0, we have

$$\Pr(S \cup \{i\} = T) = \sum_{i_0 \in T} \Pr(i = i_0 \wedge S = T \setminus \{i_0\}) = tp_0$$

and

$$\Pr(B(T) = i \,\|\, S \cup \{i\} = T) = \frac{\Pr(i = B(T) \,\wedge\, S = T \setminus \{B(T)\})}{\Pr(S \cup \{i\} = T)} = \frac{p_0}{tp_0} = \frac{1}{t}.$$

Summing over all t, we have

$$\begin{aligned}\Pr(B(S \cup \{i\}) = i) &= \sum_{t=1}^{d} \frac{1}{t} \cdot \Pr(|S \cup \{i\}| = t) \\ &\leq \Pr\left(|S| < \frac{p(d-1)}{2}\right) + \frac{2}{p(d-1)} \Pr\left(|S| \geq \frac{p(d-1)}{2}\right) \\ &< e^{-p(d-1)/8} + \frac{2}{p(d-1)} = O\left(\frac{1}{pd}\right),\end{aligned}$$

where the last line follows from the Chernoff bound [17] and from the fact that the expectation of $|S|$ is $p(d-1)$. □

THEOREM 7. *Let G be a bipartite graph which is (d_L, d_R)-biregular. If the edge set of G can be partitioned into γ-focal matchings of size $k = \Omega(m)$, then the competitive ratio of any oblivious assignment protocol for G in the multicast model is $\Omega(\min\{\gamma, \sqrt{d_L}\})$.*

Proof. Let $M^{(1)}, \ldots, M^{(s)}$ be a partition of the edge set into γ-focal matchings of size k, and let the set of active clients S be defined as follows. Every client $i \in M_L$ belongs to S, and in addition, every $i \in V_L \backslash M_L$ joins S independently with probability $p = \sqrt{\frac{m}{d_R n}}$. The set $Q = S \setminus M_L$ is referred to as the *smokescreen.*

In the discussion preceding Lemma 1, we argued that one can assume without loss of generality that the protocol operates as follows: each client broadcasts its request to all adjacent servers; each server j receives requests from a set $T_j \subseteq \Gamma(j)$ and chooses which request to satisfy by computing a function $B_j(T_j, \omega)$ which depends on T_j and the (shared) random seed ω.

Since each client in Q and each server in $\Gamma(M_L) \setminus M_R$ contributes at most one unit of throughput, we have the following bound on the expected total throughput θ (where the expectation is over the random choice of S as well as the random seed ω):

$$\begin{aligned}\theta &\leq \mathbf{E}\left(|Q|\right) + \ \mathbf{E}\left(|\Gamma(M_L) \setminus M_R|\right) + \sum_{j \in V_R} \Pr(j \in M_R \,\wedge\, B_j(T_j, \omega) \in M_L) \\ &\leq pn + k/\gamma + \sum_{j \in V_R} \Pr(B_j(T_j, \omega) \in M_L \,\|\, j \in M_R).\end{aligned}$$

We may bound $\Pr(B_j(T_j, \omega) \in M_L \,\|\, j \in M_R)$ using Lemma 1. The key observation is that, conditional on the event $j \in M_R$, the set of active clients adjacent to j consists of one element i of M_L, uniformly distributed in $\Gamma(j)$, as well as a random subset of $\Gamma(j) \setminus \{i\}$ sampled by including each element independently with probability p. Thus

$$\Pr(B_j(T_j, \omega) \in M_L \,\|\, j \in M_R) = O\left(\frac{1}{pd_R}\right) = O\left(\sqrt{\frac{n}{md_R}}\right) = O\left(\sqrt{\frac{1}{d_L}}\right),$$

where the last step follows from the fact that $md_R = |E| = nd_L$. We are assuming

$k = \Omega(m)$, and so

$$\begin{aligned}\theta &\le pn + \frac{k}{\gamma} + O\left(m\sqrt{\frac{1}{d_L}}\right),\\ \theta/k &\le O\left(\frac{pn}{m} + \frac{1}{\gamma} + \sqrt{\frac{1}{d_L}}\right)\\ &= O\left(\sqrt{\frac{n}{md_R}} + \frac{1}{\gamma} + \sqrt{\frac{1}{d_L}}\right)\\ &= O\left(\frac{1}{\gamma} + \sqrt{\frac{1}{d_L}}\right)\\ &= O\left(\max\left\{\frac{1}{\gamma}, \sqrt{\frac{1}{d_L}}\right\}\right),\end{aligned}$$

and the competitive ratio k/θ is $\Omega(\min\{\gamma, \sqrt{d_L}\})$. □

THEOREM 8. *There exists a graph G such that the competitive ratio of any oblivious assignment protocol for G in the multicast model is $\Omega(k^{1/3})$.*

Proof. For an arbitrary positive integer d, let $V_R = \{1, 2, \ldots, d^3\}$, and let V_L be the set of all d^2-element subsets of V_R. Define the edge set by joining such a set i to an element $j \in V_R$ if j is an element of i, as in the proof of Theorem 5. As in that proof, the edge set may be partitioned into matchings $M(S)$, where S runs over all (d^2-1)-subsets of V_R and $M(S)$ is the matching containing, for each $j \in V_R \setminus S$, the edge from $i = S \cup \{j\}$ to j. Each such matching has size $k = d^3 - d^2 + 1$, satisfies $|\Gamma(M_L) \setminus M_R| = |S| = d^2 - 1$, and has the property that G contains no edges between M_L and M_R other than the edges of $M(S)$. Thus $M(S)$ is a $(d-1)$-focal matching for each S. The matchings $M(S)$ also satisfy $|M(S)| = \Omega(m)$ since $m = d^3$. We may thus apply Theorem 7 with $\gamma = d - 1 = \Omega(k^{1/3})$ and $\sqrt{d_L} = d = \Omega(k^{1/3})$ to obtain the desired lower bound. □

As above, the proof of Theorem 8 also establishes a lower bound of $\Omega(m^{1/3})$ on the competitive ratio of the optimal assignment protocol in the multicast model, and we will later see a matching upper bound. However, as before, this graph gives us only a very weak lower bound, $\Omega(\log n/\log\log n)$, in terms of n. For a polynomial lower bound in terms of n, we may use the same construction as was used in Theorem 6.

THEOREM 9. *There exists a graph G such that the competitive ratio of any oblivious assignment protocol for G in the multicast model is $\Omega(n^{0.069})$.*

Proof. The graph G is defined by the same construction as in the proof of Theorem 6, but this time we choose b by rounding off $(4/3) - (2/3) \cdot \log_2(3) = 0.27669\ldots$ to the nearest multiple of $1/d$. (Note that this value of b still satisfies $3 \cdot 2^{b-1} < 1$.) We have already proved that the edge set of G may be partitioned into γ-focal matchings of size $k = |Z|$. Here $m = 2^d$ and $|Z| \ge 2^d - (3 \cdot 2^{b-1})^d = (1 - o(1))m$, and so $k = \Omega(m)$ as required by Theorem 7. Recall that for this graph G,

$$\begin{aligned}\gamma &= (1 - o(1))\left(2^{2-b}/3\right)^d,\\ d_L &= 2^{bd}.\end{aligned}$$

We have chosen b so that $2^{b/2} = \left(1 + O\left(\frac{1}{d}\right)\right) 2^{2-b}/3$, and so $\sqrt{d_L}$ and γ are equal up to constant factors, and the competitive ratio of any oblivious assignment protocol is $\Omega(\sqrt{d_L}) = \Omega\left(2^{bd/2}\right)$. Recalling that $n = 2^{2d}$, this means the competitive ratio is $\Omega\left(n^{b/4}\right) = \Omega(n^{0.069})$. □

5. Algorithm for the unicast model. In this section we present an algorithm which is $O(\sqrt{k})$-competitive, where k denotes the maximum throughput achievable for the given demand pattern. We will initially work in the fractional assignment model. Later we will show that a simple randomized rounding of the fractional assignment yields an integral assignment with the same expected throughput, up to a constant factor.

THEOREM 10. *There exists an oblivious fractional assignment algorithm which is $O(\sqrt{k})$-competitive with the optimum fractional assignment for every demand pattern D.*

Proof. The oblivious fractional assignment algorithm is extremely simple. Each active client i sends $\frac{1}{d(i)}$ units of flow into each of its outgoing edges; each inactive client sends zero flow.

For a server j, recall that the load $\ell(j)$ is defined as the sum of the flows on all incoming edges. With the flow defined according to the algorithm specified, let Φ be the set of full servers, i.e., those with $\ell(j) \geq 1$. Let $\phi = |\Phi|$. We consider two cases. If $\phi \geq \sqrt{k}$, then the algorithm's throughput is at least $\sqrt{k}$ and we are done.

Now consider the case in which $\phi < \sqrt{k}$. Let A be the set of active clients i with $\Gamma(i) \subseteq \Phi$, and let B be the set of all other active clients. Note that $k \leq |\Phi| + |B|$, since every unit of flow in the optimal assignment passes through Φ or B. Our algorithm achieves a throughput of 1 from each server in Φ and a throughput of $\ell(j)$ from each server $j \in V_R \setminus \Phi$. Therefore, to finish proving the theorem it suffices to show that

$$\sum_{j \in V_R \setminus \Phi} \ell(j) \geq \frac{|B|}{\lceil \sqrt{k} \rceil}. \tag{5.1}$$

To do so, we will show that each client $i \in B$ contributes at least $1/\lceil \sqrt{k} \rceil$ to the left-hand side of (5.1). Note that each $i \in B$ has at least $\max\{1, d(i) - \phi\}$ neighbors which are not in Φ, and so i contributes at least $\max\{1/d(i), 1 - \phi/d(i)\}$ to the left-hand side of (5.1). If $d(i) < \lceil \sqrt{k} \rceil$, then $1/d(i) \geq 1/\lceil \sqrt{k} \rceil$. If $d(i) \geq \lceil \sqrt{k} \rceil$, then using the fact that $\phi \leq \lceil \sqrt{k} \rceil - 1$ we obtain

$$1 - \frac{\phi}{d(i)} \geq 1 - \frac{\lceil \sqrt{k} \rceil - 1}{\lceil \sqrt{k} \rceil} = \frac{1}{\lceil \sqrt{k} \rceil},$$

as desired. □

Theorem 5 demonstrates that no algorithm can achieve a better competitive ratio in terms of k than our simple algorithm, up to constant factors. An obvious corollary of Theorem 10 is that our algorithm's competitive ratio, in terms of n, is $O(\sqrt{n})$. This bound is tight in terms of n *for our algorithm*; i.e., there exist instances for which the algorithm's throughput is $O(k/\sqrt{n})$.[1] We do not know if there exists an algorithm achieving a better competitive ratio in terms of n; the best known lower bound is the one specified in Theorem 6.

5.1. Rounding fractional to integral assignments. We wish to demonstrate that for any oblivious fractional assignment algorithm A achieving competitive ratio

[1] Consider sets A, B, and C, where $|A| = n$, $|B| = n$, and $|C| = \sqrt{n}$. Let $V_L = A$ and $V_R = B \cup C$. The edge set of graph G consists of a perfect matching joining A to B and a complete bipartite subgraph joining A to C. In this example each client has degree at least $\sqrt{n}$. Now if the adversary chooses A as the set of active clients, then the optimum throughput, k, is equal to n, while our algorithm's throughput is only $O(\sqrt{n})$.

R, there is a randomized integral assignment algorithm A' achieving competitive ratio $O(R)$. If f is the fractional assignment computed by A for a given demand pattern, let A' select a random integral assignment as follows: each active client i chooses a random outgoing edge independently of the other clients' random choices, with $f(e)$ representing the probability of choosing edge e.

LEMMA 2. *Let $\theta(A), \theta(A')$ denote the throughput of A, A', respectively, on the given demand pattern. Then $\mathbf{E}(\theta(A')) \geq \left(1 - \frac{1}{e}\right)\theta(A)$.*

Proof. $\theta(A')$ is equal to the expected number of servers which receive at least one packet when an assignment is sampled at random according to A. Now

$$\begin{aligned} \Pr(j \text{ receives no packets}) &= \prod_{e \in \delta^-(j)} (1 - f(e)) \\ &< \prod_{e \in \delta^-(j)} e^{-f(e)} \\ &= \exp\left(-\sum_{e \in \delta^-(j)} f(e)\right) = e^{-\ell(j)}. \end{aligned} \tag{5.2}$$

If $\ell(j) \geq 1$, the right-hand side of (5.2) is at most $1/e$, and if $\ell(j) < 1$, the right-hand side is at most $1 - \left(1 - \frac{1}{e}\right)\ell(j)$, using the inequality $e^{-x} \leq 1 \cdot (1 - x) + \left(\frac{1}{e}\right) \cdot x$, which follows from the convexity of the function e^{-x}. Thus,

$$\Pr(j \text{ receives a packet}) \geq \left(1 - \frac{1}{e}\right) \min\{1, \ell(j)\}.$$

Summing over j, we obtain $\mathbf{E}(\theta(A')) \geq \left(1 - \frac{1}{e}\right)\theta(A)$. □

COROLLARY 1. *There exists a randomized oblivious integral assignment algorithm which is $O(\sqrt{k})$-competitive in expectation with the optimum assignment (i.e., maximum matching) for every demand pattern.*

6. Algorithm for the multicast model.

In this section, we describe a simple algorithm which achieves a competitive ratio of $O(k^{1/3})$ for the multicast model, where clients are allowed to send their request to more than one server, and a server may select any one of the requests it receives and satisfy this request. The algorithm requires no shared random bits, nor does it require the parties to know the structure of the graph G. The clients need only know which servers are adjacent to them, and the servers need only know the degrees of the adjacent active clients. (If necessary, the active clients may communicate this information in their request headers.)

THEOREM 11. *There exists an oblivious assignment protocol in the multicast model which is $O(k^{1/3})$-competitive with the optimum assignment (i.e., maximum matching) for every demand pattern.*

Proof. The algorithm is as follows. Each client broadcasts its request to all adjacent servers. If i is a client whose degree in the bipartite graph is $d(i)$, then a server receiving a request from i assigns weight $\frac{1}{d(i)}$ to this request. After receiving all requests, a server chooses to satisfy a random request with probability proportional to its weight.

For a server j, define its weight $w(j)$ to be the sum of the weights of all requests it receives. Let M be a specific maximum matching from the set of active clients to V_R; as usual we denote the size of this matching by k. For every edge $e = (i, j)$ in M, at least one of the following must hold:

1. $d(i)w(j) \leq k^{1/3}$.
2. $w(j) > k^{-1/3}$.
3. $d(i) > k^{2/3}$.

Thus one of the three possibilities is applicable to at least $k/3$ of the edges in M. We deal with them case-by-case.

In case 1, for each matching edge $e = (i, j)$ satisfying (1), we have

$$\Pr(j \text{ selects the request from } i) = (1/d(i))/w(j) = 1/(d(i)w(j)) \geq k^{-1/3}.$$

There are $k/3$ such edges; each has at least a $k^{-1/3}$ chance of being satisfied, and each of them corresponds to a distinct client. Hence the expected number of satisfied clients is $\Omega(k^{2/3})$ as desired.

In case 2, let S denote the set of servers which are right endpoints of matching edges satisfying (2). By assumption, there are $\Omega(k)$ such servers. The fact that they satisfy $\Omega(k^{2/3})$ distinct requests, in expectation, is a consequence of the following lemma, which we also use for case 3.

LEMMA 3. *For any real number* $0 < r \leq 1$, *let* S *denote the set of servers of weight at least* r. *The expected number of distinct requests satisfied by the servers in* S *is at least* $\frac{r}{e}|S|$.

Proof. For each server j in S, flip an independent coin and color server j red with probability r. Consider the following two events:

$E1$: j is colored red.

$E2$: The client i whose request was satisfied by j did not have its request satisfied by any red server other than j.

It is clear that $E1$ and $E2$ are independent. ($E1$ depends only on j's choice of color, $E2$ depends only on j's choice of job and on the random choices made by other servers.) The probability of $E1$ is r. We claim that the probability of $E2$ is at least $1/e$. To see this, let $d = d(i)$. For each element $j' \in S \setminus \{j\}$ adjacent to i, the probability that j' satisfied i's request is at most $\frac{1}{d(i)r}$, and the probability that it was colored red is r, and so there is at most a $1/d(i)$ chance that j' was colored red *and* satisfied i's request. Thus the probability that j' is not a red server satisfying i's request is $\geq 1 - 1/d(i)$. Multiplying at most $d(i) - 1$ such terms together, we get a probability which is at least $1/e$.

Thus the expected number of elements of S satisfying $E1$ and $E2$ is at least $(r/e)|S|$. No client can be satisfied by more than one such server, and so altogether the expected number of distinct clients satisfied by S is at least $(r/e)|S|$. □

Finally, we address case 3. Partition the servers into two sets, A and B, where A consists of all servers whose weight is at least 1, and all others belong to B. Let X denote the set of clients i which satisfy

1. i is the left endpoint of an edge in the matching M;
2. $d(i) \geq k^{2/3}$.

By hypothesis, $|X|$ is at least $k/3$. For each server j, let $w'(j)$ denote the total weight of the requests it receives from elements of X. The sum of $w'(j)$ over all servers j is simply $|X|$ (since each client contributes exactly one unit of weight, in total), and hence one of the following subcases applies:

3.1. $\sum_{j \in A} w'(j) \geq |X|/2 \geq k/6$.

3.2. $\sum_{j \in B} w'(j) \geq |X|/2 \geq k/6$.

We handle the two subcases separately. For case 3.1, note that $w'(j)$ is bounded above by $k^{1/3}$, because j is adjacent to at most k elements of X, and each of them contributes at most $k^{-2/3}$ units of weight to $w'(j)$. So in order for case 3.1 to hold, it must be the case that $|A| \geq k^{2/3}/6$. Applying the lemma above with $r = 1$, we find that the expected number of distinct clients satisfied by servers in A is $\Omega(k^{2/3})$ as desired. For case 3.2, at least 3/4 of the clients in X have at least 1/3 of their neighbors in B. (Otherwise these clients would contribute less than $|X|/4$ to the sum on the left-hand side of case 3.2, and the remaining $|X|/4$ clients would contribute at most $|X|/4$ to that sum.) For a client with 1/3 of its neighbors in B, the probability of its request being satisfied is bounded below by a constant, namely $1 - e^{-1/3}$. To see this, let i be such a client and j any neighbor of i in B. The probability that j satisfies i's request is $\frac{1}{d(i)w(j)} \geq \frac{1}{d(i)}$, and so the probability that j does not satisfy i's job is at most $1 - 1/d(i)$. Multiplying at least $d(i)/3$ such terms together, we get a failure probability which is less than $e^{-1/3}$. So, in case 3.2, we find that the expected number of elements of X whose request is satisfied by an element of B is at least $(1 - e^{-1/3}) \cdot (3/4) \cdot |X| = \Omega(k)$, which easily beats the required $\Omega(k^{2/3})$ bound. □

Theorem 8 demonstrates that no algorithm can achieve a better competitive ratio in terms of k than our algorithm, up to constant factors. An obvious corollary of Theorem 11 is that our algorithm's competitive ratio, in terms of n, is $O(n^{1/3})$. This bound is tight in terms of n *for our algorithm*; i.e., there exist instances for which the algorithm's throughput is $O(k/n^{1/3})$.[2] We do not know if there exists an algorithm achieving a better competitive ratio in terms of n; the best known lower bound is the one specified in Theorem 9.

7. Restricted adversary model.

Returning from the setting of one-shot (oblivious) algorithms to the online setting, we now consider online fractional assignment algorithms for a sequence of demand patterns $D_t : V_L \to \{0, 1\}$, which may be adversarially specified subject to the restriction that when a client becomes active, it remains active for the next r rounds, where r is a positive integer known to all clients. (As always, we refer to a client i as active at time t if $D_t(i) = 1$, inactive otherwise.) We do not assume that any of the parties know the structure of the graph G; the only requirement is that clients should know the set of adjacent servers, and they should have common knowledge of a number Δ which is an upper bound on the degree of any client. (Such an upper bound is often easy to obtain. For example, if the number of servers m is common knowledge, this is a suitable value for Δ.)

Unlike previous sections, which assumed each server has unit capacity, we assume here that each server j has a nonnegative capacity c_j. No upper bound on c_j is assumed, but the capacities are assumed to remain constant over time. The throughput of an assignment is defined to be the sum, over all servers j, of $\min\{\ell(j), c_j\}$, where $\ell(j)$ as always denotes the load on server j.

Our algorithm runs in a series of synchronous, concurrent rounds. In each round, each client assigns load fractionally among the adjacent servers. (As in Lemma 2, such a fractional assignment may be converted into an integral assignment by randomized rounding, decreasing the expected throughput by only a constant factor.) Each server

[2] Consider a bipartite graph G whose left vertices are partitioned into two sets A, B and whose right vertices are partitioned into two sets C, D, such that $|A| = k$, $|B| = k^{2/3}$, $|C| = k$, $|D| = k^{2/3}$. A and C are joined by a perfect matching, B and C are joined by a complete bipartite graph, A and D are joined by a complete bipartite graph, and there are no edges from B to D. If each client is active, then it is an exercise to check that the algorithm specified above satisfies only $O(k^{2/3}) = O(k/n^{1/3})$ distinct jobs in expectation.

sums the assigned loads and reports its load/capacity ratio back to the adjacent clients. This is the only communication in either direction.

7.1. Algorithm. The algorithm divides time into windows of length $\lceil r/2 \rceil$. Each active client maintains a fractional assignment of load on its outgoing edges. When a client of degree d becomes active, it waits for the start of the next window and then initializes its fractional assignment by sending $1/d$ units of flow on each outgoing edge. While a client remains active, it updates its fractional assignment f at the end of each round, using the feedback from the adjacent servers as follows. Let $\alpha = (2\Delta)^{6/r}$. A server is defined to be "undersupplied," "comfortable," or "oversupplied," according to whether the corresponding server's load/capacity ratio is less than $1/\alpha$, is in the interval $[1/\alpha, 1]$, or is greater than 1, respectively. We will refer to edges as undersupplied, comfortable, or oversupplied according to the status of the corresponding server, and for a client i we will denote the total flow on undersupplied, comfortable, and oversupplied edges by $f_u(i), f_c(i), f_o(i)$, respectively. A client i with $d_o(i)$ oversupplied outgoing edges is called "unhappy" if

$$0 < (\alpha - 1) f_u(i) < f_o(i) - \frac{d_o(i)}{2\Delta},$$

"happy" otherwise. A happy client retains the same flow distribution in the next round. An unhappy client redistributes flow from the oversupplied edges to the undersupplied ones, so as to multiply the amount of flow on each undersupplied edge by α. This requires increasing the total flow on the undersupplied edges by $(\alpha - 1) f_u(i)$. After the redistribution the total amount of flow remaining on the edges which were oversupplied before redistribution will be $f_o(i) - (\alpha - 1) f_u(i)$, and our definition of an unhappy client ensures that this number is greater than $d_o(i)/2\Delta$. Thus it is always possible to arrange the redistribution in such a way that each edge which was oversupplied before redistribution has at least $1/2\Delta$ units of flow after redistribution. Each unhappy client chooses an arbitrary redistribution which satisfies this constraint.

7.2. Analysis. In a time window W, call a client *eligible* if it is active in every round belonging to W. Define a modified sequence of demands $\hat{D}_t(i)$ by specifying that $\hat{D}_t(i) = 1$ if i is eligible in the window containing round t, 0 otherwise. The analysis of the algorithm depends on proving that it is $O(\alpha)$-competitive with the throughput of the optimum sequence of assignments for the *modified* demands. The following lemma explains why this is sufficient.

LEMMA 4. *Let $\theta, \hat{\theta}$ denote the throughput of the optimum sequence of assignments for the original demands and the modified demands, respectively. Then $\hat{\theta} \geq \theta/3$.*

Proof. Let $f_1, f_2, \ldots, f_T$ be a throughput-maximizing sequence of assignments for the original demands D_t. We may assume that each f_t assigns to server j a load $\ell_t(j)$ which is at most c_j. (If not, we may adjust f_t by reducing the flow on some of the incoming edges to server j without reducing the throughput.) Now construct a sequence of assignments $\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_T$ as follows. Initially, $\hat{f}_t = f_t/3$. For each client i which is active but not eligible at time t, it must be the case that either of the following holds:

- i became active during the window W containing t. If so, i is eligible in the next window, $W + 1$. Let $t' = t + \lceil r/2 \rceil$.
- i ceased to be active during W. If so, i is eligible in the preceding window, $W - 1$. Let $t' = t - \lceil r/2 \rceil$.

Now adjust $\hat{f}$ by changing $\hat{f}_{t'}(e)$ to $\hat{f}_t(e) + \hat{f}_{t'}(e)$ for each outgoing edge e from i and setting $\hat{f}_t(e)$ to zero. In this way, we obtain a sequence of assignments $\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_T$

such that the following hold:

- The outflow from ineligible clients is zero in each round.
- The outflow from an eligible client i is at most 1. (In the original assignments f_t, the outflow from i was at most 1 in each round. In $\hat{f}_t$, the outflow from i at time t is bounded above by the average outflow in rounds $t, t-\lceil r/2\rceil, t+\lceil r/2\rceil$ of the original assignment.)
- The inflow to a server j is at most c_j. (In the original assignments f_t, the inflow to j was at most c_j in each round. In $\hat{f}_t$, the inflow to j at time t is bounded above by the average inflow in rounds $t, t-\lceil r/2\rceil, t+\lceil r/2\rceil$ of the original assignment.)
- The throughput is $\theta/3$. (We initialized $\hat{f}_t$ to $f_t/3$, and we subsequently shifted flow without changing the combined throughput.)

By definition, the throughput of $\hat{f}_1, \ldots, \hat{f}_T$ is at most $\hat{\theta}$. Thus $\hat{\theta} \geq \theta/3$. □

THEOREM 12. *The algorithm specified in section* 7.1 *is* $O(\Delta^{6/r})$*-competitive.*

Proof. For a time window W, let $\hat{\theta}(W)$ be the optimum throughput achievable by an assignment of the eligible clients only. By the preceding lemma, we know that it is sufficient to prove that the algorithm's throughput during W is $\Omega(\hat{\theta}(W)/\alpha)$. For the remainder of the analysis, we will limit our attention to the time rounds which belong to W.

First, we note that the load on a server cannot increase by a factor of more than α in any round, because the load on each edge cannot increase by a factor of more than α. If a server is comfortable, the load on its incoming edges does not change at all. Therefore a server may not become oversupplied in the next round unless it was already oversupplied in the current round.

Second, we note that for an edge $e = (i, j)$, the flow $f(e)$ does not increase while j is oversupplied; if j ever ceases to be oversupplied, in each subsequent round $f(e)$ either increases by a factor of α or remains the same. Moreover, the number of rounds in which $f(e)$ increases is at most $r/6$ because $\alpha^{r/6} = 2\Delta$, and $f(e)$ is never less than $\frac{1}{2\Delta}$ and never more than 1.

For each edge $e = (i, j)$ in each round t, one of the following applies:

1. i was happy in round t.
2. j was not undersupplied in round t.
3. The load on e increased by a factor of α at the end of round t.

We have already argued that the third case applies to at most $r/6$ of the $\lceil r/2\rceil$ rounds in W. Therefore, either the first or the second case is satisfied by edge e in at least $r/6$ of the rounds in $t \in W$.

Call a client "satisfied" if it is happy in at least $r/6$ of the rounds in W; let X be the set of all such clients. Call a server "satisfied" if it is oversupplied or comfortable in at least $r/6$ rounds of W; let Y be the set of all such servers. Above, we have proven that every edge has either its left endpoint in X or its right endpoint in Y. Therefore, in the maximum-throughput flow, every unit of flow goes through either a satisfied client or a satisfied server, resulting in the bound

$$\frac{\hat{\theta}(W)}{\lceil r/2\rceil} \leq |X| + \sum_{j \in Y} c_j. \tag{7.1}$$

However, it follows from the definition of "satisfied" that the algorithm's throughput

θ satisfies

$$\frac{\theta}{r/6} \geq \max\left\{\frac{1}{2\alpha}|X|,\ \frac{1}{\alpha}\sum_{j\in Y} c_j\right\}. \tag{7.2}$$

The lower bound $(1/\alpha)\sum_j c_j$ is immediate from the fact that a server j which is not undersupplied has throughput at least c_j/α. The lower bound $(1/2\alpha)|X|$ may be derived as follows. If a client i is happy in round t, we have $(\alpha-1)f_u(i) \geq f_o(i) - \frac{1}{2}$, whence

$$\alpha f_u(i) + \alpha f_c(i) \geq (\alpha-1)f_u(i) + f_u(i) + f_c(i) \geq (f_o(i) + f_u(i) + f_c(i)) - \frac{1}{2} = \frac{1}{2}.$$

Every unit of flow which i sends to an undersupplied or comfortable server contributes to the throughput in round t. Therefore a happy client contributes at least $f_u(i) + f_c(i) \geq \frac{1}{2\alpha}$ units of throughput in round t, which justifies (7.2).

Finally, putting together (7.1), (7.2), we obtain $18\alpha\left(\frac{\lceil r/2\rceil}{r}\right)\theta \geq \hat{\theta}(W)$. □

8. Discussion and open problems.

In the one-shot model, we proved strong lower bounds on the competitive ratio of decentralized client-server load-balancing algorithms. However, there is still a significant gap between the lower and upper bounds expressed in terms of n, the number of clients. In the unicast model the lower and upper bounds are $\Omega(n^{0.103})$ and $O(n^{1/2})$, respectively. In the multicast model the lower and upper bounds are $\Omega(n^{0.069})$ and $O(n^{1/3})$, respectively. Closing these gaps would most likely require the development of interesting new techniques.

Also, and perhaps more importantly, we do not know of a graph parameter, defined for all bipartite graphs G, which characterizes (or closely approximates) the competitive ratio of the optimum assignment algorithm for G. In this paper we have shown that when the edge set can be partitioned into some number of γ-focal matchings, then it implies a lower bound on the competitive ratio of oblivious assignment algorithms for G. We do not know of any partial converse result asserting that a sufficiently large lower bound on the competitive ratio of oblivious assignment algorithms for G implies the existence of γ-focal matchings in G for some large value of γ.

We observed earlier that our lower bounds for the one-shot model easily imply equally strong lower bounds for the online model when the adversary is unrestricted. One way of circumventing these lower bounds is to impose restrictions on the adversary such as the minimum-activity-period restriction considered in section 7. Another interesting research direction is to try circumventing the online lower bounds by comparing the algorithm against a weaker benchmark than the omniscient algorithm which chooses the best client-server assignment in each period. For example, one could adopt the approach used in the literature on *regret minimization* in online learning theory. This entails comparing the algorithm's expected throughput against the maximum throughput achievable by a *fixed* client-server assignment which does not change over time.

Acknowledgment. We would like to thank the anonymous reviewers for their constructive comments and suggestions.

REFERENCES

[1] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts, *On-line routing of virtual circuits with applications to load balancing and machine scheduling*, J. ACM, 44 (1997), pp. 486–504.

[2] B. Awerbuch and Y. Azar, *Local optimization of global objectives: Competitive distributed deadlock resolution and resource allocation*, in Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science, 1994, pp. 240–249.

[3] B. Awerbuch, Y. Azar, and S. Plotkin, *Throughput competitive on-line routing*, in Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science, 1993, pp. 32–40.

[4] B. Awerbuch and T. Leighton, *A simple local-control approximation algorithm for multicommodity flow*, in Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science, 1993, pp. 459–468.

[5] B. Awerbuch and T. Leighton, *Improved approximation algorithms for the multi-commodity flow problem and local competitive routing in dynamic networks*, in Proceedings of the 26th Annual ACM Symposium on Theory of Computing, 1994, pp. 487–496.

[6] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Racke, *Optimal oblivious routing in polynomial time*, in Proceedings of the 35th Annual ACM Symposium on Theory of Computing, 2003, pp. 383–388.

[7] Y. Bartal, J. W. Byers, and D. Raz, *Global optimization using local information with applications to flow control*, in Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science, 1997, pp. 303–312.

[8] M. Bienkowski, M. Korzeniowski, and H. Räcke, *A practical algorithm for constructing oblivious routing schemes*, in Proceedings of the 15th Annual ACM Symposium on Parallel Algorithms and Architectures, 2003, pp. 24–33.

[9] E. Cohen, *Approximate max flow on small depth networks*, in Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science, 1992, pp. 648–658.

[10] N. Garg and J. Könemann, *Faster and simpler algorithms for multicommodity flow and other fractional packing problems*, in Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science, 1998, pp. 300–309.

[11] N. Garg and N. E. Young, *On-line end-to-end congestion control*, in Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002, pp. 303–312.

[12] A. V. Goldberg and R. E. Tarjan, *A new approach to the maximum-flow problem*, J. ACM, 35 (1988), pp. 921–940.

[13] M. Hajiaghayi, R. Kleinberg, T. Leighton, and H. Räcke, *Oblivious routing on node-capacitated and directed graphs*, in Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2005, pp. 782–790.

[14] C. Harrelson, K. Hildrum, and S. Rao, *A polynomial-time tree decomposition to minimize congestion*, in Proceedings of the 15th Annual ACM Symposium on Parallel Algorithms and Architectures, 2003, pp. 34–43.

[15] R. M. Karp, E. Upfal, and A. Wigderson, *Constructing a perfect matching is in Random NC*, Combinatorica, 6 (1986), pp. 35–48.

[16] M. Luby and N. Nisan, *A parallel approximation algorithm for positive linear programming*, in Proceedings of the 25th Annual ACM Symposium on Theory of Computing, 1993, pp. 448–457.

[17] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, New York, 1995.

[18] C. H. Papadimitriou and M. Yannakakis, *Linear programming without the matrix*, in Proceedings of the 25th Annual ACM Symposium on Theory of Computing, 1993, pp. 121–129.

[19] S. A. Plotkin, D. B. Shmoys, and É. Tardos, *Fast approximation algorithms for fractional packing and covering problems*, Math. Oper. Res., 20 (1995), pp. 257–301.

[20] H. Räcke, *Minimizing congestion in general networks*, in Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002, pp. 43–52.

[21] H. Räcke and A. Rosén, *Distributed online call control on general networks*, in Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2005, pp. 791–800.

SIAM J. COMPUT.
Vol. 37, No. 4, pp. 1280–1298

(t, k)-DIAGNOSABILITY OF MULTIPROCESSOR SYSTEMS WITH APPLICATIONS TO GRIDS AND TORI[*]

GUEY-YUN CHANG[†] AND GEN-HUEY CHEN[‡]

Abstract. (t, k)-diagnosis, which is a generalization of sequential diagnosis, requires at least k faulty processors identified and replaced (or repaired) in each iteration provided there are at most t faulty processors, where $t \geq k$. This paper suggests lower bounds on the degrees of (t, k)-diagnosability of multiprocessor systems under both the PMC and the MM* models. As a consequence, grids and tori of d dimensions are shown to be $(\Omega(N^{\frac{d}{d+1}}), \Omega(d))$-diagnosable and $(\Omega(N^{\frac{d}{d+1}}), \Omega(2d))$-diagnosable, respectively, where N is the number of processors.

Key words. diagnosability, diagnosis, MM* model, multiprocessor system, PMC model, sequential diagnosis, (t, k)-diagnosis

AMS subject classifications. 94C12, 68M15

DOI. 10.1137/06065043X

1. Introduction.

System-level diagnosis, which was introduced by Preparta, Metze, and Chien [27], aims to identify faulty processors in a multiprocessor system by analyzing the outcomes of available interprocessor tests. The problem has been extensively studied in the literature [3], [10], [14], [17], [27]. The *PMC (Preparta–Metze–Chien) model*, which was originally introduced in [27], is possibly the most well-studied model for system-level diagnosis. It assumes that each processor can test its neighboring processors and declare them fault-free or faulty. The test results are considered reliable if the processor is fault-free. Previous work on the PMC model can be found in [15], [18], [19], [20], [21], [28], [30].

In [29], Sengupta and Dahbura proposed another diagnosis model, called the *MM* model*, under which each processor has to test any two of its neighboring processors. The testing processor feeds the two tested processors with the same task and input and then compares their outputs. It is assumed that the outputs are identical if they are fault-free, and distinct otherwise. Only a fault-free testing processor can guarantee a reliable test result. The MM* model was also studied in [2], [16], [24], [31].

There are three fundamental diagnosis strategies, i.e., one-step diagnosis, sequential diagnosis, and (t, k)-diagnosis, for system-level diagnosis. In *one-step diagnosis*, all faulty processors are identified before replacement (or repair) is made. In *sequential diagnosis* and *(t, k)-diagnosis*, faulty processors are identified and replaced iteratively. Differently, in each iteration, at least one faulty processor is identified and replacd in sequential diagnosis, whereas at least k faulty processors (or all faulty processors if fewer than k faulty processors remain) are identified and replaced in (t, k)-diagnosis provided there are at most t faulty processors, where $t \geq k$. Notice that one-step diagnosis (when $t = k$) and sequential diagnosis (when $k = 1$) are two instances of (t, k)-diagnosis.

[*] Received by the editors January 20, 2006; accepted for publication (in revised form) June 29, 2007; published electronically December 21, 2007.
http://www.siam.org/journals/sicomp/37-4/65043.html

[†] Department of Computer Science and Information Engineering, National Central University, Taoyuan, 32001, Taiwan (gychang@csie.ncu.edu.tw).

[‡] Department of Computer Science and Information Engineering, National Taiwan University, Taipei, 10617, Taiwan (ghchen@csie.ntu.edu.tw).

Suppose that S is a system with at most t faulty processors. Under one-step diagnosis, S is *one-step t-diagnosable* if all faulty processors can be identified [2], [11], [15], [16], [20], [21], [24], [25], [30], [31]. Similarly, under sequential diagnosis (or (t, k)-diagnosis), S is *sequentially t-diagnosable* (or *(t, k)-diagnosable*) if at least one (or at least k) faulty processor can be identified in each iteration [22], [23] (or [1]). If S is (t, k)-diagnosable, then S is also (t, k')-diagnosable for any $k' \le k$.

An algorithm that can identify all faulty processors in S is called a *diagnosis algorithm* for S. Under the PMC model, a one-step diagnosis algorithm for arbitrary systems was proposed in [14]. In [22], [23], sequential diagnosis algorithms were proposed and the following diagnosabilities were derived: sequentially $\Omega(N^{\frac{d}{d+1}})$-diagnosable for symmetric d-dimensional grids, sequentially $\Omega(\frac{N \log \log N}{\log N})$-diagnosable for hypercubes, sequentially $\Omega(\sqrt{N})$-diagnosable for cube-connected cycles, sequentially $\Omega(\sqrt{\frac{N}{k}})$-diagnosable for k-ary trees, and sequentially $\Omega(\sqrt{\frac{N}{\Delta}})$-diagnosable for arbitrary graphs, where N is the number of processors and Δ is the degree of the graph. On the other hand, under the MM* model, a one-step diagnosis algorithm was proposed in [29].

In this paper, we derive lower bounds on the degrees of (t, k)-diagnosability of multiprocessor systems under the PMC and the MM* models. As two applications, grids and tori of d dimensions are shown to be $(\Omega(N^{\frac{d}{d+1}}), \Omega(d))$-diagnosable and $(\Omega(N^{\frac{d}{d+1}}), \Omega(2d))$-diagnosable, respectively. In the next section, some necessary definitions and notation are introduced. In section 3, a fundamental result is derived, which is necessary to the computation of (t, k)-diagnosability of multiprocessor systems. Lower bounds on the degrees of (t, k)-diagnosability of multiprocessor systems under the PMC and the MM* models are suggested in section 4. Lower bounds on the degrees of (t, k)-diagnosability of grids and tori are derived in section 5. Finally, in section 6, we conclude this paper with some remarks.

2. Preliminaries.

Throughout this paper, the topology of a self-diagnosable system S is represented with a directed graph $G = (V, A)$, where each processor of S is uniquely denoted by a vertex in V and each link of S is denoted by a pair of opposite arcs, e.g., (u, v) and (v, u), in A (the two processors connected by the link are denoted by vertices u and v, respectively). In order to diagnose faults, a number of tests among processors need to be performed and the collection of all test results is referred to as a *syndrome*. For each $w \in V$, let $N(w)$ be the neighborhood of w, i.e., the set of vertices in G that is neighboring to w. For a subset V' of V, define $N(V') = \bigcup_{w \in V'} N(w) - V'$. A syndrome for S is formally defined below.

Definition 1. *A syndrome σ for a system S under the PMC model is the set $\{\sigma(u, v) : u, v \in V$ and $u \in N(v)\}$, where*

$$\sigma(u, v) = \begin{cases} 0 & \textit{if } v \textit{ is tested by } u \textit{ to be fault-free;} \\ 1 & \textit{if } v \textit{ is tested by } u \textit{ to be faulty.} \end{cases}$$

Definition 2. *A syndrome σ for a system S under the MM* model is the set $\{\sigma(u, v : w) : u, v, w \in V$ and $u, v \in N(w)\}$, where*

$$\sigma(u, v : w) = \begin{cases} 0 & \textit{if } w \textit{ declares that the outputs of testing } u \textit{ and } v \textit{ are identical;} \\ 1 & \textit{if } w \textit{ declares that the outputs of testing } u \textit{ and } v \textit{ are distinct.} \end{cases}$$

Since the test result initiated by a faulty processor is not predictable, more than one syndrome may be produced for S with faulty processors. Based on Definitions 1

Table 1
Invalidation rule of the PMC model.

Testing vertex	Tested vertex	Test result
fault-free	fault-free	0
fault-free	faulty	1
faulty	fault-free	0 or 1
faulty	faulty	0 or 1

Table 2
Invalidation rule of the MM model.*

Testing vertex	Tested vertex pair	Test result
fault-free	(fault-free, fault-free)	0
fault-free	(fault-free, faulty)	1
fault-free	(faulty, faulty)	1
faulty	(fault-free, fault-free)	0 or 1
faulty	(fault-free, faulty)	0 or 1
faulty	(faulty, faulty)	0 or 1

and 2, the invalidation rules of the PMC and MM* models are summarized in Tables 1 and 2, respectively.

Definition 3. *A system S with a fault set F is (t, k)-diagnosable if, given any syndrome σ for S, at least k faulty processors can be identified when $k < |F| \leq t$ and all faulty processors can be identified when $|F| \leq k$.*

The vertex set, denoted by Λ, of a strongly connected subgraph of G is called an *aggregate* if either all the vertices of Λ are fault-free and all the vertices of $N(\Lambda)$ are faulty or all the vertices of Λ are faulty and all the vertices of $N(\Lambda)$ are fault-free [10]. The indegree (outdegree) of a vertex u in G is the number of arcs entering (leaving) u. Since each vertex of G has indegree equal to outdegree, we simply use degree to denote either of them when there is no ambiguity. In the rest of this paper, we use Δ and δ to denote the maximal degree and minimal degree of G, respectively.

2.1. The PMC model. Let $\Upsilon_P(x)$ be the set of neighboring vertices of x that are tested by x as fault-free under the PMC model, i.e., $\Upsilon_P(x) = \{y : y \in N(x) \text{ and } \sigma(x, y) = 0\}$. Suppose that $(u_1, u_2, \ldots, u_m)$ is a directed path from vertex u_1 to vertex u_m in G. If $u_{i+1} \in \Upsilon_P(u_i)$ for all $1 \leq i \leq m-1$, then either u_1 is faulty or $u_1, u_2, \ldots, u_m$ are all fault-free. By $G_P^\dagger$ we denote the spanning subgraph of G whose arcs are all (u, v)'s with $v \in \Upsilon_P(u)$, i.e., $G_P^\dagger = (V_P^\dagger, A_P^\dagger)$, where $V_P^\dagger = V$ and $A_P^\dagger = \{(u, v) : (u, v) \in A \text{ and } v \in \Upsilon_P(u)\}$. In the rest of this paper, each strongly connected subgraph of $G_P^\dagger$ is referred to as a *P-component*. That is, if u and v are two distinct vertices belonging to the same P-component, there are two directed paths from u to v and from v to u in $G_P^\dagger$, respectively. Furthermore, for ease of subsequent discussion, we let each P-component denote only the vertex set of the representing subgraph.

Clearly, each P-component has all vertices fault-free or all vertices faulty. Moreover, each fault-free P-component is also a fault-free aggregate in G. For example, let G be a 4×4 grid, as depicted in Figure 1(a), where white (black) vertices are fault-free (faulty). There are four aggregates in G, two fault-free and two faulty, as shown in Figure 1(b). Assume that $\sigma(u, v) = 0$ if u and v are fault-free, and $\sigma(u, v) = 1$ otherwise. Then, $G_P^\dagger$ with respect to σ is shown in Figure 1(c), where an arc, say, from u to v, means $\sigma(u, v) = 0$. There are seven P-components contained in $G_P^\dagger$, as shown in Figure 1(d). It is observed from Figures 1(b) and 1(d) that each fault-free P-component is a fault-free aggregate in G.

Let C_P denote the set of all P-components in $G_P^\dagger$. When $|C_P| = 1$, either all vertices of G are fault-free or all vertices of G are faulty. So, we assume $|C_P| \geq 2$.

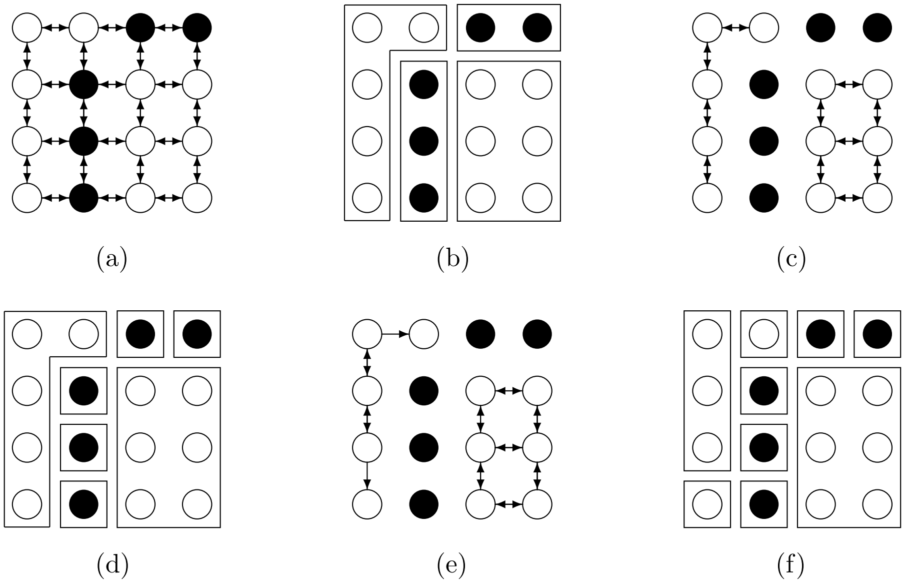


FIG. 1. *An example:* (a) G. (b) *Aggregates in* G. (c) $G_P^\dagger$. (d) *P-components in* $G_P^\dagger$. (e) $G_M^\dagger$. (f) *M-components in* $G_M^\dagger$.

2.2. The MM* model. When the MM* model is considered, the definitions of $\Upsilon_P(x)$, $G_P^\dagger$, and P-component should be adapted, in accordance with the difference between Definitions 1 and 2. We introduce the new notation $\Upsilon_M(x)$, $G_M^\dagger$ and M-component for the MM* model. Similarly, $\Upsilon_M(x) = \{y : y \in N(x) \text{ and } \sigma(y, z : x) = 0 \text{ for some } z \in N(x)\}$ is the set of neighboring vertices of x that are tested by x as fault-free under the MM* model. A directed path $(u_1, u_2, \ldots, u_m)$ from vertex u_1 to vertex u_m with $u_{i+1} \in \Upsilon_M(u_i)$ for all $1 \leq i \leq m-1$ means that either u_1 is faulty or $u_1, u_2, \ldots, u_m$ are all fault-free. $G_M^\dagger = (V_M^\dagger, A_M^\dagger)$, where $V_M^\dagger = V$ and $A_M^\dagger = \{(u, v) : (u, v) \in A \text{ and } v \in \Upsilon_M(u)\}$, is the spanning subgraph of G containing all arcs (u, v)'s with $v \in \Upsilon_M(u)$. An *M-component* is a strongly connected subgraph of $G_M^\dagger$. For every two distinct vertices u, v in the same M-component, there are two directed paths from u to v and from v to u in $G_M^\dagger$, respectively. We let each M-component denote only the vertex set of the representing subgraph.

Like P-components, each M-component has all vertices fault-free or all vertices faulty. But, differently, each fault-free aggregate in G may consist of one or more fault-free M-components. For example, consider the graph G of Figure 1(a) again. Assume that $\sigma(u, v; w) = 0$ if u, v, and w are all fault-free, and $\sigma(u, v; w) = 1$ otherwise. Then, $G_M^\dagger$ with respect to σ is shown in Figure 1(e), where an arc, say, from w to u, means $\sigma(u, v; w) = 0$ for some v adjacent to w. As shown in Figure 1(f), there are four fault-free M-components contained in $G_M^\dagger$; one itself is a fault-free aggregate, and the other three constitute another fault-free aggregate.

We use C_M to denote the set of all M-components in $G_M^\dagger$. When $|C_M| = 1$, either all vertices of G are fault-free or all vertices of G are faulty. So, we assume $|C_M| \geq 2$. Let $C_M^{(1)} = \{\{x\} : \{x\} \in C_M\}$ be the set of all 1-vertex M-components in $G_M^\dagger$. Clearly, $C_M^{(1)} \subseteq C_M$.

Lemma 1. *Suppose that Λ is a fault-free aggregate in G with $|\Lambda| \geq \Delta + 2$. If $\{x\} \in C_M^{(1)}$ and $x \in \Lambda$, then $|N(x) \cap \Lambda| \leq 1$ under the MM* model.*

Proof. We have $\Lambda - (N(x) \cup \{x\})$ not empty, as a consequence of $|\Lambda| \geq \Delta + 2$. Suppose that $z \in \Lambda - (N(x) \cup \{x\})$ is adjacent to $(N(x) \cup \{x\}) \cap \Lambda$, i.e., $z \in N(y_1)$ for some $y_1 \in N(x) \cap \Lambda$. If $|N(x) \cap \Lambda| \geq 2$, then there exists $y_2 \in N(x) \cap \Lambda - \{y_1\}$. Consequently, $\sigma(y_1, y_2; x) = 0$ and $\sigma(x, z; y_1) = 0$, i.e., $y_1 \in \Upsilon_M(x)$ and $x \in \Upsilon_M(y_1)$, which means that x and y_1 belong to the same M-component, which is a contradiction. □

In Appendix A, we list some important variables and notation that are used in this paper.

3. Identification of fault-free aggregates. Since all neighbors of a fault-free aggregate are faulty, faulty vertices can be found provided some fault-free aggregates are identified. Recall that when (t, k)-diagnosis is concerned, the number of faulty vertices is bounded above by t. For the PMC model, since each fault-free P-component is a fault-free aggregate, a vertex subset in G can be guaranteed to be a fault-free aggregate provided it is a P-component and has size greater than or equal to $t + 1$. In other words, $t + 1$ is a threshold for guaranteeing a fault-free aggregate under the PMC model. Let ζ_P denote such a threshold. Clearly, $t + 1$ is a feasible value of ζ_P.

In the rest of this section, we consider the MM* model and use ζ_M to denote a threshold for guaranteeing a fault-free aggregate. Since a fault-free aggregate may contain multiple M-components under the MM* model, it is not an easy problem to determine a feasible value of ζ_M. In the following, we intend to find a feasible value of ζ_M, together with some conditions, so that a vertex subset in G can be guaranteed to be a fault-free aggregate under the MM* model provided it satisfies these conditions and has size greater than or equal to ζ_M.

First, a sufficient and necessary condition for a subset V' of V with $|V'| \geq \frac{\Delta}{\delta-2} \times t(\geq \Delta + 2)$ to be a fault-free aggregate is proposed below.

Lemma 2. *Suppose that $V' = (\bigcup_{1 \leq i \leq m} X_i) \cup (\bigcup_{1 \leq j \leq n} Y_j)$ is a subset of V and $|V'| \geq \frac{\Delta}{\delta-2} \times t$ $(\geq \Delta + 2)$, where $m \geq 0$, $n \geq 0$, $X_i \in C_M^{(1)}$, $Y_j \in C_M - C_M^{(1)}$, and $\delta > 2$. Then, under the MM* model, V' is a fault-free aggregate in G if and only if the following five conditions hold:*

(C1) *$|N(X_i) \cap V'| = 1$ for $1 \leq i \leq m$;*

(C2) *$N(X_i) \cap V' \subseteq \bigcup_{1 \leq j \leq n} Y_j$ for $1 \leq i \leq m$;*

(C3) *$n = 1$;*

(C4) *for each $X_i = \{x_i\}$, $x_i \in \Upsilon_M(y_i)$ for some $y_i \in Y_1$;*

(C5) *$\bigcup_{1 \leq i \leq m} X_i = \bigcup_{X_j \in C_M^{(1)'}} X_j$, where $C_M^{(1)'} = \{X_j : X_j = \{x_j\} \in C_M^{(1)}$ and $x_j \in \Upsilon_M(y_j)$ for some $y_j \in Y_1\}$.*

Proof. (⇒) (C1) Clearly, $|N(X_i) \cap V'| \geq 1$. On the other hand, since $X_i \in C_M^{(1)}$ is fault-free, Lemma 1 ensures that X_i has at most one fault-free neighbor, i.e., $|N(X_i) \cap V'| \leq 1$. So, $|N(X_i) \cap V'| = 1$.

(C2) Suppose conversely that $(N(X_r) \cap V') \cap (\bigcup_{1 \leq i \leq m} X_i)$ is not empty for some $1 \leq r \leq m$. By (C1), $N(X_r) \cap V' = X_l$, where $1 \leq l \leq m$ and $l \neq r$. It follows that there is no arc between $X_l \cup X_r$ and $V' - (X_l \cup X_r)$, i.e., $|V'| = 2$, which is a contradiction.

(C3) Suppose conversely that $n \geq 2$. Since V' is an aggregate, it is implied by (C2) that for each Y_j, there exists Y_l satisfying $Y_l \cap N(Y_j) \neq \emptyset$. Suppose, without loss of generality, that $Y_2 \cap N(Y_1) \neq \emptyset$; i.e., there are arcs between $y_1 \in Y_1$ and $y_2 \in Y_2$. Since y_1, y_2 are fault-free and Y_1, $Y_2 \in C_M - C_M^{(1)}$, we have $y_1 \in \Upsilon_M(y_2)$ and $y_2 \in \Upsilon_M(y_1)$. It follows that y_1 and y_2 should belong to the same M-component (in $G_M^{\dagger}$), which is a contradiction.

(C4) By (C2) and (C3), we have $y_i \in N(X_i)$ for some $y_i \in Y_1$. Since x_i, y_i are fault-free and $Y_1 \in C_M - C_M^{(1)}$, we have $x_i \in \Upsilon_M(y_i)$.

(C5) Clearly, $\bigcup_{1\leq i\leq m} X_i \subseteq \bigcup_{X_j \in C_M^{(1)\prime}} X_j$. Suppose conversely that there exists $z \in \bigcup_{X_j \in C_M^{(1)\prime}} X_j - \bigcup_{1\leq i\leq m} X_i$. Then, $z \in \Upsilon_M(y_j)$ for some $y_j \in Y_1$. Since y_j is fault-free, we have z fault-free, which implies $z \in V'$, which is a contradiction.

For example, a fault-free aggregate Λ under the MM* model is shown in Figure 2, where $\Lambda = X_1 \cup X_2 \cup Y_1$ is assumed. As a consequence of (C1), (C2), and (C3), X_1 and X_2 each have exactly one neighbor in Y_1, and their other neighbors, which are faulty, are outside Λ.

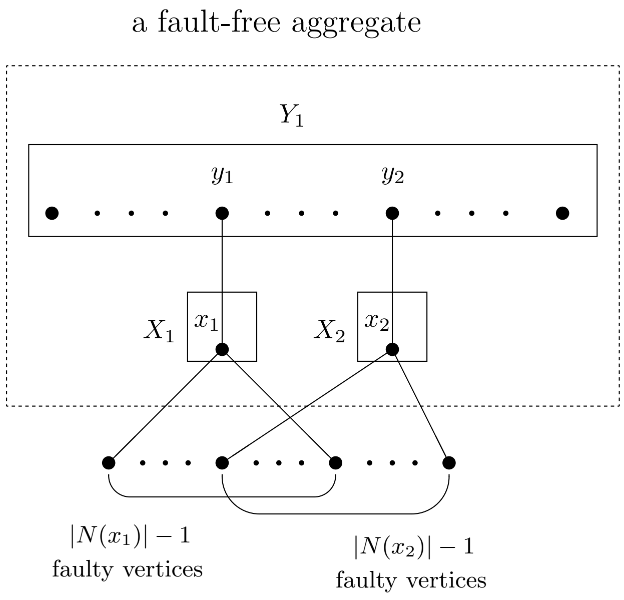


FIG. 2. *A fault-free aggregate under the MM* model.*

($\Leftarrow$) We first show that V' is fault-free. Suppose conversely that there are faulty vertices in V'. If Y_1 is fault-free, then by (C4), each X_i is also fault-free, which is a contradiction. Therefore, Y_1 is faulty. Since $|V'| \geq \frac{\Delta}{\delta-2} \times t$, we suppose that there are α fault-free vertices in $\bigcup_{1\leq i\leq m} X_i$ and they have β faulty neighbors outside V'. Then, we have

$$\beta < t \tag{1}$$

and $|V'| - \alpha + \beta \leq t$, which further imply

$$\frac{\Delta}{\delta-2} \times t - \alpha + \beta \leq t. \tag{2}$$

Lemma 1 ensures that each fault-free vertex in $\bigcup_{1\leq i\leq m} X_i$ has at least $\delta-1$ faulty neighbors. As a consequence of (C1), (C2), and (C3), $|N(X_i)\cap V'| = |N(X_i)\cap Y_1| = 1$ for each X_i. Since Y_1 is faulty, each fault-free vertex in $\bigcup_{1\leq i\leq m} X_i$ has at least $\delta - 2$ faulty neighbors outside V'. Also notice that each vertex of G has indegree (and outdegree) smaller than or equal to Δ. So, we have

$$\alpha \times (\delta - 2) \leq \beta \times \Delta. \tag{3}$$

Combining (1) and (3), we have

$$\alpha \times (\delta - 2) < t \times \Delta. \tag{4}$$

On the other hand, since $\beta \geq \frac{\alpha\times(\delta-2)}{\Delta}$ (from (3)), (2) can be written as follows:

$$\begin{aligned} t \times (\Delta - \delta + 2) &\leq (\alpha - \beta) \times (\delta - 2) \\ &\leq \left(\alpha - \frac{\alpha \times (\delta - 2)}{\Delta}\right) \times (\delta - 2) \\ &= \frac{\alpha \times (\Delta - \delta + 2) \times (\delta - 2)}{\Delta}, \end{aligned}$$

from which $t \times \Delta \leq \alpha \times (\delta - 2)$ is derived, which is a contradiction to (4).

Next, we show that V' is an aggregate in G. Suppose conversely that V' is not an aggregate. Then, V' is a proper subset of a fault-free aggregate Λ, and there exists an M-component Z in $G_M^{\dagger}$ with $Z \subset \Lambda - V'$. By (C3), we have $Z \in C_M^{(1)}$; by (C4), we have $Z \subseteq \bigcup_{X_j\in C_M^{(1)\prime}} X_j$. Further, (C5) ensures that $Z \subseteq \bigcup_{1\leq i\leq m} X_i \subset V'$, which is a contradiction. □

According to Lemma 2, when the five conditions hold, $\frac{\Delta}{\delta-2} \times t$ can be a threshold for guaranteeing a fault-free aggregate under the MM* model (i.e., set ζ_M to be $\frac{\Delta}{\delta-2} \times t$). In the next section, it is shown that the value of t under the MM* model depends on ζ_M, and in section 5, $\frac{\Delta}{\delta-2} \times t$ is evaluated in order to compute the degrees of (t, k)-diagnosability for grids and tori.

4. (t, k)-diagnosability.

In [23], Khanna and Fuchs defined a function ϕ in order to determine a lower bound on the degree of sequential diagnosability of a general graph G. In this section, we intend to determine a lower bound on the degree of (t, k)-diagnosability of G. We assume that the diagnosis model is the MM* model. The discussion for the PMC model is all the same. We need only to replace M-component, $G_M^{\dagger}$, and ζ_M with P-component, $G_P^{\dagger}$, and ζ_P, respectively. First we adapt the definition of ϕ to (t, k)-diagnosis, as elaborated below.

Suppose that $V' \subset V$ and $\{V_1, V_2, \ldots, V_r\}$ is a partition of V' so that the induced subgraph of each V_i in G is strongly connected and no arc exists between V_i and V_j if $i \neq j$. Let $\pi(V_i)$ denote the set of all possible partitions of V_i, and define $\Pi(V') = \{\pi_1 \cup \pi_2 \cup \cdots \cup \pi_r : \pi_i \in \pi(V_i) \text{ for all } 1 \leq i \leq r\}$ to be the set of all possible partitions of V' that are induced by further partitioning $V_1, V_2, \ldots, V_r$. An illustrative example is shown in Figure 3, where $V' = V_1 \cup V_2$ is the set of all black vertices in G. For our purpose, $V' = F$ is the fault set of G, each V_i is a faulty aggregate in G, and each element (i.e., a subset of V_i) of $\pi_i \in \pi(V_i)$ is a faulty M-component in $G_M^{\dagger}$ under the MM* model.

Suppose that $W \subseteq V - V'$ and the induced subgraph of W in $G - V'$ is maximally strongly connected. For each $\Gamma \in \Pi(V')$, let $\vartheta_{\Gamma,W} = \Sigma_{Z\in\Gamma and\ Z\cap N(W)\neq\emptyset}|Z|$, which

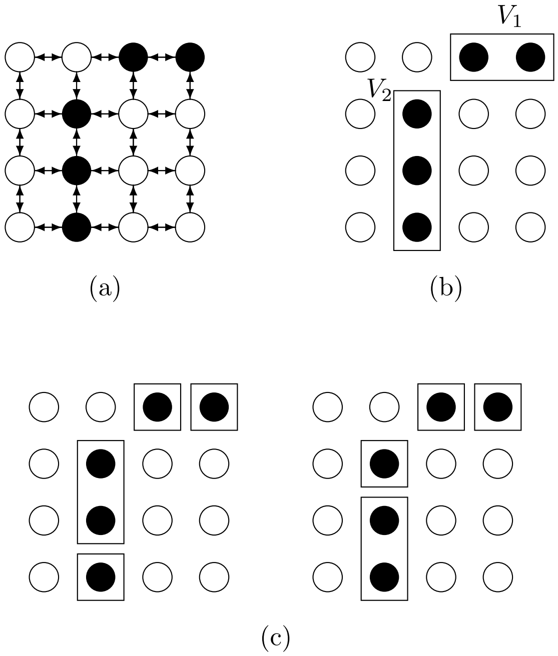


FIG. 3. *An example of* (a) G, (b) $V' = V_1 \cup V_2$, *and* (c) *two elements of* $\Pi(V')$.

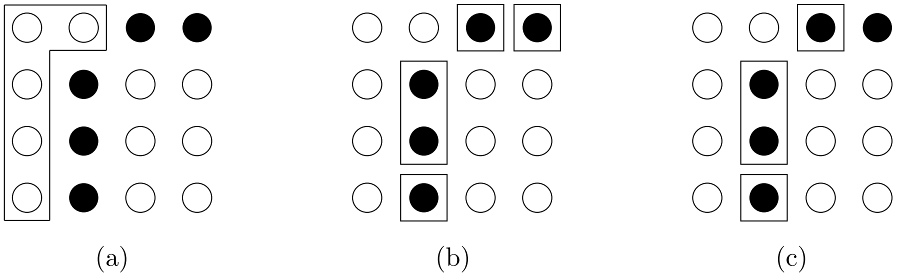


FIG. 4. *An example of* (a) W, (b) Γ, *and* (c) *the elements of* Γ *that overlap with* $N(W)$.

is the total size (i.e., total number of vertices) of those elements (i.e., subsets of V_1, V_2, or V_r) of Γ that overlap with $N(W)$. An illustrative example is shown in Figure 4, where G and V' are the same as those used in Figure 3. Further, we define $\xi_{V',W} = \min\{\vartheta_{\Gamma,W} : \Gamma \in \Pi(V')\}$, which is the minimum of $\vartheta_{\Gamma,W}$'s for all $\Gamma \in \Pi(V')$. We have the following lemma.

LEMMA 3. *If* $W \subset V - V'$, *then* $\xi_{V',W} \geq \kappa(G)$, *where* $\kappa(G)$ *is the vertex connectivity of* G.

Proof. Since $W \subset V - V'$ and the induced subgraph of W in $G - V'$ is maximally strongly connected, there is no arc in G whose two end vertices belong to W and $G - V' - W$, respectively. Hence, we have $\vartheta_{\Gamma,W} \geq \kappa(G)$ for all $\Gamma \in \Pi(V')$, which further implies $\xi_{V',W} \geq \kappa(G)$. □

For our purpose, W is a fault-free aggregate in G. Additionally, there exists $\Gamma' \in \Pi(V')$, which is the set of faulty M-components in $G_M^\dagger$ under the MM* model. Consequently, $\vartheta_{\Gamma',W}$ is the total number of vertices contained in those faulty M-components that are neighboring to the fault-free aggregate W under the MM* model.

Definition 4. *Suppose that $G = (V, A)$ is a directed graph. Define $\Phi(\chi_1, \chi_2)$ to be the greatest integer p so that for each subset V' of V with $|V'| \leq p$, there exists $W \subseteq V - V'$ satisfying the following three conditions:*

(C1) *the induced subgraph of W in $G - V'$ is maximally strongly connected;*

(C2) $|W| \geq \chi_1$*;*

(C3) $\xi_{V',W} \geq \chi_2$ *provided $W \subset V - V'$.*

Recall that the process of (t, k)-diagnosis is iterative. In each iteration (exclusive of the last iteration), at least k faulty vertices can be identified provided there are at most t faulty vertices in G. For the purpose of (t, k)-diagnosis under the MM* model, we let $V' = F$, $\chi_1 = \zeta_M$, and $\chi_2 = \ddot{q}$, where $\Phi(\zeta_M, \ddot{q}) \geq \Phi(\zeta_M, q)$ for all nonnegative integers q. Then, G is $(\hat{t}, \hat{k})$-diagnosable under the MM* model, where $\hat{t} = \Phi(\zeta_M, \ddot{q})$ and $\hat{k} = \ddot{q}$. We have $|F| \leq \hat{t}(= \Phi(\zeta_M, \ddot{q}))$. There are at least $\hat{k}$ faulty vertices identified in each iteration (exclusive of the last iteration), as explained below.

There is a fault-free aggregate W with $|W| \geq \zeta_M$, which is guaranteed by (C1) and (C2). Moreover, W can be identified, according to the discussion of section 3. Suppose that n total iterations are required to complete the $(\hat{t}, \hat{k})$-diagnosis of G. Let $F^{(i)}$ be the resulting fault set of G at the beginning of the ith iteration, and let $W^{(i)}$ be the fault-free aggregate identified in the ith iteration, where $1 \leq i \leq n$. Without loss of generality, suppose that $W^{(i)} \subset V - F^{(i)}$ for $i < m$ and $W^{(i)} = V - F^{(i)}$ for $i \geq m$, where $1 < m \leq n$. For $1 \leq i < m$, at least $\ddot{q}(= \hat{k})$ faulty vertices of G can be identified in the ith iteration, which is ensured by (C3). For $m \leq i < n$, at least $\ddot{q}(= \hat{k})$ faulty vertices of G can be identified in the ith iteration as well, which is shown in Appendix B. In the nth iteration, all remaining faulty vertices of G are identified.

In the rest of this section, we aim to evaluate $\hat{t}(= \Phi(\zeta_M, \ddot{q}))$ and $\hat{k}(= \ddot{q})$ under the MM* model. By Lemma 3, we have $\xi_{V',W} \geq \kappa(G)$ if $W \subset V - V'$. Refer to Definition 4 again. We have $\Phi(\zeta_M, \kappa(G)) \geq \Phi(\zeta_M, q)$ for all nonnegative integers q. So, $\ddot{q} = \kappa(G)$ is a feasible value of $\hat{k}$. On the other hand, if (C3) holds for $\chi_2 = \ddot{q}$, then (C3) holds for $\chi_2 < \ddot{q}$ as well. That is, $\Phi(\zeta_M, \ddot{q}) \leq \Phi(\zeta_M, \ddot{q} - 1) \leq \cdots \leq \Phi(\zeta_M, 0)$. Also, since $\Phi(\zeta_M, \ddot{q}) \geq \Phi(\zeta_M, q)$ for all nonnegative integers q, we have $\Phi(\zeta_M, \ddot{q}) = \Phi(\zeta_M, 0)$. In order to evaluate $\Phi(\zeta_M, 0)$, we define $\hat{\Phi}(\chi_1) = \min\{|V''| : V'' \subset V$ and there is no $W' \subseteq V - V''$ with $|W'| \geq \chi_1$ so that the induced subgraph of W' in $G - V''$ is maximally strongly connected$\}$. Clearly, $\Phi(\zeta_M, 0) = \hat{\Phi}(\zeta_M) - 1$.

Further, in order to evaluate $\hat{\Phi}(\zeta_M)$, we let $A_{\tilde{V}} = \{(z_1, z_2) : z_1, z_2 \in \tilde{V}$ and (z_1, z_2) is an arc in $G\}$, where $\tilde{V} \subseteq V$, and define $I(\alpha) = \max\{|A_{\tilde{V}}| : \tilde{V} \subseteq V$ and $|\tilde{V}| = \alpha\}$ to be the maximal number of arcs in G whose two end vertices are contained in a fixed α-vertex subset of V. It is rather difficult to evaluate $I(\alpha)$; instead, an approximation of $I(\alpha)$ is evaluated below. Let $\hat{I} : [0, |V|] \to R^+ \cup \{0\}$ be a convex function with $\hat{I}(0) = 0$ and $\hat{I}(\alpha) \geq I(\alpha)$ for all positive integers α, where $[0, |V|]$ denotes the set of real numbers ranging from 0 to $|V|$ and R^+ denotes the set of all positive real numbers. Notice that $\hat{I}$ is a continuous function defined on $[0, |V|]$, while I is a discrete function defined on $\{0, 1, 2, \ldots, |V|\}$. We have the following lemma.

Lemma 4. *Suppose that $V' \subset V$ and $\{W_1, W_2, \ldots, W_r\}$ is a partition of $V - V'$ so that the induced subgraph of each W_i in $G - V'$ is maximally strongly connected, where $1 \leq i \leq r$. If $|W_i| < \zeta_M$ for all $1 \leq i \leq r$, then $\Sigma_{i=1}^{r} \hat{I}(|W_i|) \leq \frac{\Sigma_{i=1}^{r}|W_i|}{\zeta_M} \times \hat{I}(\zeta_M)$.*

Proof. Since $\hat{I}$ is convex, we have $\frac{\hat{I}(x) - \hat{I}(0)}{x} \leq \hat{I}'(x)$, where $\hat{I}'(x)$ is the first-order derivative of $\hat{I}(x)$. Hence, $x \times \hat{I}'(x) - \hat{I}(x) \geq 0$. Let $h(x) = \frac{\hat{I}(x)}{x}$. Then, $h'(x) = \frac{x \times \hat{I}'(x) - \hat{I}(x)}{x^2} \geq 0$. That is, $\frac{\hat{I}(x)}{x}$ is nondecreasing. So, $\frac{\hat{I}(\zeta_M)}{\zeta_M} \geq \frac{\hat{I}(|W_i|)}{|W_i|}$ for all

$1 \leq i \leq r$, which implies $\Sigma_{i=1}^{r}\hat{I}(|W_i|) \leq \frac{\Sigma_{i=1}^{r}|W_i|}{\zeta_M} \times \hat{I}(\zeta_M)$. □

By the aid of Lemma 4, a lower bound on $\hat{\Phi}(\zeta_M)$ can be obtained as follows.

LEMMA 5. $\hat{\Phi}(\zeta_M) \geq \frac{|A|\times\zeta_M - |V|\times\hat{I}(\zeta_M)}{2\times\Delta\times\zeta_M - \hat{I}(\zeta_M)}$.

Proof. Suppose $\hat{\Phi}(\zeta_M) = q$. That is, there exists $V'' \subset V$ with $|V''| = q$ so that there is no $W' \subseteq V - V''$ with $|W'| \geq \zeta_M$ and the induced subgraph of W' in $G - V''$ is maximally strongly connected. Also suppose that $\{W_1, W_2, \ldots, W_r\}$ is a partition of $V - V''$ so that the induced subgraph of each W_i in $G - V''$ is maximally strongly connected, where $1 \leq i \leq r$. Since each vertex of G has outdegree (and indegree) smaller than or equal to Δ, $|V''| \times \Delta$ is greater than or equal to $I(|V''|)$ plus the number of arcs from V'' to $V - V''$. Also, $|V''| \times \Delta$ is greater than or equal to the number of arcs from $V - V''$ to V''. Hence, we have

$$\begin{aligned} |A| &\leq \Sigma_{i=1}^{r} I(|W_i|) \ + \ I(|V''|) \ + \ (\text{the number of arcs between } V'' \text{ and } V - V'') \\ &\leq \Sigma_{i=1}^{r} I(|W_i|) \ + \ 2 \times |V''| \times \Delta. \qquad (5) \end{aligned}$$

Since $|W_i| < \zeta_M$ for $1 \leq i \leq r$ and $\Sigma_{i=1}^{r}|W_i| = |V - V''|$, we have $\Sigma_{i=1}^{r} I(|W_i|) \leq \Sigma_{i=1}^{r}\hat{I}(|W_i|) \leq \frac{|V-V''|}{\zeta_M} \times \hat{I}(\zeta_M)$ by Lemma 4. Thus, (5) can be rewritten as follows:

$$|A| \leq \frac{|V - V''|}{\zeta_M} \times \hat{I}(\zeta_M) + 2 \times |V''| \times \Delta,$$

from which we have

$$|V''| \geq \frac{|A| \times \zeta_M - |V| \times \hat{I}(\zeta_M)}{2 \times \Delta \times \zeta_M - \hat{I}(\zeta_M)}. \qquad □$$

By Lemma 5, we have $\Phi(\zeta_M, 0) = \hat{\Phi}(\zeta_M) - 1 \geq \frac{|A|\times\zeta_M - |V|\times\hat{I}(\zeta_M)}{2\times\Delta\times\zeta_M - \hat{I}(\zeta_M)} - 1$, i.e.,

$$\hat{t} \geq \frac{|A| \times \zeta_M - |V| \times \hat{I}(\zeta_M)}{2 \times \Delta \times \zeta_M - \hat{I}(\zeta_M)} - 1, \qquad (6)$$

which means that $\frac{|A|\times\zeta_M - |V|\times\hat{I}(\zeta_M)}{2\times\Delta\times\zeta_M - \hat{I}(\zeta_M)} - 1$ is a lower bound on $\hat{t}$.

Now that G is $(\hat{t}, \hat{k})$-diagnosable, G is also $(\hat{t}', \hat{k})$-diagnosable for all $\hat{t}' \leq \hat{t}$. The following theorem summarizes the discussion above.

THEOREM 6. *G is $(\hat{t}', \hat{k})$-diagnosable under the MM* model, where* $\hat{t}' = \frac{|A|\times\zeta_M - |V|\times\hat{I}(\zeta_M)}{2\times\Delta\times\zeta_M - \hat{I}(\zeta_M)} - 1$ *and* $\hat{k} = \kappa(G)$.

Since the discussion for the PMC model is the same as the discussion for the MM* model, we have the following theorem for the PMC model.

THEOREM 7. *G is $(\hat{t}', \hat{k})$-diagnosable under the PMC model, where* $\hat{t}' = \frac{|A|\times\zeta_P - |V|\times\hat{I}(\zeta_P)}{2\times\Delta\times\zeta_P - \hat{I}(\zeta_P)} - 1$ *and* $\hat{k} = \kappa(G)$.

5. Applications.

In this section, as two applications of Theorems 6 and 7, we compute the (t,k)-diagnosability of grids and tori.

5.1. Grids.

The *Cartesian product* [32] of $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, written as $G_1 \otimes G_2$, is the graph $G = (V, E)$, where $V = \{\langle u,v \rangle : u \in V_1 \text{ and } v \in V_2\}$ and $E = \{(\langle u,v \rangle, \langle u',v' \rangle) : u = u' \text{ and } (v, v') \in E_2 \text{ or } v = v' \text{ and } (u, u') \in E_1\}$. Let P_k denote a path of k vertices. A *d-dimensional grid* [23], denoted by

$\mathcal{G}_d(n_1, n_2, \ldots, n_d)$, is the graph obtained by $P_{n_1} \otimes P_{n_2} \otimes \cdots \otimes P_{n_d}$, where each n_i $(1 \leq i \leq d)$ is the length of dimension i. There are $n_1 \times n_2 \times \cdots \times n_d$ vertices contained in $\mathcal{G}_d(n_1, n_2, \ldots, n_d)$. In subsequent discussion, we use $\mathcal{G}_d(n)$ to represent $\mathcal{G}_d(n_1, n_2, \ldots, n_d)$ with $n_1 = n_2 = \cdots = n_d = n$. Figure 5 shows the topology of $\mathcal{G}_2(5)$.

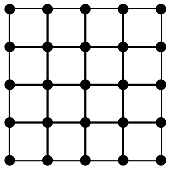


FIG. 5. *The topology of* $\mathcal{G}_2(5)$.

For $\mathcal{G}_d(n)$, we have $|A| = 2 \times d \times (|V| - |V|^{\frac{d-1}{d}})$ (see [23]), $\hat{I}(\alpha) = 2 \times d \times (\alpha - \alpha^{\frac{d-1}{d}})$ (see [23]), $\Delta = 2 \times d$, and $\delta = d$ (considering each edge of $\mathcal{G}_d(n)$ bi-directional). Moreover, we have $\kappa(\mathcal{G}_d(n)) = d$. Then, as a consequence of Theorems 6 and 7, we have the following two theorems.

THEOREM 8. *$\mathcal{G}_d(n)$ is $(\Omega(|V|^{\frac{d}{d+1}}), \Omega(d))$-diagnosable under the MM* model.*

Proof. According to Theorem 6, we have

$$\hat{t}' = \frac{|A| \times \zeta_M - |V| \times \hat{I}(\zeta_M)}{2 \times \Delta \times \zeta_M - \hat{I}(\zeta_M)} - 1$$

$$= \frac{2 \times d \times (|V| - |V|^{\frac{d-1}{d}}) \times \zeta_M - |V| \times 2 \times d \times (\zeta_M - \zeta_M^{\frac{d-1}{d}})}{4 \times d \times \zeta_M - 2 \times d \times (\zeta_M - \zeta_M^{\frac{d-1}{d}})} - 1$$

$$= \frac{|V| - \zeta_M^{\frac{1}{d}} \times |V|^{\frac{d-1}{d}}}{1 + \zeta_M^{\frac{1}{d}}} - 1, \tag{7}$$

and $\hat{k} = \kappa(\mathcal{G}_d(n)) = d$.

Since $\zeta_M = \frac{\Delta}{\delta-2} \times \hat{t}' = \frac{2\times d}{d-2} \times \hat{t}'$ (as a consequence of Lemma 2), we have $\hat{t}' > \frac{1-|V|^{\frac{-1}{d\times(d+1)}}}{3} \times |V|^{\frac{d}{d+1}}$, which can be derived from (7). Now that G is $(\hat{t}', \hat{k})$-diagnosable, G is also $(\hat{t}'', \hat{k})$-diagnosable for all $\hat{t}'' < \hat{t}'$. Hence, the theorem follows. □

THEOREM 9. *$\mathcal{G}_d(n)$ is $(\Omega(|V|^{\frac{d}{d+1}}), \Omega(d))$-diagnosable under the PMC model.*

Proof. Similarly, according to Theorem 7, we have

$$\hat{t}' = \frac{|V| - \zeta_P^{\frac{1}{d}} \times |V|^{\frac{d-1}{d}}}{1 + \zeta_P^{\frac{1}{d}}} - 1, \tag{8}$$

and $\hat{k} = d$. Since $\zeta_P = \hat{t}' + 1$ (refer to section 3), we have $\hat{t}' > \frac{1-|V|^{\frac{-1}{d\times(d+1)}}}{2} \times |V|^{\frac{d}{d+1}} - 1$, which can be derived from (8). Hence, the theorem follows. □

5.2. Tori. If we replace each path P_{n_i} of $\mathcal{G}_d(n_1, n_2, \ldots, n_d)$ with a cycle C_{n_i}, then a d-dimensional torus [8], denoted by $\mathcal{T}_d(n_1, n_2, \ldots, n_d)$, results, where C_{n_i} denotes a cycle of n_i vertices. That is, $\mathcal{T}_d(n_1,n_2,\ldots,n_d)=C_{n_1}\otimes C_{n_2}\otimes\cdots\otimes C_{n_d}$. We use $\mathcal{T}_d(n)$ to represent $\mathcal{T}_d(n_1,n_2,\ldots,n_d)$ with $n_1=n_2=\cdots=n_d=n$. Figure 6 shows the topology of $\mathcal{T}_2(5)$. For $\mathcal{T}_d(n)$, we have $|A|=2\times d\times|V|$, $\Delta=\delta=2\times d$ (considering each edge of $\mathcal{T}_d(n)$ bidirectional), and $\kappa(\mathcal{T}_d(n))=2d$. The value of $\hat{I}(\alpha)$ is computed below.

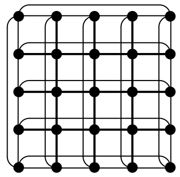


FIG. 6. *The topology of $\mathcal{T}_2(5)$.*

LEMMA 10. *For $\mathcal{T}_d(n)$, $\hat{I}(\alpha)=2\times d\times\alpha-2\times n^{\frac{d-\mu}{\mu}}\times\mu\times\alpha^{\frac{\mu-1}{\mu}}$, where $\mu=\ln(\frac{n^d}{\alpha})$.*

Proof. Refer to Appendix C. □

Similarly, as a consequence of Theorems 6 and 7, we have the following two theorems.

THEOREM 11. *$\mathcal{T}_d(n)$ is $(\Omega(|V|^{\frac{d}{d+1}}),\Omega(2\times d))$-diagnosable under the MM* model.*

Proof. According to Theorem 6 and Lemma 10, we have

$$\hat{t}'=\frac{|A|\times\zeta_M-|V|\times\hat{I}(\zeta_M)}{2\times\Delta\times\zeta_M-\hat{I}(\zeta_M)}-1$$

$$=\frac{2\times d\times|V|\times\zeta_M-|V|\times(2\times d\times\zeta_M-2\times n^{\frac{d-\mu}{\mu}}\times\mu\times\zeta_M^{\frac{\mu-1}{\mu}})}{4\times d\times\zeta_M-(2\times d\times\zeta_M-2\times n^{\frac{d-\mu}{\mu}}\times\mu\times\zeta_M^{\frac{\mu-1}{\mu}})}-1$$

$$(9)\qquad=\frac{|V|\times n^{\frac{d}{\mu}}\times\mu\times\zeta_M^{\frac{-1}{\mu}}}{d\times n+n^{\frac{d}{\mu}}\times\mu\times\zeta_M^{\frac{-1}{\mu}}}-1,$$

and $\hat{k}=\kappa(\mathcal{T}_d(n))=2\times d$. Since $\mu=\ln(\frac{n^d}{\zeta_M})$ and $\zeta_M=\frac{\Delta}{\delta-2}\times\hat{t}'=\frac{2\times d}{2\times d-2}\times\hat{t}'$ (as a consequence of Lemma 2), we have $\hat{t}'>0.1\times|V|^{\frac{d}{d+1}}$, which can be derived from (9). Hence, the theorem follows. □

THEOREM 12. *$\mathcal{T}_d(n)$ is $(\Omega(|V|^{\frac{d}{d+1}}),\Omega(2\times d))$-diagnosable under the PMC model.*

Proof. Similarly, according to Theorem 7 and Lemma 10, we have

$$(10)\qquad\hat{t}'=\frac{|V|\times n^{\frac{d}{\mu}}\times\mu\times\zeta_P^{\frac{-1}{\mu}}}{d\times n+n^{\frac{d}{\mu}}\times\mu\times\zeta_P^{\frac{-1}{\mu}}}-1,$$

and $\hat{k}=\kappa(\mathcal{T}_d(n))=d$. Since $\mu=\ln(\frac{n^d}{\zeta_P})$ and $\zeta_P=\hat{t}'+1$, we have $\hat{t}'>0.1\times|V|^{\frac{d}{d+1}}$, which can be derived from (10). Hence, the theorem follows. □

For $\mathcal{T}_2(n)$, $\hat{I}(\alpha)=2\times d\times\alpha-\min\{4\times\sqrt{\alpha},2\times\sqrt{|V|}\}$ was computed in [26]. Since $|A|=4\times(|V|-|V|^{\frac{1}{2}})$, $\Delta=\delta=4$, and $\kappa(\mathcal{T}_2(n))=2$, we have the following two theorems.

THEOREM 13. *$\mathcal{T}_2(n)$ is $(\Omega(|V|^{\frac{2}{3}}),4)$-diagnosable under the MM* model.*

Proof. According to Theorem 6, we have

$$\hat{t}' = \frac{|A| \times \zeta_M - |V| \times \hat{I}(\zeta_M)}{2 \times \Delta \times \zeta_M - \hat{I}(\zeta_M)} - 1$$

$$= \frac{2 \times d \times |V| \times \zeta_M - |V| \times (2 \times d \times \zeta_M - \min\{4 \times \sqrt{\zeta_M}, 2 \times \sqrt{|V|}\})}{4 \times d \times \zeta_M - (2 \times d \times \zeta_M - \min\{4 \times \sqrt{\zeta_M}, 2 \times \sqrt{|V|}\})} - 1$$

$$(11) \quad = \frac{|V| \times \min\{2 \times \sqrt{\zeta_M}, \sqrt{|V|}\}}{2 \times \zeta_M + \min\{2 \times \sqrt{\zeta_M}, \sqrt{|V|})} - 1.$$

Since $\zeta_M = \frac{\Delta}{\delta - 2} \times \hat{t}' = \frac{2 \times d}{2 \times d - 2} \times \hat{t}' = 2 \times \hat{t}'$ (as a consequence of Lemma 2), we have $\hat{t}' > \min\{\frac{|V|^{\frac{2}{3}}}{2}, \frac{|V|}{8}\}$, which can be derived from (11). Hence, the theorem follows. □

THEOREM 14. *$\mathcal{T}_2(n)$ is $(\Omega(|V|^{\frac{2}{3}}), 4)$-diagnosable under the PMC model.*

Proof. Similarly, according to Theorem 7, we have

$$(12) \qquad \hat{t}' = \frac{|V| \times \min\{2 \times \sqrt{\zeta_P}, \sqrt{|V|}\}}{2 \times \zeta_P + \min\{2 \times \sqrt{\zeta_P}, \sqrt{|V|})} - 1.$$

Since $\zeta_P = \hat{t}' + 1$, we have $\hat{t}' > \min\{\frac{|V|^{\frac{2}{3}}}{2}, \frac{|V|-4}{4}\}$, which can be derived from (12). Hence, the theorem follows. □

The values of $\hat{I}(\alpha)$ for $\mathcal{G}_d(n_1, n_2, \ldots, n_d)$ with $n_1 \leq n_2 \leq \cdots \leq n_d$ and $\mathcal{T}_d(n)$ with even d were computed in [4] and [9], respectively. Their (t, k)-diagnosability can also be obtained by the aid of Theorems 6 and 7.

6. Discussion and conclusion. Research on fault diagnosis has spanned about 40 years, and now emphasis has shifted toward the identification of a large fraction of faulty processors, i.e., sequential diagnosis and (t, k)-diagnosis. Now that (t, k)-diagnosis is a generalization of sequential diagnosis, we focused our attention on (t, k)-diagnosis under two popular diagnosis models: PMC and MM*. Theorems 6 and 7, which are the main results of this paper, suggested lower bounds on the degrees of (t, k)-diagnosability of general multiprocessor systems under the two models.

Previously, lower bounds on the degrees of (t, k)-diagnosability of general multiprocessor systems under the PMC model were proposed in [1]. An immediate consequence of [1] is that under the PMC model, $\mathcal{G}_d(n)$ is $(2 \times \sqrt{N - 2 \times d + 2} + d - 3, d)$-diagnosable if $N \geq 2 \times d + 7$ and $(\frac{N+d+2}{3}, d)$-diagnosable otherwise, and $\mathcal{T}_d(n)$ is $(2 \times \sqrt{N - 4 \times d + 2} + 2 \times d - 3, 2 \times d)$-diagnosable if $N \geq 4 \times d + 7$ and $(\frac{N+4 \times d+2}{3}, 2 \times d)$-diagnosable otherwise, where $N = n^d$ is the number of processors. By the aid of Theorems 6 and 7, we have shown in this paper that $\mathcal{G}_d(n)$ and $\mathcal{T}_d(n)$ are $(\Omega(N^{\frac{d}{d+1}}), \Omega(d)$-diagnosable and $(\Omega(N^{\frac{d}{d+1}}), \Omega(2 \times d))$-diagnosable, respectively, under both the PMC and the MM* models.

Matching composition networks (MCNs), which were first proposed in [24], include hypercubes, twisted cubes, crossed cubes, and Möbius cubes. Their (t, k)-diagnosability under the PMC and the MM* models was investigated in [12] and [13], respectively. The values of $\hat{I}(\alpha)$ for MCNs are available in [12]. When applying them to Theorems 6 and 7, we can obtain the same results as in [12] and [13]. In Table 3, we summarize the degrees of the (t, k)-diagnosability in asymptotical notations of $\mathcal{G}_d(n)$ and $\mathcal{T}_d(n)$ that are derivable from [1], [12], [13], and this paper.

Table 3
Asymptotic degrees of the (t, k)-diagnosability of $\mathcal{G}_d(n)$ and $\mathcal{T}_d(n)$.

	PMC	MM*
$\mathcal{G}_d(n)$	1. $(2 \times \sqrt{\lvert V\rvert - 2 \times d + 2} + d - 3, d)$ if $\lvert V\rvert \geq 2 \times d + 7$, and $(\frac{\lvert V\rvert+d+2}{3}, d)$ else [1] 2. $(\Omega(\lvert V\rvert^{\frac{d}{d+1}}), \Omega(d))$ [this paper]	$(\Omega(\lvert V\rvert^{\frac{d}{d+1}}), \Omega(d))$ [this paper]
$\mathcal{T}_d(n)$	1. $(2 \times \sqrt{\lvert V\rvert - 4 \times d + 2} + 2 \times d - 3, 2 \times d)$ if $\lvert V\rvert \geq 4 \times d + 7$, and $(\frac{\lvert V\rvert+2\times d+2}{3}, 2 \times d)$ else [1] 2. $(\Omega(\lvert V\rvert^{\frac{d}{d+1}}), \Omega(2 \times d))$ [this paper]	$(\Omega(\lvert V\rvert^{\frac{d}{d+1}}), \Omega(2 \times d))$ [this paper]
MCNs	$\Omega(\frac{\lvert V\rvert * \log\log \lvert V\rvert}{\log \lvert V\rvert})$ [12], [this paper]	$\Omega(\frac{\lvert V\rvert * \log\log \lvert V\rvert}{\log \lvert V\rvert})$ [13], [this paper]

Table 4
Numerical degrees of the (t, k)-diagnosability of $\mathcal{T}_2(n)$ under the PMC model.

	[1]	Theorem 12	Theorem 14
$\mathcal{T}_2(20)$	$(40, 4)$	$(48, 4)$	$(48, 4)$
$\mathcal{T}_2(40)$	$(80, 4)$	$(125, 4)$	$(128, 4)$
$\mathcal{T}_2(80)$	$(160, 4)$	$(310, 4)$	$(341, 4)$
$\mathcal{T}_2(160)$	$(320, 4)$	$(738, 4)$	$(859, 4)$
$\mathcal{T}_2(320)$	$(640, 4)$	$(1729, 4)$	$(2166, 4)$
$\mathcal{T}_2(640)$	$(1280, 4)$	$(3971, 4)$	$(5460, 4)$
$\mathcal{T}_2(1280)$	$(2560, 4)$	$(8894, 4)$	$(13758, 4)$
$\mathcal{T}_2(2560)$	$(5120, 4)$	$(19961, 4)$	$(34670, 4)$
$\mathcal{T}_2(5120)$	$(10240, 4)$	$(44122, 4)$	$(87363, 4)$
$\mathcal{T}_2(10240)$	$(20480, 4)$	$(95616, 4)$	$(220141, 4)$
$\mathcal{T}_2(20480)$	$(40960, 4)$	$(554721, 4)$	$(554721, 4)$
$\mathcal{T}_2(40960)$	$(81920, 4)$	$(1397811, 4)$	$(1397811, 4)$
$\mathcal{T}_2(81920)$	$(163840, 4)$	$(3522263, 4)$	$(3522263, 4)$
$\mathcal{T}_2(163840)$	$(327680, 4)$	$(8875546, 4)$	$(8875546, 4)$

Table 5
Numerical degrees of the (t, k)-diagnosability of $\mathcal{T}_3(n)$ under the PMC model.

	[1]	Theorem 12
$\mathcal{T}_3(20)$	$(181, 6)$	$(761, 6)$
$\mathcal{T}_3(40)$	$(508, 6)$	$(3822, 6)$
$\mathcal{T}_3(80)$	$(1434, 6)$	$(18374, 6)$
$\mathcal{T}_3(160)$	$(4050, 6)$	$(86495, 6)$
$\mathcal{T}_3(320)$	$(11451, 6)$	$(402782, 6)$
$\mathcal{T}_3(640)$	$(32384, 6)$	$(1812957, 6)$
$\mathcal{T}_3(1280)$	$(91592, 6)$	$(8133931, 6)$
$\mathcal{T}_3(2560)$	$(259056, 6)$	$(35894726, 6)$
$\mathcal{T}_3(5120)$	$(732717, 6)$	$(1.59 \times 10^8, 6)$
$\mathcal{T}_3(10240)$	$(2072433, 6)$	$(6.96 \times 10^8, 6)$
$\mathcal{T}_3(20480)$	$(5861721, 6)$	$(3.01 \times 10^9, 6)$
$\mathcal{T}_3(40960)$	$(16579445, 6)$	$(2.36 \times 10^{10}, 6)$
$\mathcal{T}_3(81920)$	$(46893747, 6)$	$(1.13 \times 10^{11}, 6)$
$\mathcal{T}_3(163840)$	$(1.32 \times 10^8, 6)$	$(5.39 \times 10^{11}, 6)$

Table 6

Numerical degrees of the (t, k)-diagnosability of $\mathcal{T}_5(n)$ under the PMC model.

	[1]	Theorem 12
$\mathcal{T}_5(20)$	$(3584, 10)$	$(216158, 10)$
$\mathcal{T}_5(40)$	$(20245, 10)$	$(4213470, 10)$
$\mathcal{T}_5(80)$	$(114493, 10)$	$(79922574, 10)$
$\mathcal{T}_5(160)$	$(647641, 10)$	$(1.48 \times 10^9, 10)$
$\mathcal{T}_5(320)$	$(3663580, 10)$	$(2.71 \times 10^{10}, 10)$
$\mathcal{T}_5(640)$	$(20724309, 10)$	$(4.87 \times 10^{11}, 10)$
$\mathcal{T}_5(1280)$	$(1.17 \times 10^8, 10)$	$(8.66 \times 10^{12}, 10)$
$\mathcal{T}_5(2560)$	$(6.63 \times 10^8, 10)$	$(1.52 \times 10^{14}, 10)$
$\mathcal{T}_5(5120)$	$(3.75 \times 10^9, 10)$	$(2.65 \times 10^{15}, 10)$
$\mathcal{T}_5(10240)$	$(2.12 \times 10^{10}, 10)$	$(4.61 \times 10^{16}, 10)$
$\mathcal{T}_5(20480)$	$(1.20 \times 10^{11}, 10)$	$(9.10 \times 10^{17}, 10)$
$\mathcal{T}_5(40960)$	$(6.79 \times 10^{11}, 10)$	$(1.63 \times 10^{19}, 10)$
$\mathcal{T}_5(81920)$	$(3.84 \times 10^{12}, 10)$	$(2.93 \times 10^{20}, 10)$
$\mathcal{T}_5(163840)$	$(2.17 \times 10^{13}, 10)$	$(5.28 \times 10^{21}, 10)$

We also evaluate the degrees of the (t, k)-diagnosability for some particular $\mathcal{T}_d(n)$'s in Tables 4, 5, and 6. They show that our result (i.e., Theorem 12) is superior to the result of [1] for all experimental values of n. Moreover, the value of t obtained from Theorem 12 increases more quickly than that obtained from [1] as the value of n increases. Table 4 also shows that the result of Theorem 14 is better than the result of Theorem 12 for $n = 40, 80, \ldots, 10240$, which is a consequence of using a tighter upper bound of $I(\alpha)$ (i.e., a smaller value of $\hat{I}(\alpha)$). A better (t, k)-diagnosability will result from Theorems 6 and 7 if a tighter upper bound of $I(\alpha)$ is used.

In order to apply Theorems 6 and 7, the values of $\hat{I}(\zeta_P)$ and $\hat{I}(\zeta_M)$ must be available. Previously, the values of $\hat{I}(\alpha)$ were computed for some particular graphs, e.g., Cartesian power of regular graphs [6], products of trees [7], powers of the Petersen graphs [5], and products of complete p-partite graphs [6]. Their (t, k)-diagnosability can be obtained as well by the aid of Theorems 6 and 7.

Appendix A. A table of important variables and notation.

$G = (V, A)$: A directed graph that represents a multiprocessor system
σ: A syndrome
$\Upsilon_P(x)$ ($\Upsilon_M(x)$): The set of neighboring vertices of x that are tested by x as fault-free under the PMC (MM*) model
$G_P^\dagger$ ($G_M^\dagger$): The spanning subgraph of G whose arcs are all (u, v)'s with $v \in \Upsilon_P(u)$ ($\Upsilon_M(u)$)
P-component (M-component): The vertex set of a strongly connected subgraph of $G_P^\dagger$ ($G_M^\dagger$)
C_P (C_M): The set of all P-components (M-components) in $G_P^\dagger$ ($G_M^\dagger$)
$C_M^{(1)}$: The set of all 1-vertex M-components in $G_M^\dagger$
ζ_P (ζ_M): A threshold for guaranteeing a fault-free aggregate under the PMC (MM*) model
Λ: An aggregate
Δ (δ): Maximal (minimal) degree of G
F: The set of faulty vertices in G
W: A fault-free aggregate in G
$N(v)$: The neighborhood of vertex v in G
$N(W)$: The neighborhood of W in G, i.e., $N(W) = \bigcup_{v \in W} N(v) - W$
$\pi(V_i)$: The set of all possible partitions of V_i
$\Pi(V')$: The set of all possible partitions of V' that are induced by further partitioning V_1, V_2, ..., V_r ($\{V_1, V_2, \ldots, V_r\}$ is a partition of V' so that the induced subgraph of each V_i in G is strongly connected and no arc exists between V_i and V_j if $i \neq j$), i.e., $\Pi(V') = \{\pi_1 \cup \pi_2 \cup \cdots \cup \pi_r : \pi_i \in \pi(V_i)$ for all $1 \leq i \leq r\}$

$\vartheta_{\Gamma,W}$: The total number of vertices in those elements of Γ that overlap with $N(W)$, where $W \subseteq V - V'$ and $\Gamma \in \Pi(V')$, i.e., $\vartheta_{\Gamma,W} = \Sigma_{Z\in\Gamma and\ Z\cap N(W)\neq\emptyset}\lvert Z\rvert$
$\xi_{V',W}$: Min$\{\vartheta_{\Gamma,W} : \Gamma \in \Pi(V')\}$
$\Phi(\chi_1, \chi_2)$: Refer to Definition 4
$I(\alpha)$: The maximal number of arcs whose two end vertices belong to a fixed α-vertex subset of V
$\hat{I}(\alpha)$: A convex function with $\hat{I}(0) = 0$ and $\hat{I}(\alpha) \geq I(\alpha)$
$W^{(i)}$: The fault-free aggregate identified in the ith iteration
$F^{(i)}$: The set of faulty vertices at the beginning of the ith iteration
$\mathcal{G}_d(n)$ ($\mathcal{T}_d(n)$): A d-dimesional grid (torus) with equal dimension length n

Appendix B. Proof of at least $\ddot{q}$ faulty vertices identified when $W^{(i)} = V - F^{(i)}$. Suppose conversely that there exists a syndrome σ that may induce fewer than $\ddot{q}$ faulty vertices identified in the m'th iteration, where $m \leq m' < n$. Refer to Figure 7, where $F^{(m')} - F^{(m'+1)}$ is the fault set identified in the m'th iteration, $F^{(m'+1)} - F^{(m'+2)}$ is the fault set identified in the $(m'+1)$th iteration, and so on. Let $X(\subseteq F^{(n)})$ be a faulty M-component (in $G_M^\dagger$) identified in the last iteration. Each M-component that is neighboring to X in $G_M^\dagger$ is contained in $F^{(n)}$ or $F^{(n-1)} - F^{(n)}$; i.e., all vertices in $N(x)$ are faulty if $x \in X$.

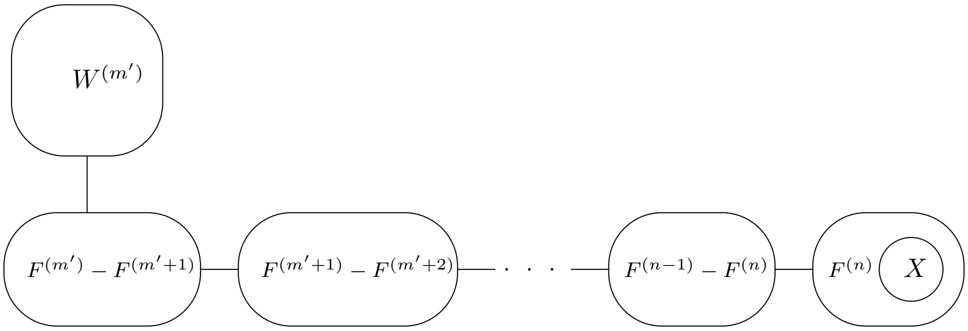


FIG. 7. *An illustrative example.*

Let us construct $\tilde{\sigma}$ from σ as follows: set $\tilde{\sigma}(y, z : x) = 1$ if y or z is in $N(x) - X$ and x is in X, and $\tilde{\sigma}(y, z : x) = \sigma(y, z : x)$ otherwise. Notice that $\tilde{\sigma}$ is a syndrome that may be generated by $F^{(1)}$. Besides, in each iteration, the fault-free M-components induced by $\tilde{\sigma}$ are identical with those induced by σ. It follows that in each iteration, each fault-free aggregate identified with respect to $\tilde{\sigma}$ is also identifiable with respect to σ. In other words, each diagnosis process induced by $\tilde{\sigma}$ can be induced by σ as well.

Notice that $\tilde{\sigma}$ can also be generated by $F^{'(1)} = F^{(1)} - X$ (i.e., considering X fault-free). It follows that with the same syndrome $\tilde{\sigma}$, the fault-free aggregate and the faulty vertices identified in the m'th iteration with respect to $F^{(1)}$ are also identifiable in the m'th iteration with respect to $F^{'(1)}$. Let $F^{'(m')}$ be the resulting fault set of G at the beginning of the m'th iteration and $W^{'(m')}$ be the fault-free aggregate identified in the m'th iteration when $F^{'(1)}$ is the initial fault set of G and $\tilde{\sigma}$ is the syndrome generated by $F^{'(1)}$. Since X is fault-free, we have $W^{'(m')} \subset V - F^{'(m')}$ (or $W^{'(m')} \cup X = V - F^{'(m')}$). Then, according to Definition 4, there are at least $\ddot{q}$ faulty vertices contained in the neighboring M-components of $W^{'(m')}$; i.e., at least $\ddot{q}$ faulty vertices can be identified. It is implied that the syndrome σ generated by $F^{(1)}$ can also induce a diagnosis process that will identify at least $\ddot{q}$ faulty vertices in the m'th iteration, which is a contradiction.

Appendix C. Proof of Lemma 10. We denote each vertex v of $\mathcal{T}_d(n)$ by a d-tuple $(v_1, v_2, \ldots, v_d)$, where each v_i, ranging from 0 to $n-1$, is the ith coordinate of v. Suppose that H is a set of α vertices in $\mathcal{T}_d(n)$. We first recoordinate the vertices of $\mathcal{T}_d(n)$ so that

(1) the minimum of the ith coordinates of all vertices in H is 0,
(2) the maximum of the ith coordinates of all vertices in H is x_i,
(3) x_i obtained in (2) is minimum, i.e., there is no coordinate system of $\mathcal{T}_d(n)$ in which there exists $x'_i < x_i$ and for each $v = (v_1, v_2, \ldots, v_d) \in H$, $0 \le v_i \le x'_i$, and
(4) $0 \le x_1 \le x_2 \le \cdots \le x_d \le n-1$,

where $1 \le i \le d$.

Along a fixed dimension, say, γ, we first partition $\mathcal{T}_d(n)$ into n $\mathcal{T}_{d-1}(n)$'s, denoted by $T_1, T_2, \ldots, T_n$, where $1 \le \gamma \le d$ and each T_i contains all the vertices of $\mathcal{T}_d(n)$ whose γth coordinates are i. Without loss of generality, assume $x_1 \le x_2 \le \cdots \le x_\mu < n-1$ and $x_{\mu+1} = x_{\mu+2} = \cdots = x_d = n-1$, where $0 < \mu \le d$. If $1 \le \gamma \le \mu$, then $T_{x_\gamma+1}$ contains no vertex in H. In order to estimate $\hat{I}(\alpha)$, we intend to minimize the number of arcs that each have one end vertex in H and the other end vertex not in H, as elaborated below.

Let $\mathcal{E}_{H,\gamma}(v'_1, \ldots, v'_{\gamma-1}, v'_{\gamma+1}, \ldots, v'_d) = \{(v'_1, \ldots, v'_{\gamma-1}, u_{\gamma,q_1}, v'_{\gamma+1}, \ldots, v'_d), (v'_1, \ldots, v'_{\gamma-1}, u_{\gamma,q_2}, v'_{\gamma+1}, \ldots, v'_d), \ldots, (v'_1, \ldots, v'_{\gamma-1}, u_{\gamma,q_p}, v'_{\gamma+1}, \ldots, v'_d)\}$ be the set of vertices in H that have the same jth coordinate v'_j for $j \in \{1, \ldots, \gamma-1, \gamma+1, \ldots, d\}$ and have different γth coordinates $u_{\gamma,q_1} \le u_{\gamma,q_2} \le \cdots \le u_{\gamma,q_p}$. When $(u_{\gamma,q_1} - 1) \bmod n \notin \{u_{\gamma,q_1}, u_{\gamma,q_2}, \ldots, u_{\gamma,q_p}\}$ (or $(u_{\gamma,q_p} + 1) \bmod n \notin \{u_{\gamma,q_1}, u_{\gamma,q_2}, \ldots, u_{\gamma,q_p}\}$), the arc $((v'_1, \ldots, v'_{\gamma-1}, u_{\gamma,q_1}, v'_{\gamma+1}, \ldots, v'_d), (v'_1, \ldots, v'_{\gamma-1}, (u_{\gamma,q_1} - 1) \bmod n, v'_{\gamma+1}, \ldots, v'_d))$ (or $((v'_1, \ldots, v'_{\gamma-1}, u_{\gamma,q_p}, v'_{\gamma+1}, \ldots, v'_d), (v'_1, \ldots, v'_{\gamma-1}, (u_{\gamma,q_p} + 1) \bmod n, v'_{\gamma+1}, \ldots, v'_d)))$ is referred to as an *out-arc* (thus named because $(u_{\gamma,q_1} - 1) \bmod n$ (or $(u_{\gamma,q_p} + 1) \bmod n$) is out of the range from u_{γ,q_1} to u_{γ,q_p}).

On the other hand, when $(u_{\gamma,q_i} - 1) \bmod n \notin \{u_{\gamma,q_1}, u_{\gamma,q_2}, \ldots, u_{\gamma,q_p}\}$ and $u_{\gamma,q_1} \le (u_{\gamma,q_i} - 1) \bmod n \le u_{\gamma,q_p}$ (or $(u_{\gamma,q_i} + 1) \bmod n \notin \{u_{\gamma,q_1}, u_{\gamma,q_2}, \ldots, u_{\gamma,q_p}\}$ and $u_{\gamma,q_1} \le (u_{\gamma,q_i} + 1) \bmod n \le u_{\gamma,q_p}$), the arc $((v'_1, \ldots, v'_{\gamma-1}, u_{\gamma,q_i}, v'_{\gamma+1}, \ldots, v'_d), (v'_1, \ldots, v'_{\gamma-1}, (u_{\gamma,q_i} - 1) \bmod n, v'_{\gamma+1}, \ldots, v'_d))$ (or $((v'_1, \ldots, v'_{\gamma-1}, u_{\gamma,q_i}, v'_{\gamma+1}, \ldots, v'_d), (v'_1, \ldots, v'_{\gamma-1}, (u_{\gamma,q_i} + 1) \bmod n, v'_{\gamma+1}, \ldots, v'_d)))$ is referred to as an *in-arc*. Both out-arcs and in-arcs are directed from vertices in H to vertices not in H. A lower bound on the number of out-arcs is estimated below.

When $1 \le \gamma \le \mu$, there are at least $2 * |\{(v_1, \ldots, v_{\gamma-1}, v_{\gamma+1}, \ldots, v_d) : \mathcal{E}_{H,\gamma}(v_1, \ldots, v_{\gamma-1}, v_{\gamma+1}, \ldots, v_d) \ne \emptyset\}|$ arcs that each have one end vertex in H and the other end vertex not in H, because $(u_{\gamma,q_1} - 1) \bmod n \notin \{u_{\gamma,q_1}, u_{\gamma,q_2}, \ldots, u_{\gamma,q_p}\}$ and $(u_{\gamma,q_p} + 1) \bmod n \notin \{u_{\gamma,q_1}, u_{\gamma,q_2}, \ldots, u_{\gamma,q_p}\}$. Clearly, $2 * |\{(v_1, \ldots, v_{\gamma-1}, v_{\gamma+1}, \ldots, v_d) \colon \mathcal{E}_{H,\gamma}(v_1, \ldots, v_{\gamma-1}, v_{\gamma+1}, \ldots, v_d) \ne \emptyset\}| \ge \min\{|H_i| : 1 \le i \le n\}$, where H_i is the set of vertices in H that are located in T_i. The equality holds when $|H_1| = |H_2| = \cdots = |H_n|$ and for $i \ne j$, if $v \in H_i$, then there exists $u \in H_j$ so that the coordinates of u and v differ only in dimension γ.

When $\mu < \gamma \le d$, there are at least $|\{(v_1, \ldots, v_{\gamma-1}, v_{\gamma+1}, \ldots, v_d) : \mathcal{E}_{H,\gamma}(v_1, \ldots, v_{\gamma-1}, v_{\gamma+1}, \ldots, v_d) \ne \emptyset$ and $(v_1, \ldots, v_{\gamma-1}, 0, v_{\gamma+1}, \ldots, v_d) \notin H\}| + |\{(v_1, \ldots, v_{\gamma-1}, v_{\gamma+1}, \ldots, v_d) : \mathcal{E}_{H,\gamma}(v_1, \ldots, v_{\gamma-1}, v_{\gamma+1}, \ldots, v_d) \ne \emptyset$ and $(v_1, \ldots, v_{\gamma-1}, n-1, v_{\gamma+1}, \ldots, v_d) \notin H\}| \ge 0$ arcs that each have one end vertex in H and the other end vertex not in H. The equality holds when $u_{\gamma,q_1} = 0$ and $u_{\gamma,q_p} = n-1$ for every $\mathcal{E}_{H,\gamma}(v_1, \ldots,$

$v_{\gamma-1}, v_{\gamma+1}, \ldots, v_d) \neq \emptyset$.

A lower bound on the number of in-arcs is estimated as follows. There are at least $2 \times \Sigma_{i=1}^{p-1} |\{(v_1, \ldots, v_{\gamma-1}, u_{\gamma,q_i}, v_{\gamma+1}, \ldots, v_d) : (v_1, \ldots, v_{\gamma-1}, u_{\gamma,q_i}, v_{\gamma+1}, \ldots, v_d) \in H$ and $(v_1, \ldots, v_{\gamma-1}, u_{\gamma,q_i} + 1, v_{\gamma+1}, \ldots, v_d) \notin H\}| \geq 0$ arcs that each have one end vertex in H and the other end vertex not in H. The equality holds when $u_{\gamma,q_i} + 1 = u_{\gamma,q_{i+1}}$ for all $1 \leq i \leq p-1$.

When the three equalities above hold, H forms a d-dimensional hyperrectangle with $|H| = y_1 \times y_2 \times \cdots \times y_\mu \times n^{d-\mu}$, where each y_j is the side length of the hyperrectangle along the dimension j. Now that $\min\{|H_i| : 1 \leq i \leq n\} = \frac{\alpha}{y_\gamma}$, we have

$$I(\alpha) \leq 2 \times d \times \alpha - 2 \times \min\left\{\sum_{j=1}^{\mu} \frac{\alpha}{y_j} : 1 \leq \mu \leq d\right\}.$$

Notice that $\frac{1}{y_1} + \frac{1}{y_2} + \cdots + \frac{1}{y_\mu} \geq \mu \times \sqrt[\mu]{\frac{1}{y_1 \times y_2 \times \cdots \times y_\mu}} = \mu \times (\Pi_{j=1}^{\mu} y_j)^{\frac{-1}{\mu}}$. Since the equality holds when $y_1 = y_2 = \cdots = y_\mu = (\frac{\alpha}{n^{d-\mu}})^{\frac{1}{\mu}}$, we have $\Sigma_{j=1}^{\mu} \frac{\alpha}{y_j} \geq \alpha \times (\mu \times ((\frac{\alpha}{n^{d-\mu}})^{\frac{1}{\mu} \times \mu})^{\frac{-1}{\mu}}) = \alpha \times \mu \times (\frac{\alpha}{n^{d-\mu}})^{\frac{-1}{\mu}} = n^{\frac{d-\mu}{\mu}} \times \mu \times \alpha^{\frac{\mu-1}{\mu}}$ whose first-order derivative with respect to μ is 0 at $\mu = \ln(\frac{n^d}{\alpha})$ and whose second-order derivative with respect to μ is positive. Hence, $n^{\frac{d-\mu}{\mu}} \times \mu \times \alpha^{\frac{\mu-1}{\mu}}$ is minimum when $\mu = \ln(\frac{n^d}{\alpha})$. That is, $I(\alpha) \leq 2 \times d \times \alpha - 2 \times n^{\frac{d-\mu}{\mu}} \times \mu \times \alpha^{\frac{\mu-1}{\mu}}$ with $\mu = \ln(\frac{n^d}{\alpha})$.

REFERENCES

[1] T. Araki and Y. Shibata, *(t,k)-diagnosable system: A generalization of the PMC models*, IEEE Trans. Comput., 52 (2003), pp. 971–975.

[2] T. Araki and Y. Shibata, *Diagnosability of butterfly networks under the comparison approach*, IEICE Trans. Fundamentals of Electronics Communications and Computer Science, E85-A (2002), pp. 1152–1160.

[3] J. R. Armstrong and F. G. Gray, *Fault diagnosis in a Boolean n-cube array of microprocessors*, IEEE Trans. Comput., C-30 (1981), pp. 587–590.

[4] M. C. Azizoğlu and Ö. Eğecioğlu, *The bisection width and the isoperimetric number of arrays*, Discrete Appl. Math., 138 (2004), pp. 3–12.

[5] S. L. Bezrukov, S. K. Das, and R. Elsässer, *An edge-isoperimetric problem for powers of the Petersen graph*, Ann. Comb., 4 (2000), pp. 153–169.

[6] S. L. Bezrukov and R. Elsässer, *Edge-isoperimetric problems for Cartesian powers of regular graphs*, Theoret. Comput. Sci., 307 (2003), pp. 473–492.

[7] S. L. Bezrukov, *On an equivalence in discrete extremal problems*, Discrete Math., 203 (1999), pp. 9–22.

[8] L. Bhuyan and D. P. Agrawal, *Generalized hypercube and hyperbus structures for a computer network*, IEEE Trans. Comput., 33 (1984), pp. 323–333.

[9] B. Bollobas and I. Leader, *An isoperimetric inequality on the discrete torus*, SIAM J. Discrete Math., 3 (1990), pp. 32–37.

[10] A. Caruso, S. Chessa, P. Maestrini, and P. Santi, *Diagnosability of regular systems*, J. Algorithms, 45 (2002), pp. 126–143.

[11] C. P. Chang, P. L. Lai, J. M. Tan, and L. H. Hsu, *Diagnosability of t-connected networks and product networks under the comparison diagnosis model*, IEEE Trans. Comput., 53 (2004), pp. 1582–1590.

[12] G. Y. Chang, G. H. Chen, and G. J. Chang, *(t, k)-diagnosis for matching composition networks*, IEEE Trans. Comput., 55 (2006), pp. 88–92.

[13] G. Y. Chang, G. H. Chen, and G. J. Chang, *(t, k)-diagnosis for matching composition networks under the MM* model*, IEEE Trans. Comput., 56 (2007), pp. 73–79.

[14] A. T. Dahabura and G. M. Masson, *An $O(n^{2.5})$ fault identification algorithm for diagnosable system*, IEEE Trans. Comput., 33 (1984), pp. 486–492.

[15] J. Fan, *Diagnosability of the Möbius cubes*, IEEE Trans. Parallel Distributed Systems, 9 (1998), pp. 923–927.

[16] J. Fan, *Diagnosability of crossed cubes under the comparison diagnosis model*, IEEE Trans. Parallel Distributed Systems, 13 (2002), pp. 687–692.

[17] A. D. Friedman and L. Simoncini, *System-level fault diagnosis*, Comput. J., 13 (1980), pp. 47–53.

[18] H. Fujiwara and K. Kinoshita, *On the computational complexity of system diagnosis*, IEEE Trans. Comput., 27 (1978), pp. 881–885.

[19] S. L. Hakimi and A. T. Amin, *Characterization of connection assignment*, IEEE Trans. Comput., 23 (1974), pp. 86–88.

[20] A. Kavianpour, *Sequential diagnosability of star graphs*, Computers & Electrical Engineering, 22 (1996), pp. 37–44.

[21] A. Kavianpour and K. H. Kim, *Diagnosabilities of hypercubes under the pessimistic one-step diagnosis strategy*, IEEE Trans. Comput., 40 (1991), pp. 232–237.

[22] S. Khanna and W. K. Fuchs, *A linear time algorithm for sequential diagnosis in hypercubes*, J. Parallel Distributed Computing, 26 (1995), pp. 38–53.

[23] S. Khanna and W. K. Fuchs, *A graph partitioning approach to sequential diagnosis*, IEEE Trans. Comput., 46 (1997), pp. 39–47.

[24] P. L. Lai, J. M. Tan, C. H. Tsai, and L. H. Hsu, *The diagnosability of the matching composition network under the comparison diagnosis model*, IEEE Trans. Comput., 53 (2004), pp. 1064–1069.

[25] P. L. Lai, J. M. Tan, C. P. Chang, and L. H. Hsu, *Conditional diagnosability measures for large multiprocessor systems*, IEEE Trans. Comput., 54 (2005), pp. 165–175.

[26] I. Leader, *Discrete isoperimetric inequalities*, in Probabilistic Combinatonics and Its Applications, Proc. Sympos. Appl. Math. 44, AMS, Providence, RI, 1991, pp. 57–80.

[27] F. P. Preparata, G. Metze, and R. T. Chien, *On the connection assignment problem of diagnosable systems*, IEEE Trans. Electronic Comput., EC-16 (1967), pp. 848–854.

[28] P. Santi and S. Chessa, *Reducing the number of sequential diagnosis iterations in hypercubes*, IEEE Trans. Comput., 43 (2004), pp. 89–92.

[29] A. Sengupta and A. T. Dahbura, *On self-diagnosable multiprocessor systems: Diagnosis by the comparison approach*, IEEE Trans. Comput., 41 (1992), pp. 1386–1396.

[30] D. Wang, *Diagnosability of enhanced hypercubes*, IEEE Trans. Comput., 43 (1994), pp. 1054–1061.

[31] D. Wang, *Diagnosability of hypercubes and enhanced hypercubes under the comparison diagnosis model*, IEEE Trans. Comput., 48 (1999), pp. 1369–1374.

[32] D. B. West, *Introduction to Graph Theory*, 2nd ed., Prentice Hall, Upper Saddle River, NJ, 2001.

SIAM J. Comput.
Vol. 37, No. 5, pp. 1299–1318

ORACLES FOR DISTANCES AVOIDING A FAILED NODE OR LINK*

CAMIL DEMETRESCU†, MIKKEL THORUP‡, REZAUL ALAM CHOWDHURY§, AND VIJAYA RAMACHANDRAN§

Abstract. We consider the problem of preprocessing an edge-weighted directed graph G to answer queries that ask for the length and first hop of a shortest path from any given vertex x to any given vertex y avoiding any given vertex or edge. As a natural application, this problem models routing in networks subject to node or link failures. We describe a deterministic oracle with constant query time for this problem that uses $O(n^2 \log n)$ space, where n is the number of vertices in G. The construction time for our oracle is $O(mn^2 + n^3 \log n)$. However, if one is willing to settle for $\Theta(n^{2.5})$ space, we can improve the preprocessing time to $O(mn^{1.5} + n^{2.5} \log n)$ while maintaining the constant query time. Our algorithms can find the shortest path avoiding a failed node or link in time proportional to the length of the path.

Key words. graph algorithms, data structures, shortest paths, network failures

AMS subject classifications. 05C85, 68W01, 68P05, 05C38, 90B18

DOI. 10.1137/S0097539705429847

1. Introduction.

In the *distance sensitivity* problem, we wish to construct a data structure (which we call *distance sensitivity oracle*) for an edge-weighted directed graph G that supports the following queries:

`v-dist`(x, y, v):	returns the distance from vertex x to vertex y in $G - \{ v \}$, i.e., the length (sum of edge weights) of the shortest possible path from x to y in G avoiding vertex v, if one exists, and $+\infty$ otherwise.
`e-dist`(x, y, u, v):	returns the distance from vertex x to vertex y in $G - \{ (u, v) \}$, i.e., the length (sum of edge weights) of the shortest possible path from x to y in G avoiding edge (u, v), if one exists, and $+\infty$ otherwise.

We also consider the corresponding path queries, which we denote by `v-path`(x, y, v) and `e-path`(x, y, u, v), respectively. In this article, we denote by n the number of vertices and by m the number of edges in G. We also assume that edge weights are nonnegative. In our bounds, space is measured as the number of memory words,

*Received by the editors May 22, 2005; accepted for publication (in revised form) July 11, 2007; published electronically January 4, 2008. Part of the results presented in this article for link failures appeared in the Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'02) and in the Proceedings of the 13th International Symposium on Algorithms and Computation (ISAAC'02).

http://www.siam.org/journals/sicomp/37-5/42984.html

†Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza," via Ariosto 25, 00185 Rome, Italy (demetres@dis.uniroma1.it, http://www.dis.uniroma1.it/~demetres). This author's research was supported in part by the Italian Ministry of University and Scientific Research (Project "MAINSTREAM: Algorithms for massive information structures and data streams").

‡AT& T Labs-Research, 180 Park Avenue, Florham Park, NJ 07932 (mthorup@research.att.com, http://www.research.att.com/info/mthorup).

§Department of Computer Sciences, The University of Texas at Austin, Austin, TX 78712 (shaikat@cs.utexas.edu, http://www.cs.utexas.edu/~shaikat, vlr@cs.utexas.edu, http://www.cs.utexas.edu/~vlr). The third author's research was supported in part by an MCD Graduate Fellowship and NSF grant CCR-9988160. The fourth author's research was supported in part by Texas Advanced Research Program grant 003658-0029-1999 and NSF grants CCR-9988160 and CCF-0514876.

where each word can hold the label of a vertex or edge or the weight of an edge or path.

In our directed case, we note that edge failures subsume vertex failures. The reduction is well known from work on disjoint paths. We split each vertex v into an in-vertex v_0 and an out-vertex v_1, with an in-out edge (v_0, v_1). The original incoming edges are moved to v_0 and the outgoing edges are moved to v_1. Now losing the edge (v_0, v_1) due to a network failure has the same effect as losing vertex v in the original graph.

1.1. Motivation. Our motivating scenario is a network where node/link failures happen quite rarely. As soon as a node or link failure has been noticed, we want to be able to answer distance queries and provide directions for shortest paths in the network without the failed node or link. On the other hand, we assume that we have plenty of time to compute a new data structure in the background. We model a network as a weighted directed graph where vertices correspond to network nodes, and edges correspond to network links. In this scenario, $\texttt{v-dist}(x, y, v)$ yields the distance from node x to node y in the network avoiding failed node v, and $\texttt{e-dist}(x, y, u, v)$ yields the distance from node x to node y in the network avoiding failed link (u, v).

We note that the ability to deal with node/link failures enables us to deal with some other related aspects of the network. For example, by dealing with a link failure (u, v), we actually deal with arbitrary changes to its weight. More precisely, we can simply compute the distance from vertex x to vertex y in the graph where (u, v) has its weight changed to w as $\min\{\texttt{e-dist}(x, y, u, v), d_{xu} + w + d_{vy}\}$, where d_{xu} is the distance from vertex x to vertex u and d_{vy} is the distance from vertex v to vertex y in the graph. Here, a weight change could model that traffic is moving slower/faster along a certain link. An interesting application of dealing with single weight changes is a local search like the one in [9]. There, one wants to consider a neighborhood of a given weight setting, where each neighbor is obtained by changing a single weight.

Another motivation for solving the distance sensitivity problem arises from recent interest in Vickrey pricing of networks [11, 19]. We describe this application in more detail in section 1.4.

1.2. Related work. A variant of the distance sensitivity problem, related to reachability in directed acyclic graphs w.r.t. edge failures, was first introduced by King and Sagert in [14], where they consider the problem of supporting *sensitivity queries* of the kind: “Is there a path from vertex x to vertex y that does not contain edge (u, v)?” Here, we are concerned with general weighted digraphs and distance queries instead of reachability queries, and we consider both vertex and edge failures.

This problem is similar to the *replacement paths* problem [11] (see also the erratum for [11] and [12, 18, 21]) which, given a pair of vertices x and y in G, computes the set of shortest paths from x to y avoiding each of the vertices (or edges) on π_{xy} one at a time, where π_{xy} is the original shortest path from x to y in G. A method for solving this problem on undirected graphs in $O(m + n \log n)$ preprocessing time and $O(n)$ space is given for the vertex removal case in [18] and for the edge removal case in [11]. A method for solving this problem on unweighted directed graphs in $\tilde{O}(m\sqrt{n})$ preprocessing time and $\tilde{O}(n^{3/2})$ space is given in [21].

However, in this paper we are interested in finding replacement paths for all possible sources and destinations, blowing up each of the above bounds by a factor n^2. In particular, the space becomes $O(n^3)$. Moreover, we consider the general case of a weighted directed graph.

Of a similar flavor is the *most vital node* (or *arc*) problem [1, 2, 4, 18], which is the problem of identifying the vertex (or edge) on a given shortest path, whose removal results in the longest replacement path.

The most natural approach to the distance sensitivity problem would be to use one of the recent *dynamic* all pairs shortest paths (APSP) algorithms [5, 6, 13, 15] and delete the failed vertex or edge. The best bounds [6, 25] take $\widetilde{O}(n^2)$ amortized time for real weighted directed graphs, but then queries avoiding the failed vertex or edge are answered in constant time. However, our goal here is to answer a query as quickly as possible after a vertex or edge failure, and then it may be faster to compute the answer from scratch at each query using an $O(n \log n + m)$ single-source shortest paths (SSSP) algorithm [10].

Another extreme solution would be to construct a table that for each vertex pair (x, y) and each vertex/edge stores the distance from x to y avoiding that vertex/edge. For vertex failures, such a table of size $O(n^3)$ is trivially computed by n APSP computations in $O(mn^2 + n^3 \log n)$ time. However, for edge failures the size of the trivial table is $\Theta(mn^2)$ and requires m APSP computations. Space can be reduced (at least for the edge failure case) by working from one source x at a time, constructing a shortest paths tree $T(x)$. This tree changes only if we remove any of the $O(n)$ vertices or edges in it. Hence, it is only for these vertices and edges that we need to record new distances from x. An implementation of this idea is given in [24]. If h is the maximal hop count of a path in $T(x)$, the construction uses h SSSP computations in $O(hm + hn \log n)$ time and $O(hn)$ space. However, we may have $h = \Omega(n)$, and repeating the construction from all sources, we end up with a construction time of $O(mn^2 + n^3 \log n)$ and a space bound of $O(n^3)$ for our all pairs case. The fundamental question considered here is if the cubic space bound can be improved.

Shortest paths computation. In the above time bounds, we have assumed that SSSP is solved in $O(m + n \log n)$ time using Fibonacci heaps in Dijkstra's SSSP algorithm [10] and that APSP is solved with an SSSP from each vertex in $O(mn + n^2 \log n)$ time. We will make the same assumptions when stating our own results.

We note here that there are alternative faster algorithms in different situations. For example, if the weights are represented as integer or floating point numbers, we can compute SSSP in $O(m + n \log \log n)$ time [26], or even in $O(m)$ time if the graph is undirected [23]. In the case of real weights, we can solve APSP in $O(mn + n^2 \log \log n)$ time for sparse graphs [20] and in $O(n^3 \sqrt{\log \log n} / \log n)$ time for dense graphs [22, 27].

However, the above improvements are "only" by logarithmic factors, and in our results for distance sensitivity oracles, we are aiming at polynomial improvements in space and construction time. Hence we are satisfied just stating the time bounds assuming Fibonacci heaps in Dijkstra's algorithm.

1.3. Our results. The main result of this article is a deterministic oracle with fast query time for both vertex and edge failures that uses nearly the same space as that required for storing the distance matrix of the input graph. More precisely, we construct an oracle that uses $O(n^2 \log n)$ space and answers distance queries subject to a vertex or edge failure in $O(1)$ worst-case time. This result is quite surprising, since the space bound is significantly smaller than $\Theta(n^3)$ and yet our scheme answers queries in $O(1)$ time. We also present an $\Omega(m)$ space lower bound for the single-source version of the problem. Since m can be as high as $\Omega(n^2)$, our oracle is thus almost space-optimal.

The construction time for our oracle is $O(mn^2 + n^3 \log n)$ in the worst case. However, if one is willing to settle for $\Theta(n^{2.5})$ space, we can improve the preprocessing

Table 1.1
Known results and our contribution.

References	Context	Graph type	Construction time	Space	Query time
Nardelli et al. [18]	Most vital node detection [$\texttt{v-dist}(x, y, v)$]	Undirected	$O(mn^2 + n^3 \log n)$	$O(n^3)$	$O(1)$
Hershberger & Suri [11]	Vickrey pricing in networks [$\texttt{e-dist}(x, y, u, v)$]	Undirected	$O(mn^2 + n^3 \log n)$	$O(n^3)$	$O(1)$
Our contribution	Vertex/edge failure [$\texttt{v-dist}(x, y, v)$] [$\texttt{e-dist}(x, y, u, v)$]	Directed (or undirected)	Method 1		
			$O(mn^2 + n^3 \log n)$	$O(n^2 \log n)$	$O(1)$
			Method 2		
			$O(mn^{1.5} + n^{2.5} \log n)$	$O(n^{2.5})$	$O(1)$

time to $O(mn^{1.5} + n^{2.5} \log n)$, and the query time remains constant. In Table 1.1 we place our bounds in perspective by comparing them to the bounds for related problems obtainable with previous algorithms: in the context of most vital node detection and Vickrey pricing, we are extrapolating the performance of algorithms designed for a single source-destination pair to the all pairs case.

To achieve our bounds, we construct data structures where we store information about APSP excluding only vertices or edges with specific properties, rather than excluding all possible vertices or edges. We also store information about shortest paths where we exclude vertices on entire subpaths, rather than single vertices. We remark that our algorithms are very simple and thus amenable to efficient implementations.

We note that vertex failures in directed graphs trivially subsume vertex failures in undirected graphs, but the same does not hold for link failures as an undirected link failure corresponds to two symmetric directed link failures. However, our solutions happen to work for the undirected case as well.

Part of the results presented in this paper for link failures were presented in two conference papers: one by the first two authors [7] which presented the two algorithms, and the other by the last two authors [3]. The paper [3] improves the query time from [7], but it also has an extra claim on construction time which is incorrect. Additionally, this paper presents results obtained by the last two authors on extending the oracles in [7, 3] to efficiently handle node failures in addition to link failures. This paper also presents a lower bound result obtained by the first two authors on the space requirement for the single-source version of the distance oracle problem.

1.4. Vickrey pricing in networks. The Vickrey mechanism is a generalization of the *sealed bid second price* auction, in which the highest bidder wins the auction but pays a price equal to the second highest bid. This auction protocol motivates a rational bidder to bid truthfully [17]. In a distributed network in which multiple rational self-interested agents own different parts of the network, Vickrey mechanism is often the best way to determine the utility of various network elements. In order to elicit truthful responses from the agents, each agent is compensated in proportion to the marginal utility he/she brings to the network. Willing manipulations by participating

Table 1.2

Complexity of computing Vickrey payments for all vertex pairs.

Algorithm	Time	Space	Query time
Naïve	$O(n^2(m + n \log n))$	$O(n^3)$	$O(1)$
Our 1st result	$O(n^2(m + n \log n))$	$O(n^2 \log n)$	$O(1)$
Our 2nd result	$O(n^{1.5}(m + n \log n))$	$O(n^{2.5})$	$O(1)$

agents is eliminated by making an agent's payment depend only on the declarations of other agents.

Consider a scenario in which we need to find shortest paths in a network G, where links are owned by self-interested agents. Agents are assumed to bid on each link individually. Nisan and Ronen [19] formulated the following expression as the payment $p^e(x, y)$ to be made to the owner of a link e for a given vertex pair (x, y):

$$p^e(x,y) = \begin{cases} d_{xy}(G|_{w_e=\infty}) - d_{xy}(G|_{w_e=0}) & \text{if } e \in \pi_{xy}, \\ 0 & \text{otherwise,} \end{cases}$$

where π_{xy} is a shortest path from vertex x to vertex y in G, and $d_{xy}(G|_{w_e=k})$ is the distance from vertex x to vertex y in G, where the weight of edge e is set to k. For any $e \in \pi_{xy}$, the term $d_{xy}(G|_{w_e=0})$ can be simply computed as $d_{xy}(G|_{w_e=0}) = d_{xy} - w_e$, where d_{xy} is the distance from x to y in G. However, computing $d_{xy}(G|_{w_e=\infty})$ naïvely for all $e \in \pi_{xy}$ requires running an $O(m+n \log n)$ time shortest paths algorithm [8, 10] on $G - \{e\}$ for each $e \in \pi_{xy}$. This can be as high as $O(mn + n^2 \log n)$ in the worst case. This problem was studied in [11], but no improvement to this trivial bound is known.

The distance sensitivity problem we study in this article is a generalization of the above problem to the situation where one is potentially interested in finding all Vickrey payments for all vertex pairs (instead of a single pair). In this case our first algorithm can carry out the entire computation using significantly less space than that used by the naïve algorithm. On the other hand, our second algorithm reduces both time and space requirements of the computation. The complexities of all three algorithms are compared in Table 1.2. Observe, however, that for very sparse graphs the running time of the naïve algorithm can be improved to $O(mn(m + n \log \log n))$ [20].

1.5. Organization of the article. The remainder of this article is organized as follows. In section 2 we introduce the notation used in the article, and we discuss some simple properties that will be useful in the description of our results. In particular, we define the notion of "path cover," showing how to use information about shortest paths which avoid all vertices on certain paths in the graph to determine a shortest path that avoids a single vertex. In section 3 we show how to efficiently compute shortest paths from any given vertex to all other vertices in a directed graph G with nonnegative real-valued edge weights where we avoid all vertices on certain types of paths. These tools are used in section 4 to devise an oracle for the distance sensitivity problem that answers `v-dist` queries in constant worst-case time using nearly the same space required for storing a single distance matrix. This oracle can be constructed in $O(mn^2 + n^3 \log n)$ worst-case time. In section 5 we show that, if one is willing to settle for more space, we can reduce the preprocessing time to $O(mn^{1.5} + n^{2.5} \log n)$. This second oracle uses $O(n^{2.5})$ space while still answering `v-dist` queries in constant worst-case time. Section 6 shows how to extend the oracles designed for vertex failures to also deal with edge failures, and section 7 addresses the problem of supporting path queries `v-path` and `e-path`. A space lower bound for the single-source version of the distance sensitivity problem is discussed in section 8. Finally, section 9 provides some concluding remarks.

2. Preliminaries.

Let $G = (V, E, w)$ be a directed graph with vertex set V, edge set E, and edge weight function w. Throughout the article, we assume that, for each pair of vertices x and y such that y is reachable from x, there is a unique shortest path from x to y. This is without loss of generality, since ties can be broken arbitrarily (see, e.g., [6]).

2.1. Notation.

In this article, a path p_{xy} is a sequence of vertices of the form $p_{xy} = \langle v_0, v_1, v_2, \ldots, v_{k-1}, v_k \rangle$ such that $v_0 = x$, $v_k = y$, and $(v_i, v_{i+1}) \in E$ for every i, $0 \leq i < k$. Thus, $G - p_{xy}$ is the subgraph of G induced by the vertex set $V - \{x, v_1, \ldots, v_{k-1}, y\}$. We let $p_{xy} \cdot p_{yz}$ denote the concatenation of path p_{xy} with path p_{yz}. We denote by w_{xy} the weight of edge (x, y) in G, and we indicate with $w(p_{xy})$ the length of p_{xy}, i.e., the sum of weights of edges in p_{xy}. We also denote by $T(x)$ the single-source shortest path tree of G with source x, and we denote by π_{xy} the (unique) shortest path from vertex x to vertex y in G, if any. Using a geometrical analogy, we sometimes look at shortest paths as "segments," using the notation $[x, y]$ instead of π_{xy}. Similarly, we sometimes use $]x, y[$ to denote $\pi_{xy} - \{\, x, y \,\}$, $[x, y[$ to denote $\pi_{xy} - \{\, y \,\}$, and $]x, y]$ to denote $\pi_{xy} - \{\, x \,\}$. We indicate with d_{xy} the length of π_{xy} (distance from x to y in G), and with h_{xy} the number of edges in π_{xy} (number of hops). If y is not reachable from x, we assume that $d_{xy} = h_{xy} = +\infty$. By $\widehat{G}$ we denote the directed graph obtained from G by reversing the orientation of edges. Thus, $\widehat{\pi}$ and $\widehat{T}$ denote π and T in $\widehat{G}$. If T is a rooted tree, we let $B_T(i, j)$ denote the set of paths in T that connect vertices at level i with vertices at level $j > i$ in the tree, assuming that the root of T has level 0. Notice that, if $\pi_{cd} \in B_{T(x)}(i, j)$, then $h_{cd} = j - i$. Let a and b be vertices on p_{xy}; we say that $a < b$ if a appears before b on p_{xy}, and $a \leq b$ if $a < b$ or $a = b$. Finally, by π_{xy}^{uv} we denote a shortest path from vertex x to vertex y in $G - [u, v]$. The path π_{xy}^{uv} could be thought of as an optimal "replacement path" from x to y to be used in case all vertices in $[u, v]$ fail. Observe that, for each internal vertex v in π_{xy}, $\texttt{v-dist}(x, y, v) = w(\pi_{xy}^{vv})$. Indeed, the goal of this article is to provide methods for answering queries about π_{xy}^{vv}. The notation used in this article is summarized in Table 2.1.

2.2. Structural properties.

By our assumption of uniqueness of shortest paths, for any pair of vertices $a, b \in \pi_{xy}$ such that $a \leq b$, π_{ab} is a subpath of π_{xy}. Moreover, if $\pi_{xy} = \{v_0, v_1, \ldots, v_{k-1}, v_k\}$, then $\hat{\pi}_{yx} = \{v_k, v_{k-1}, \ldots, v_1, v_0\}$; i.e., one is the reversal of the other.

We now discuss a simple structural property of π_{xy}^{uv}. In particular, the following claim shows that, if y is reachable from x in $G - [u, v]$, then π_{xy}^{uv} and π_{xy} share a common prefix and a common suffix and are vertex-disjoint elsewhere (see Figure 2.1). Intuitively, the internal subpath of π_{xy}^{uv} that is vertex-disjoint from π_{xy} can be thought

Table 2.1
Notation used in this article.

Notation	Meaning
G	edge-weighted directed graph $G = (V, E, w)$
w_{xy}	weight of edge (x, y) in G
p_{xy}	path $\langle x, v_1, v_2, \cdots, v_{k-1}, y\rangle$ from vertex x to vertex y in G
$p_{xy} \cdot p_{yz}$	concatenation of path p_{xy} with path p_{yz}
$w(p_{xy})$	length of path p_{xy} (sum of weights of edges in p_{xy})
π_{xy} or $[x, y]$	shortest path from vertex x to vertex y in G (we assume it is unique)
$]x, y[$	$\pi_{xy} - \{\, x, y \,\}$
$[x, y[$	$\pi_{xy} - \{\, y \,\}$
$]x, y]$	$\pi_{xy} - \{\, x \,\}$
h_{xy}	number of edges in π_{xy}
d_{xy}	distance from vertex x to vertex y in G $\quad (d_{xy} = w(\pi_{xy}))$
$T(x)$	shortest path tree rooted at x in G (x is at level 0 in $T(x)$)
$\widehat{G}$	reversed graph $(V, \widehat{E}, \widehat{w})$ s.t. $\widehat{E} = \{(x, y) : (y, x) \in E\}$ and $\widehat{w}_{xy} = w_{yx}$
$\widehat{T}(x)$, $\widehat{\pi}_{xy}$	$T(x)$ and π_{xy} in $\widehat{G}$ instead of G
$B_T(i, j)$	set of all paths in tree T connecting vertices at level i to vertices at level j
$a < b$ in p_{xy}	vertex a precedes vertex b in p_{xy}
π^{uv}_{xy}	shortest path from x to y in $G - [u, v]$
$d_{xy}(l_1, l_2, r_1, r_2)$	$\min_{\substack{a \in\ [l_1, l_2[\\ b \in\ ]r_1, r_2]}} \{\, d_{xa} + w(\pi^{l_2 r_1}_{ab}) + d_{by} \,\}$ (see Definition 2.1)

of as an "optimal detour" that avoids vertices in $[u, v]$.

Claim 1. *Let $G = (V, E, w)$ be an edge-weighted directed graph. For any $x, y \in V$ and for any $u, v \in \pi_{xy}$ with $x < u \leq v < y$, if $\pi^{uv}_{xy} \neq \emptyset$, then there exist two vertices $a, b \in \pi_{xy}$ such that $\pi^{uv}_{xy} = \pi_{xa} \cdot p_{ab} \cdot \pi_{by}$, where $p_{ab} \cap \pi_{ab} = \{a, b\}$.*

Proof. We first notice that π_{xy} and π^{uv}_{xy} are both paths in G, since every path in $G - [u, v]$ is also a path in G. Moreover, $\pi_{xy} \neq \pi^{uv}_{xy}$, since $u \in \pi_{xy}$, but $u \notin \pi^{uv}_{xy}$.

Now, let p_{xa} be the longest common prefix of π^{uv}_{xy} and π_{xy}, and let p_{by} be their longest common suffix. These subpaths are never empty, since π^{uv}_{xy} and π_{xy} share at least their endpoints. This proves the existence of vertices a and b in our claim.

Since every subpath of π_{xy} is a shortest path (by the optimal substructure property of shortest paths) and shortest paths are unique in G, then $p_{xa} = \pi_{xa}$ and $p_{by} = \pi_{by}$. Furthermore, since $\pi_{xy} \neq \pi^{uv}_{xy}$, then $a \neq b$ and $p_{xa} \cap p_{by} = \emptyset$. Thus, we can write π^{uv}_{xy} as $\pi_{xa} \cdot p_{ab} \cdot \pi_{by}$ for some p_{ab}.

It remains to prove that $p_{ab} \cap \pi_{ab} = \{a, b\}$. Suppose by contradiction that there is a vertex $c \in p_{ab} \cap \pi_{ab}$ such that $a < c < b$. Now, observe that $c \notin [u, v]$, since $p_{ab} \cap [u, v] = \emptyset$, and that $[u, v]$ is a subpath of π_{ab}, since $\pi_{xa} \cap [u, v] = \emptyset$ and $\pi_{by} \cap [u, v] = \emptyset$. Assume without loss of generality that $c < u$ in π_{xy} (the case $v < c$ is completely analogous). Consider the subpath π_{ac} of π_{ab}, and notice that it is a shortest path in both G and $G - [u, v]$. Now, p_{ab} is a shortest path in $G - [u, v]$, and thus, by the optimal-substructure property, its subpath p_{ac} has to be shortest as well in $G - [u, v]$. Since shortest paths are unique in $G - [u, v]$, then $p_{ac} = \pi_{ac}$, and thus π_{xy} and π^{uv}_{xy}

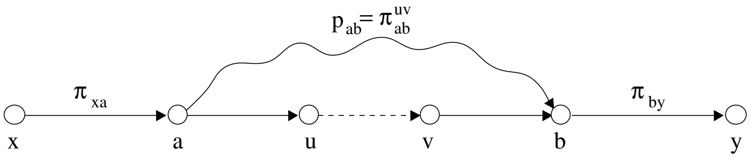


FIG. 2.1. *Structure of a replacement path* $\pi^{uv}_{xy} = \pi_{xa} \cdot p_{ab} \cdot \pi_{by}$.

share the same subpath from x to $c > a$. This means that a cannot be the last vertex of the longest common prefix of π^{uv}_{xy} and π_{xy}, which is clearly a contradiction. □

2.3. Path covering. We now discuss the notion of "path covering" that will be crucial to proving the correctness of our query algorithms.

DEFINITION 2.1. *Let $x \le l_1 < l_2 \le r_1 < r_2 \le y$ be vertices on π_{xy}. We define $d_{xy}(l_1, l_2, r_1, r_2)$ as*

$$d_{xy}(l_1, l_2, r_1, r_2) = \min_{\substack{a \in\ [l_1, l_2[\\ b \in\ ]r_1, r_2]}} \{\, d_{xa} + w(\pi^{l_2 r_1}_{ab}) + d_{by} \,\},$$

and we say that a value d covers *$[l_1, l_2[\ \times\ ]r_1, r_2]$ w.r.t. x, y if $d \le d_{xy}(l_1, l_2, r_1, r_2)$. In this case, we also say that d* covers *all paths of the form $\pi_{xa} \cdot \pi^{l_2 r_1}_{ab} \cdot \pi_{by}$ for each $a \in\ [l_1, l_2[$ and for each $b \in\ ]r_1, r_2]$.*

Observe that $d_{xy}(l_1, l_2, r_1, r_2)$ is the distance from vertex x to vertex y in G using paths that first follow a shortest path from x to some vertex a in $[l_1, l_2[$, then take an optimal detour that avoids all vertices in $[l_2, r_1]$, and finally go through a shortest path from some vertex b in $]r_1, r_2]$ to y. The following claim, which easily follows from Definition 2.1 and from Claim 1, states that, if $l_1 = x$ and $r_2 = y$, then $d_{xy}(l_1, l_2, r_1, r_2)$ is equal to the length of the shortest path from x to y avoiding $[u, v]$.

CLAIM 2. *Let $x < u \le v < y$ be vertices on π_{xy}. Then $w(\pi^{uv}_{xy}) = d_{xy}(x, u, v, y)$.*

We now show how information about the distance from x to y avoiding $[u, v]$ with detours having constrained positions of their endpoints can be used to compute $w(\pi^{uv}_{xy})$. The following claim will be useful to prove the correctness of our query algorithms.

CLAIM 3. *For any $x < \widehat{u} \le \widetilde{u} < u \le v < \widetilde{v} \le \widehat{v} < y$ on π_{xy},*

$$d_{xy}(x, u, v, y) \;=\; \min\{d_{xy}(x, \widetilde{u}, \widetilde{v}, y),\ d_{xy}(\widehat{u}, u, v, y),\ d_{xy}(x, u, v, \widehat{v})\}.$$

Proof. Let the endpoints of the optimal detour in π^{uv}_{xy} be a and b, $a < b$, and consider all possible positions of a in $[x, u[$:

- a in $[x, \widehat{u}[$: $d_1 = d_{xy}(x, u, v, \widehat{v})$ handles the cases when b lies in $]v, \widehat{v}]$; i.e., d_1 covers $[x, \widehat{u}[\ \times\ ]v, \widehat{v}]$ w.r.t. x, y. Moreover, $d_2 = d_{xy}(x, \widetilde{u}, \widetilde{v}, y)$ handles the cases when b lies in $]\widehat{v}, y]$; i.e., d_2 covers $[x, \widehat{u}[\ \times\ ]\widehat{v}, y]$ w.r.t. x, y. Thus, $d_3 = \min\{d_{xy}(x, u, v, \widehat{v}),\ d_{xy}(x, \widetilde{u}, \widetilde{v}, y)\}$ handles all possible positions of b in $]v, y]$; i.e., d_3 covers $[x, \widehat{u}[\ \times\ ]v, y]$ w.r.t. x, y.
- a in $[\widehat{u}, u[$: $d_4 = d_{xy}(\widehat{u}, u, v, y)$ handles the cases where b lies in $]v, y]$; i.e., d_4 covers $[\widehat{u}, u[\ \times\ ]v, y]$ w.r.t. x, y.

Thus, for each possible position of a in $[x, u[$, $d_5 = \min(d_{xy}(x, \widetilde{u}, \widetilde{v}, y),\ d_{xy}(\widehat{u}, u, v, y),\ d_{xy}(x, u, v, \widehat{v}))$ handles all possible positions of b in $]v, y]$; i.e., d_5 covers $[x, u[\ \times\ ]v, y]$ w.r.t. x, y. Since d_5 equals the length of some path from x to y in $G - [u, v]$, we can then conclude that $d_5 = d_{xy}(x, u, v, y)$. □

3. Distances under deletion of paths.

In this section we provide simple algorithms for computing distances in a directed graph G with nonnegative real-valued edge weights where we avoid all vertices on certain paths. These algorithms will be useful in sections 4 and 5 for constructing distance sensitivity oracles.

Let x be a vertex and let P be a set of shortest paths in G. We consider the problem of designing a procedure `exclude`(G, x, P) that computes for each path $\pi \in P$ the distances from vertex x to all other vertices in $G - \pi$. Throughout this article we assume that the deletion of the vertices on a given path π results in the deletion of all its vertices including its endpoints.

We can compute `exclude`(G, x, P) with a straightforward algorithm that runs in $O(|P|(m + n \log n))$ worst-case time using a Dijkstra computation [10] on the graph G with all vertices in π deleted for each $\pi \in P$. In the remainder of this section we show that this computation can be made considerably more efficient if we restrict our attention to $P \subseteq T(x)$ (i.e., every path in P is also a path in $T(x)$) and if we assume that paths in P are "independent," a notion we define in Definition 3.1.

3.1. Algorithm `fast-exclude`.

In this section we devise a variant of `exclude` (G, x, P), which we call `fast-exclude`(G, x, P), for the case when $P \subseteq T(x)$. As above, our goal is to compute for each path $\pi \in P$ the distances from vertex x to all other vertices in $G - \pi$.

Let $P \subseteq T(x)$, and for any path $\pi \in P$, denote by $T_x(\pi)$ the subtree of $T(x)$ rooted at the first vertex of π, and let W_π be the set of all vertices in $T_x(\pi)$ except the vertices on π. We observe that only vertices in W_π may have their distances from x increased if we remove from G the vertices on π. Now, consider the following directed graph $G_\pi = (V_\pi, E_\pi, w_\pi)$, where the following hold:

- $V_\pi = W_\pi \cup \{x\}$.
- E_π contains an edge from x to each vertex in W_π and the edges in G induced by vertices in W_π.
- $w_{\pi,ab}$ is the weight of edge (a, b) in G_π defined as

  $$w_{\pi,ab} = \begin{cases} \min_{c \notin T_x(\pi)} \{d_{xc} + w_{cb}\} & \text{if } a = x, \\ w_{ab} & \text{otherwise,} \end{cases}$$

  where we assume that $d_{xc} = +\infty$ if c is not reachable from x in G and $w_{cb} = +\infty$ if (c, b) is not an edge of G.

It is not difficult to see (see proof of Claim 4) that the shortest path from vertex x to a vertex v in W_π has the same length in $G - \pi$ as the shortest path from x to v in G_π. Hence, distances in $G - \pi$ from x to all vertices in W_π can be computed by a Dijkstra computation on G_π with source x. The algorithm `fast-exclude`(G, x, P) works in the same way as `exclude`(G, x, P), but it uses G_π instead of G for each π in P.

Since the graph G_π is typically smaller than G, `fast-exclude` can be expected to have better performance than `exclude` for any collection of paths $P \subseteq T(x)$. We now define the notion of *independent shortest paths*, and we show that `fast-exclude` performs significantly better than `exclude` when $P \subseteq T(x)$ is a collection of independent shortest paths.

Definition 3.1. *Let T be a rooted tree and let π_{uv} and $\pi_{u'v'}$ be two paths in T. We say that π_{uv} and $\pi_{u'v'}$ are* independent *in T if the subtree of T rooted at u and the subtree of T rooted at u' are disjoint.*

Claim 4. *If $P \subseteq T(x)$ is a set of pairwise independent shortest paths in $T(x)$, then algorithm* `fast-exclude`(G, x, P) *requires $O(m + n \log n)$ time in the worst case* *and computes the same output as* `exclude`$(G,\ x, P)$.

Proof. Using the notation given above, for a given $\pi \in P$, let π'_{xy} be the shortest path from x to any $y \in W_\pi$ avoiding the vertices on π. Let b be the first vertex on π'_{xy} such that $b \in W_\pi$, and let c be the vertex preceding b. Then let us write $\pi'_{xy} = p_{xc} \cdot \langle c, b\rangle \cdot \pi'_{by}$, where $\langle c, b\rangle$ is the path from c to b formed by the single edge (c, b). Since for any $z \notin T_x(\pi)$, the path π_{xz} is composed entirely of the vertices in $T(x) - T_x(\pi)$, it follows that $p_{xc} = \pi_{xc}$. Also note that π'_{by} cannot contain any vertex $z \notin T_x(\pi)$, since if it contains such a vertex z, then π_{xz} will be a shorter path to z, contradicting our choice of b. These two observations justify the use of G_π instead of $G - \pi$ in order to compute the shortest path tree rooted at x avoiding the vertices on π.

For each $\pi \in P$, $|V_\pi|$ is never greater than the number of vertices in $T_x(\pi)$. Since the paths in P are pairwise independent, any two such subtrees are disjoint for distinct paths in P. Since each E_π contains edges in G induced by vertices in W_π and edges from x to only the vertices in W_π, it follows that the sum of the cardinalities of all V_π and E_π are linear in n and m, respectively. Hence, `fast-exclude`(G, x, P) runs in $O(m + n \log n)$ time when $P \subseteq T(x)$ is a set of independent shortest paths. □

4. Oracle with $O(1)$ query time and $O(n^2 \log n)$ space.

In this section we describe a deterministic oracle for single-vertex failure with constant query time that uses nearly the same space as that required for storing a single distance matrix. In particular, we show how to preprocess a graph with nonnegative real-valued edge weights in $O(mn^2 + n^3 \log n)$ worst-case time, producing a compact oracle that uses $O(n^2 \log n)$ space and answers `v-dist` queries in $O(1)$ worst-case time.

4.1. Data structure.

Using $O(n^2 \log n)$ space, we maintain each d_{xy} and h_{xy}, and we maintain six matrices dl, dr, sl, sr, vl, and vr of size $n \times n \times \lfloor \log_2 n \rfloor$ defined for every pair of distinct vertices x and y as follows:

- $dl[x, y, i]$ = distance from vertex x to vertex y in $G - \pi$, where π is the subpath of π_{xy} starting at level 2^{i-1} and ending at level $2^i - 1$ in $T(x)$, and $1 \le i \le \log_2 h_{xy}$;
- $dr[x, y, i]$ = distance from vertex y to vertex x in $\widehat{G} - \pi$, where π is the subpath of $\widehat{\pi}_{yx}$ starting at level 2^{i-1} and ending at level $2^i - 1$ in $\widehat{T}(y)$, and $1 \le i \le \log_2 h_{xy}$;
- $sl[x, y, i]$ = distance from vertex x to vertex y in $G - \{v\}$, where v is the vertex of π_{xy} at level 2^{i-1} in $T(x)$, and $1 \le i < 1 + \log_2 h_{xy}$;
- $sr[x, y, i]$ = distance from vertex y to vertex x in $\widehat{G} - \{v\}$, where v is the vertex of $\widehat{\pi}_{yx}$ at level 2^{i-1} in $\widehat{T}(y)$, and $1 \le i < 1 + \log_2 h_{xy}$;
- $vl[x, y, i]$ = vertex of π_{xy} at level 2^{i-1} in $T(x)$, where $1 \le i \le 1 + \log_2 h_{xy}$;
- $vr[x, y, i]$ = vertex of $\widehat{\pi}_{yx}$ at level 2^{i-1} in $\widehat{T}(y)$, where $1 \le i \le 1 + \log_2 h_{xy}$.

4.2. Preprocessing.

The above quantities are computed in the preprocessing phase as follows:

- Distances d_{xy} and h_{xy} and matrices vl and vr are easily initialized from shortest path trees of G.
- To compute $dl[x, y, i]$, we call procedure `exclude`$(G, x, B_{T(x)}(2^{i-1}, 2^i - 1))$, discussed in section 3, for each x and for each i, $1 \le i < \log_2 n$.
- To compute $dr[x, y, i]$, we call procedure `exclude`$(\widehat{G}, y, B_{\widehat{T}(y)}(2^{i-1}, 2^i - 1))$, for each y and for each i, $1 \le i < \log_2 n$.

```
    function v-dist$(x, y, v) : \mathcal{R}$
1.      if $d_{xv} + d_{vy} > d_{xy}$ then return $d_{xy}$
2.      $l \leftarrow \lceil \log_2 h_{xv} \rceil$
3.      if $l = \log_2 h_{xv}$ then return $sl[x, y, l+1]$
4.      $r \leftarrow \lceil \log_2 h_{vy} \rceil$
5.      if $r = \log_2 h_{vy}$ then return $sr[x, y, r+1]$
6.      $\hat{u} \leftarrow vr[x, v, l], \hat{v} \leftarrow vl[v, y, r]$
7.      $d \leftarrow \min\{d_{x\hat{u}} + sl[\hat{u}, y, l], sr[x, \hat{v}, r] + d_{\hat{v}y}\}$
8.      if $h_{xv} \le h_{vy}$ then $d \leftarrow \min\{d, dl[x, y, l]\}$        $\{v$ in left half$\}$
9.          else $d \leftarrow \min(d, dr[x, y, r])$                            $\{v$ in right half$\}$
10.     return $d$
```

FIG. 4.1. *Query algorithm for the first oracle.*

- For each x and for each i, $1 \le i < 1 + \log_2 (n-1)$, we compute $sl[x, y, i]$ by calling procedure `fast-exclude`$(G, x, B_{T(x)}(2^{i-1}, 2^{i-1}))$.
- We compute $sr[x, y, i]$ by calling procedure `fast-exclude`$(\widehat{G}, y, B_{\widehat{T}(y)}(2^{i-1}, 2^{i-1}))$ for each y and for each i, $1 \le i < 1 + \log_2 (n-1)$.

4.3. Query. The query algorithm is shown in Figure 4.1. We address only the general interesting case where $v \ne x$ and $v \ne y$; otherwise, the answer is clearly $+\infty$. In line 1 of the algorithm, we get rid of the case where $v \notin \pi_{xy}$ and return d_{xy} as the answer. Lines 2 and 3 take care of the case where v is 2^l edges away from vertex x on π_{xy} for some nonnegative integer l, $0 \le l < \log_2 h_{xy}$. Lines 4 and 5 handle the case where v is 2^r edges away from vertex y on $\widehat{\pi}_{yx}$ for some nonnegative integer r, $0 \le r < \log_2 h_{xy}$. Lines 6 to 9 take care of the remaining cases.

4.4. Analysis. We first discuss the correctness of our query procedure. Using the matrices sl and sr, lines 2 to 5 of the query algorithm answer the following two types of trivial queries: (1) $h_{xv} = 2^l$ for some nonnegative integer l, $0 \le l < \log_2 h_{xy}$, and (2) $h_{vy} = 2^r$ for some nonnegative integer r, $0 \le r < \log_2 h_{xy}$. So in order to prove the correctness of `v-dist`, we need only to prove the correctness of the code segment of lines 6 to 9 that handles the nontrivial case when neither of the above two conditions holds.

In line 7, we assign d to the minimum of $d_{x\hat{u}} + sl[\widehat{u}, y, l]$ and $sr[x, \widehat{v}, r] + d_{\hat{v}y}$, where $d_{x\hat{u}} + sl[\widehat{u}, y, l] = d_{xy}(\widehat{u}, v, v, y)$ and $sr[x, \widehat{v}, r] + d_{\hat{v}y} = d_{xy}(x, v, v, \widehat{v})$. In line 8, we consider the case where $h_{xv} \le h_{vy}$, i.e., v is in the first half of π_{xy} (see Figure 4.2). In this case, $0 < h_{x\hat{u}} < h_{\hat{u}v} = 2^{l-1} \le h_{v\hat{v}}$. The value $dl[x, y, l]$ is the distance from x to y avoiding a subpath having 2^{l-1} vertices and starting at a vertex that is 2^{l-1} edges away from x on π_{xy}. Let the endpoints of that subpath be $\widetilde{u}$ and $\widetilde{v}$, and $h_{x\tilde{u}} < h_{x\tilde{v}}$. So, we have $x < \widehat{u} \le \widetilde{u} < v < \widetilde{v} \le \widehat{v} < y$. Thus, by Claim 3, d covers $[x, v[\times]v, y]$ following the assignment in line 8. By construction of matrices dl, sr, and sr, d is always equal to the weight of some path from x to y that does not use vertex v. Thus, we can conclude that d is the desired answer to `v-dist`(x, y, v). A similar argument holds for the case where $h_{xv} > h_{vy}$ (line 9).

We now address the running times of preprocessing and query procedures.

CLAIM 5. *Preprocessing requires $O(mn^2 + n^3 \log n)$ worst-case time and any* `v-dist` *operation requires $O(1)$ worst-case time.*

Proof. We observe that the number of distinct paths in $B_{T(x)}(2^{i-1}, 2^i - 1)$ is

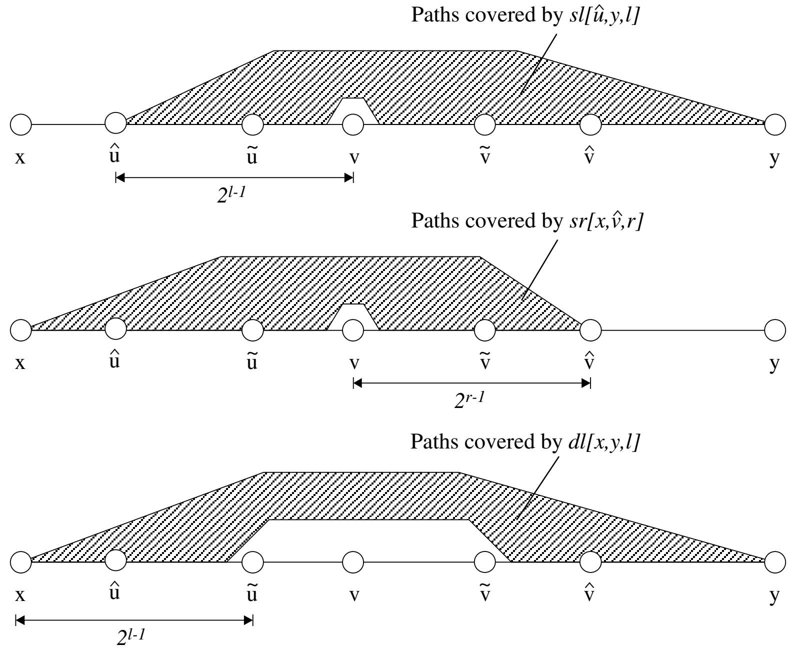

Paths covered by sl[û,y,l]
x
û
ũ
v
ṽ
v̂
y
2^{l-1}
Paths covered by sr[x,v̂,r]
x
û
ũ
v
ṽ
v̂
y
2^{r-1}
Paths covered by dl[x,y,l]
x
û
ũ
v
ṽ
v̂
y
2^{l-1}

FIG. 4.2. *Query example with $h_{xv} \leq h_{vy}$: The distance from vertex x to vertex y in $G - \{v\}$ can be obtained by taking the minimum of $d_{x\hat{u}} + sl[\hat{u}, y, l]$, $sr[x, \hat{v}, r] + d_{\hat{v}y}$, and $dl[x, y, l]$. Notice that the union of the paths in the grey areas in the figure is the set of all possible detours avoiding vertex v.*

exactly the same as the number of vertices on level $2^i - 1$ in $T(x)$. Hence, the total number of distinct paths in all $B_{T(x)}(2^{i-1}, 2^i - 1)$ for $1 \leq i < \log_2 n$ is bounded from above by the number of vertices in $T(x)$. Since `exclude`(G, x, P) runs in $O(|P|(m + n \log n))$ time and $\sum_{1 \leq i < \log_2 n} |B_{T(x)}(2^{i-1}, 2^i - 1)| = O(n)$, the matrices dl and dr can be calculated in $O(mn^2 + n^3 \log n)$ worst-case time. On the other hand, for each x and for each i, $1 \leq i < 1 + \log_2 (n - 1)$, we can compute $sl[x, y, i]$ by calling procedure `fast-exclude`$(G, x, B_{T(x)}(2^{i-1}, 2^{i-1}))$ since the single-vertex paths in $B_{T(x)}(2^{i-1}, 2^{i-1})$ are trivially pairwise independent in $T(x)$. Since `fast-exclude`$(G, x, B_{T(x)}(2^{i-1}, 2^{i-1}))$ runs in $O(m + n \log n)$ time, the total time required to compute the matrix sl is $O(mn \log n + n^2 \log^2 n)$. Similarly the matrix sr can be computed in $O(mn \log n + n^2 \log^2 n)$ time. Hence the preprocessing time is dominated by the time to compute the dl and dr matrices and requires $O(mn^2 + n^3 \log n)$ worst-case time.

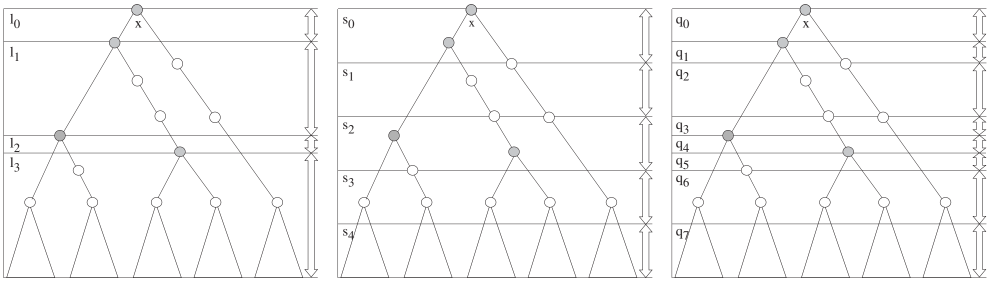
Since the query algorithm executes a constant number of steps, it runs in $O(1)$ worst-case time. □

5. Improving the preprocessing time.

In this section we show that, if one is willing to settle for more space, we can design a distance sensitivity oracle where we reduce the preprocessing time to $O(mn^{1.5} + n^{2.5} \log n)$. This second oracle uses $O(n^{2.5})$ space and answers distance queries in $O(1)$ worst-case time.

5.1. Data structure.

We maintain each d_{xy} and h_{xy}, and we maintain five matrices dh, dt, vc, dc, and bc using $O(n^{2.5})$ space. Matrix dh has size $n \times n \times \lfloor \sqrt{n} \rfloor$, matrices dt, vc, and dc have size $n \times n \times \lfloor 2\sqrt{n} \rfloor$, and matrix bc has size $n \times n$. They



(a) Sequence $\{l_i\}$ obtained by cutting T'(x) at vertices of degree >1

(b) Sequence $\{s_i\}$ obtained by cutting at regular intervals of height $\sqrt{n}$

(c) Sequence $\{q_i\}$ obtained by merging sequences $\{l_i\}$ and $\{s_i\}$

FIG. 5.1. *Constructing matrix dc: Cutting tree $T'(x)$ to form bands of pairwise independent shortest paths in it.*

are defined as follows:

- $dh[x, y, i]$ = distance from vertex x to vertex y in $G - \{v\}$, where v is the vertex of π_{xy} at level i in $T(x)$ and $0 < i \leq \sqrt{n}$;
- $dt[x, y, i]$ = distance from vertex x to vertex y in $G - \{v\}$, where v is the vertex of π_{xy} at level i in $\widehat{T}(y)$ and $0 < i \leq 2\sqrt{n}$;
- $vc[x, y, i]$ = vertex of π_{xy} at level q_i in $T(x)$, where $q_0 = 0 < q_1 < \cdots < q_k < n$ is any increasing sequence of $k + 1 \leq 2\sqrt{n}$ positive numbers depending on x, and $q_i - q_{i-1} \leq \sqrt{n}$ for any i, $1 \leq i \leq k$ (a method for obtaining the sequence $\{q_i : 0 \leq i \leq k\}$ is described in section 5.2.1);
- $dc[x, y, i]$ = distance from vertex x to vertex y in $G - [\hat{u}, \hat{v}]$, where $\hat{u}$ is the successor of $vc[x, y, i]$ in π_{xy} and $\hat{v} = vc[x, y, i+1]$ if $h_{\hat{v}y} > \sqrt{n}$ and is undefined otherwise;
- $bc[x, y]$ = index i such that $q_i + 1 \leq h_{xy} \leq q_{i+1}$.

5.2. Preprocessing. As distances d_{xy} and h_{xy} are easily initialized from shortest path trees of G, we focus on constructing matrices *dh*, *dt*, *dc*, *vc*, and *bc*. Since bands $B_{T(x)}(i, i)$ contain paths formed by single vertices, which are trivially pairwise independent in $T(x)$, constructing matrix *dh* can be done by performing calls to algorithm fast-exclude(G, x, $B_{T(x)}(i, i)$), presented in section 3, for each vertex x and for each i such that $0 < i \leq \sqrt{n}$. Similarly, matrix *dt* can be initialized via calls to algorithm fast-exclude($\widehat{G}$, y, $B_{\widehat{T}(y)}(i, i)$) for each vertex y and for each i such that $0 < i \leq 2\sqrt{n}$.

To compute *dc*, we consider the problem of cutting each shortest path tree $T(x)$ into at most $2\sqrt{n}$ bands of height $\leq \sqrt{n}$, finding a suitable subset of each band containing pairwise independent shortest paths in $T(x)$, and calling fast-exclude to compute distances without those paths. To do so, we need to compute for each vertex x a suitable sequence $\{q_i : 0 \leq i \leq k\}$.

5.2.1. Computing the q_i's. For each vertex x we wish to find a sequence $q_0 = 0 < q_1 < \cdots < q_k < n$ of length $k + 1 \leq 2\sqrt{n}$ such that $q_i - q_{i-1} \leq \sqrt{n}$ for any i, $1 \leq i \leq k$, and compute a subset of paths in $B_{T(x)}(q_i + 1, q_{i+1})$ that are pairwise independent in $T(x)$. The following claim provides a nice combinatorial property on trees that helps us solve the problem.

Let T be a rooted directed tree with n vertices and let $size(v)$ be the number of vertices in the subtree of T rooted at v. Let T' be obtained from T by deleting any vertex u such that $size(u) \leq \sqrt{n}$ in T.

Claim 6. *For any directed tree T with n vertices, at most $\sqrt{n}$ vertices in T' have out-degree > 1.*

Proof. Since T' contains only vertices that have size greater than $\sqrt{n}$ in T, T' has at most $\sqrt{n}$ leaves. This implies that at most $\sqrt{n}$ vertices of T' have out-degree > 1. □

Let $l_0 < l_1 < \cdots < l_k$ be the sequence of levels in T' such that at each level l_i there is a vertex with out-degree > 1 in T' (Figure 5.1(a)). By Claim 6, $k \leq \sqrt{n}$. We now observe that cutting T' at each l_i yields bands of vertex-disjoint paths.

Claim 7. *For any i, $1 \leq i < k$, $B_{T'}(l_i+1, l_{i+1})$ is a band of vertex-disjoint paths.*

Proof. $B_{T'}(l_i + 1, l_{i+1})$ contains all paths that connect vertices at level $l_i + 1$ with vertices at level l_{i+1} in T'. The proof easily follows by observing that, by the definition of sequence $\{l_i\}$, all vertices in T' at levels $l_i + 1$ to $l_{i+1} - 1$ have out-degree ≤ 1. □

Notice that, since by Claim 7 $B_{T'}(l_i + 1, l_{i+1})$ is a band of vertex-disjoint paths and all of them start at the same level $l_i + 1$ in T', then they are clearly pairwise independent in T'. As T' is obtained by pruning T, $B_{T'}(l_i+1, l_{i+1}) \subseteq B_T(l_i+1, l_{i+1})$ and paths in $B_{T'}(l_i + 1, l_{i+1})$ are also pairwise independent in T. Unfortunately, however, we are not guaranteed that $l_i - l_{i-1} \leq \sqrt{n}$, as we need for constructing dc. However, we note that splitting a band of vertex-disjoint paths yields again bands of vertex-disjoint paths. This leads to the following claim.

Claim 8. *If $B_T(i+1, j)$ is a band of vertex-disjoint paths, then for any $i < h < j$, both $B_T(i + 1, h)$ and $B_T(h + 1, j)$ are bands of vertex-disjoint paths.*

Let $s_0 < s_1 < \cdots < s_{\lfloor\sqrt{n}\rfloor}$ be a sequence such that $s_i = i \cdot \lfloor\sqrt{n}\rfloor$ (Figure 5.1b). By Claim 8, if we merge sequences $\{l_i\}$ and $\{s_i\}$ and get rid of duplicates, we obtain an ordered sequence $\{q_i\}$ of length at most $2\sqrt{n}$ with the desired properties (Figure 5.1c).

5.2.2. Computing *vc*, *dc*, and *bc*.

We remark that $\{q_i\}$ induces at most $2\sqrt{n}$ bands of vertex-disjoint paths in $T'(x)$ with height $\leq \sqrt{n}$. Clearly, these paths are pairwise independent in $T'(x)$. To initialize dc, we can thus perform calls to `fast-exclude`$(G, x, B_{T'(x)}(q_i + 1, q_{i+1}))$ for each vertex x and for each $0 < i \leq 2\sqrt{n}$. Again, we can use `fast-exclude` instead of `exclude`.

At this point, one may argue that excluding only independent paths in $T'(x)$, which is obtained by pruning $T(x)$, might not give the correct result for some $dc[x, y, i]$ if $y \notin T'(x)$. However, $dc[x, y, i]$ is defined only when $h_{\hat{v}y} > \sqrt{n}$, where $\hat{v} = vc[x, y, i + 1]$, and $\hat{v} \in T'(x)$ in this case. Thus $dc[x, y, i]$ is correctly computed by calling `fast-exclude`$(G, x, B_{T'(x)}(q_i + 1, q_{i+1}))$.

Finally, we observe that once sequences $\{q_i\}$ have been computed for each $T(x)$, matrices vc and bc can be easily initialized.

5.3. Query.

The query algorithm is shown in Figure 5.2. We first get rid of the cases where $v \notin \pi_{xy}$ and $h_{vy} \leq 2\sqrt{n}$, for which the answers are stored explicitly in d_{xy} and $dt[x, y, h_{vy}]$, respectively (lines 1–2). In line 3 we retrieve the unique index i such that v falls in $B_{T(x)}(q_i + 1, q_{i+1})$, and then in lines 4–5 we identify the vertices $\widehat{u}$ and $\widehat{v}$ on the path π_{xy} in $T(x)$ that are at levels $q_i + 1$ and q_{i+1}, respectively. The correct answer is given in line 6 by accessing matrices dc, dh, and dt.

5.4. Analysis.

To prove the correctness of `v-dist` in the case that lines 3–7 are executed, we first note that $h_{vy} > 2\sqrt{n}$ implies $h_{\hat{v}y} > \sqrt{n}$, since by construction $q_i - q_{i-1} \leq \sqrt{n}$, and thus $dc[x, y, i]$ is well defined. We now prove that the answer takes into account all possible configurations of the endpoints of detours π_{ab}^{uv} (see Figure 2.1). It is easy to see that $x < \widehat{u} < v < \widehat{v} < y$ and

```
function v-dist($x, y, v$) : $\mathcal{R}$
    if $d_{xv} + d_{vy} > d_{xy}$ then return $d_{xy}$
    if $h_{vy} \leq 2\sqrt{n}$ then return $dt[x, y, h_{vy}]$
    $i \leftarrow bc[x, v]$
    $\hat{u} \leftarrow$ successor of $vc[x, y, i]$ in $\pi_{xy}$
    $\hat{v} \leftarrow vc[x, y, i + 1]$
    $d \leftarrow \min\{\ dc[x, y, i],$
                 $d_{x\hat{u}} + dh[\hat{u}, y, h_{\hat{u}v}],$
                 $dt[x, \hat{v}, h_{v\hat{v}}] + d_{\hat{v}y}\ \}$
    return $d$
```

FIG. 5.2. *Query algorithm for the second oracle.*

- $dc[x, y, i] = d_{xy}(x, \widehat{u}, \widehat{v}, y)$,
- $d_{x\hat{u}} + dh[\hat{u}, y, h_{\hat{u}v}] = d_{xy}(\widehat{u}, v, v, y)$, and
- $dt[x, \hat{v}, h_{v\hat{v}}] + d_{\hat{v}y} = d_{xy}(x, v, v, \widehat{v})$.

Thus, by Claim 3, $d = d_{xy}(x, v, v, y)$.

CLAIM 9. *Preprocessing requires $O(mn^{1.5} + n^{2.5} \log n)$ worst-case time and any* `v-dist` *operation requires $O(1)$ worst-case time.*

Proof. Growing shortest path trees $T(x)$ for all vertices x requires $O(mn+n^2 \log n)$ time in the worst case [10]. The proof for the preprocessing follows from Claim 4 by observing that initializing dc, dh, and dt is carried out via $O(\sqrt{n})$ calls to `fast-exclude` for each vertex x. The bound for queries is straightforward. □

6. Handling edge failures. The oracles in the previous two sections can be easily extended to handle edge failures by maintaining one additional matrix de of size $n \times n$ for any x and y:

- $de[x, y]$ = distance from vertex x to vertex y in G without the first edge of π_{xy}.

CLAIM 10. *The matrix de can be initialized in $O(mn + n^2 \log n)$ time.*

Proof. The proof directly follows from the properties of an earlier version of the algorithm `fast-exclude` in [7] based on the notion of *edge-independent paths*. However, since in this article we use the notion of *vertex-independent paths* instead, we present a proof of the claim based on it.

Consider a given $T(x)$, and let $v_1, v_2, \ldots, v_k$ be the children of x in $T(x)$. We extend $T(x)$ to $T'(x)$ and thus G to G' by introducing k new vertices $u_1, u_2, \ldots, u_k$ and for $1 \leq i \leq k$, replacing each edge (x, v_i) in $T(x)$ by two consecutive edges (x, u_i) and (u_i, v_i). Clearly removing any vertex u_i from $T'(x)$ has the same effect as removing the corresponding edge (x, v_i) from $T(x)$. Therefore, since the single-vertex paths in $B_{T'(x)}(1, 1)$ are trivially independent, we can compute $de[x, y]$ for all $y \in V - \{x\}$ in time $O(m + n \log n)$ by calling `fast-exclude`$(G', x, B_{T'(x)}(1, 1))$. (Note that we can perform the same computation in the same time bound without constructing G' explicitly but instead extending `fast-exclude` to handle this special case.) Since we need to call `fast-exclude` once for each $x \in V$, the total initialization time for de is $O(mn + n^2 \log n)$. □

6.1. Query. The query algorithm is shown in Figure 6.1. In line 1 of the algorithm, we get rid of the case where the failed edge is not on π_{xy}. In line 2, d_1 is assigned the distance from x to y avoiding vertex u and thus edge (u, v). In line 3, d_2 is assigned the distance from x to y that avoids the edge (u, v) but passes through the

```
    function e-dist$(x, y, u, v) : \mathcal{R}$
1.      if $d_{xu} + w_{u,v} + d_{vy} > d_{xy}$ then return $d_{xy}$
2.      $d_1 \leftarrow$ v-dist$(x, y, u)$
3.      $d_2 \leftarrow d_{xu} + de[u, y]$
4.      $d \leftarrow \min\{d_1, d_2\}$
5.      return $d$
```

FIG. 6.1. *Query algorithm for link failure.*

vertex u. In line 4, d is assigned the minimum of d_1 and d_2 which is then returned in line 5 as the shortest x to y distance avoiding (u, v).

6.2. Analysis. We observe that the paths in G from x to y that avoid (u, v) can be divided into two groups: (1) paths that avoid u and (2) paths that pass through u. Line 2 of the algorithm finds the length of the shortest path in group (1), and line 3 does the same for group (2). Thus the minimum of the two distances obtained in lines 2 and 3 gives the required distance. Note that since `v-dist` runs in constant worst-case time, so does `e-dist`.

6.3. Avoiding two consecutive edges. In this section we observe that the distance from any vertex x to another vertex y avoiding two *consecutive* failed edges on π_{xy} can be computed in constant time by maintaining another $n \times n$ matrix $\widehat{de}$, which is the dual of de in $\widehat{G}$:

- $\widehat{de}[x, y]$ = distance from vertex y to vertex x in $\widehat{G}$ without the first edge of $\widehat{\pi}_{yx}$.

Assuming that the two consecutive failed links are (u, v) and (v, w), all paths in G from x to y avoiding those two edges can be divided into two groups: (1) paths that avoid v (the length of the shortest such path can be found by calling `v-dist`(x, y, v)) and (2) paths that pass through v (the length of the shortest such path is given by $\widehat{de}[x, v] + de[v, y]$). Thus the smaller of these distances is the required distance.

7. Supporting path queries. The oracles presented in this paper can easily be extended to support path queries of the form `v-path`(x, y, v) and `e-path`(x, y, u, v), which return the first edge on the shortest path from vertex x to vertex y in $G - \{v\}$ and in $G - \{(u, v)\}$, respectively.

In this section, we show how to extend the oracle given in section 4 to support `v-path`(x, y, v) queries. Extending the other oracles and supporting `e-path`(x, y, u, v) queries can easily be done in a similar way. We add to the data structure of section 4.1 the following additional matrices:

- $dle[x, y, i] = (x, x')$, where (x, x') is the first edge on the shortest path from vertex x to vertex y $(\neq x)$ in $G - \pi$, and π is the subpath of π_{xy} starting at level 2^{i-1} and ending at level $2^i - 1$ $(1 \leq i \leq \log_2 h_{xy})$ in $T(x)$;
- $dre[x, y, i] = (x, x')$, where (x', x) is the last edge on the shortest path from vertex y to vertex x $(\neq y)$ in $\widehat{G} - \pi$, and π is the subpath of $\widehat{\pi}_{yx}$ starting at level 2^{i-1} and ending at level $2^i - 1$ $(1 \leq i \leq \log_2 h_{xy})$ in $\widehat{T}(y)$;
- $sle[x, y, i] = (x, x')$, where (x, x') is the first edge on the shortest path from vertex x to vertex y $(\neq x)$ in $G - \{v\}$, and v is the vertex of π_{xy} at level 2^{i-1} $(1 \leq i < 1 + \log_2 h_{xy})$ in $T(x)$;
- $sre[x, y, i] = (x, x')$, where (x', x) is the last edge on the shortest path from

```
function v-path($x, y, v$) : $E$
    if $d_{xv} + d_{vy} > d_{xy}$ then return first edge of $\pi_{xy}$
    $l \leftarrow \lceil \log_2 h_{xv} \rceil$
    if $l = \log_2 h_{xv}$ then return $sle[x, y, l+1]$
    $r \leftarrow \lceil \log_2 h_{vy} \rceil$
    if $r = \log_2 h_{vy}$ then return $sre[x, y, r+1]$
    $\hat{u} \leftarrow vr[x, v, l], \hat{v} \leftarrow vl[v, y, r]$
    $d \leftarrow \min\{d_{x\hat{u}} + sl[\hat{u}, y, l], sr[x, \hat{v}, r] + d_{\hat{v}y}\}$
    if $d = d_{x\hat{u}} + sl[\hat{u}, y, l]$ then $e \leftarrow$ first edge of $\pi_{x\hat{u}}$
    else $e \leftarrow sre[x, \hat{v}, r]$
    if $h_{xv} \leq h_{vy}$ and $dl[x, y, l] < d$ then $e \leftarrow dle[x, y, l]$      {$v$ in left half}
    if $h_{xv} > h_{vy}$ and $dr[x, y, r] < d$ then $e \leftarrow dre[x, y, r]$      {$v$ in right half}
    return $e$
```

FIG. 7.1. *Path version of the query algorithm for our first oracle.*

vertex y to vertex x $(\neq y)$ in $\widehat{G} - \{v\}$, and v is the vertex of $\widehat{\pi}_{yx}$ at level 2^{i-1} $(1 \leq i < 1 + \log_2 h_{xy})$ in $\widehat{T}(y)$.

Matrices *dle*, *dre*, *sle*, and *sre* can easily be initialized in the preprocessing phase within the same time bounds by a simple extension of procedures `exclude` and `fast-exclude` described in section 3. Figure 7.1 shows an implementation of operation `v-path` obtained as a modification of the query procedure `v-dist` given Figure 4.1. The analysis is straightforward and is left to the reader.

8. A space lower bound. In this section, we briefly discuss a space lower bound for the single-source version of the distance sensitivity problem. This version is relevant for shortest path routing in networks [16]: a router cares only about itself as a source when deciding which outgoing link to use when forwarding a packet on a shortest path to its destination. We would therefore like a single-source routing table working under each possible failure. The solution in [24] uses $O(hn)$ space if h is the maximal hop count on a shortest path to a destination. However, we might have $h = \Omega(n)$. Corresponding to our $O(n^2 \log n)$ space solution for the all pairs case, we would like an $O(n \log n)$ space solution for the single-source case. For single-source and single-destination we can get down to $O(n)$ space if the graph is undirected, and this includes a representation of the alternative paths [11, 19], but now we want an $\tilde{O}(n)$ solution working for all possible destinations and failures. Below we show that this is impossible if m is large. In fact, we will prove that the $O(h \cdot n)$ space bound from [24] is tight. More precisely, for any number of vertices $n > 2$, and $h < n$, we will demonstrate a graph with $m = \Theta(hn)$ edges, with maximal hop count h in shortest paths from a specified source, and so that any single-source distance oracle for failures requires $\Omega(hn)$ space no matter whether it is for edge or vertex failures or for directed or undirected graphs.

First we present the construction for $h = n - 1$. We assume that the word length w is even and at least $2(1 + \log n)$. Let the vertices be $v_0, \dots, v_{n-1}$ where v_0 is the source vertex. For each $i < j$, we have an edge (v_i, v_j), and hence a total of $\binom{n}{2}$ edges. Each edge will have an arbitrary half word stored in the least significant bits of its weight. From the answers of a failure distance oracle, we will be able to recover all these half words. It then follows that the full representation of a failure distance oracle requires at least $\binom{n}{2}/2$ words.

The low part of a weight represents numbers below $2^{w/2}$, and the high part represents multiples of $2^{w/2}$. Hence, if the low part is x_0 and the large part is x_1, the weight represented is $x_0 + 2^{w/2}x_1$. If we know the full weight, we can easily recover the low half word with the stored information.

We will now describe how to fill the high parts of the weights. The edges (v_i, v_{i+1}) in the path $(v_0, v_1, \ldots, v_{n-1})$ are all given a high part of 0. The shortest path tree will consist of this path. For an edge (v_i, v_j) with $j \geq i+2$, the high part is $2n-j+i$. It is easy to see that if link (v_i, v_{i+1}) or vertex v_{i+1} fails and we want to go to v_j, $j \geq i+2$, then the unique shortest path first follows the original shortest path from v_0 to v_i and then switches to the link (v_i, v_j). All this is true in both the directed and the undirected cases.

With the above setting of the high parts of the weights, we can first use the regular distances to find the weights along the path; that is, the weight of (v_i, v_{i+1}) is the distance to v_{i+1} minus that to v_i. Next, for each i and $j \geq i+2$, we fail (v_i, v_{i+1}) to get a distance to v_j, which is the known weight of the path $(v_0, \ldots, v_i)$ plus the weight of (v_i, v_j). Thus, if we have a distance oracle that can handle failures, then we can recover all the weights and hence all the low parts with arbitrary stored information.

In the case where $h < n-1$, we start with the source vertex v_0, and then we create $\lceil n-1/h \rceil$ paths from v_0, each of length between $h/2$ and h. Thus, if we removed v_0, the graph would fall into a set of disjoint paths. We now apply the previous construction to each of the paths from v_0. If a path has length h', it uses $\binom{h'}{2} = \Theta(h^2)$ edges. Thus we get a total of $\Theta(nh)$ edges, all of whose weights can be recovered by a failure distance oracle. The representation of the distance oracle therefore requires a space of $\Theta(nh)$ words.

9. Conclusions.

We have presented compact data structures for maintaining information about shortest paths in a weighted directed graph in cases of both vertex failures and edge failures. We have shown that, surprisingly, such a data structure can be stored using nearly the same space required to store a single distance matrix, while still supporting queries in constant time. Our oracle can easily be constructed in $O(mn^2+n^3 \log n)$ time, matching the preprocessing time of all pairs variants of similar problems such as most vital node detection [18] and Vickrey pricing [11] in networks; while these algorithms require $\Theta(n^3)$ space, our oracle requires only $O(n^2 \log n)$ space. Furthermore, we have shown that by using $O(n^{2.5})$ space we can improve construction time to $O(mn^{1.5} + n^{2.5} \log n)$.

Our oracles are different from the case of dynamic algorithms, where distances have to be updated after each vertex or edge failure. Instead, our oracles are already prepared to answer distance queries following the failure of any single vertex or edge, and so the delay time in answering a query is minimized. If failures in a network happen quite rarely, when a node or link goes down we have time to construct a new oracle in the background to cope with a possible additional failure. It would be interesting to explore whether compact oracles with fast query time that are able to deal with more than one failure at a time can be constructed. Finally, can we further improve construction time?

REFERENCES

[1] M. O. Ball, B. L. Golden, and R. V. Vohra, *Finding the most vital arcs in a network*, Oper. Res. Lett., 8 (1989), pp. 73–76.

[2] A. Bar-Noy, S. Khuller, and B. Schieber, *The Complexity of Finding Most Vital Arcs and Nodes*, Technical report CS-TR-3539, Institute for Advanced Studies, University of Maryland, College Park, MD, 1995.

[3] R. A. Chowdhury and V. Ramachandran, *Improved distance oracles for avoiding a link-failure*, in Proceedings of the 13th International Symposium on Algorithms and Computation (ISAAC'02), Vancouver, Canada, Lecture Notes in Comput. Sci. 2518, Springer-Verlag, New York, 2002, pp. 523–534.

[4] H. W. Corley and D. Y. Sha, *Most vital links and nodes in weighted networks*, Oper. Res. Lett., 8 (1982), pp. 157–160.

[5] C. Demetrescu and G. F. Italiano, *Fully dynamic all pairs shortest paths with real edge weights*, in Proceedings of the 42nd IEEE Annual Symposium on Foundations of Computer Science (FOCS'01), Las Vegas, NV, 2001, pp. 260–267.

[6] C. Demetrescu and G. F. Italiano, *A new approach to dynamic all pairs shortest paths*, J. ACM, 51 (2004), pp. 968–992.

[7] C. Demetrescu and M. Thorup, *Oracles for distances avoiding a link-failure*, in Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'02) (San Francisco, CA), SIAM, Philadelphia, ACM, New York, 2002, pp. 838–843.

[8] E. W. Dijkstra, *A note on two problems in connection with graphs*, Numer. Math., 1 (1959), pp. 269–271.

[9] B. Fortz and M. Thorup, *Internet traffic engineering by optimizing OSPF weights*, in Proceedings of the 19th IEEE INFOCOM, The Conference on Computer Communications, Tel-Aviv, Israel, 2000, pp. 519–528.

[10] M. L. Fredman and R. E. Tarjan, *Fibonacci heaps and their use in improved network optimization algorithms*, J. ACM, 34 (1987), pp. 596–615.

[11] J. Hershberger and S. Suri, *Vickrey prices and shortest paths: What is an edge worth?*, in Proceedings of the 42nd IEEE Annual Symposium on Foundations of Computer Science (FOCS'01), Las Vegas, NV, 2001, pp. 252–259. Erratum in FOCS'02.

[12] J. Hershberger, S. Suri, and A. Bhosle, *On the difficulty of some shortest path problems*, in Proceedings of the 20th International Symposium of Theoretical Aspects of Computer Science (STACS'03), Berlin, Germany, 2003, pp. 343–354.

[13] V. King, *Fully dynamic algorithms for maintaining all-pairs shortest paths and transitive closure in digraphs*, in Proceedings of the 40th IEEE Symposium on Foundations of Computer Science (FOCS'99), New York, 1999, pp. 81–99.

[14] V. King and G. Sagert, *A fully dynamic algorithm for maintaining the transitive closure*, J. Comput. System Sci., 65 (2002), pp. 150–167.

[15] V. King and M. Thorup, *A space saving trick for directed dynamic transitive closure and shortest path algorithms*, in Proceedings of the 7th Annual International Computing and Combinatorics Conference (COCOON'01), Guilin, China, Lecture Notes in Comput. Sci. 2108, Springer-Verlag, New York, 2001, pp. 268–277.

[16] J. Moy, *OSPF: Anatomy of an Internet Routing Protocol*, Addison-Wesley, Reading, MA, 1999.

[17] A. Mas-Collel, W. Whinston, and J. Green, *Microeconomic Theory*, Oxford University Press, Oxford, UK, 1999.

[18] E. Nardelli, G. Proietti, and P. Widmayer, *Finding the most vital node of a shortest path*, in Proceedings of the 7th Annual International Computing and Combinatorics Conference (COCOON'01), Guilin, China, Lecture Notes in Comput. Sci. 2108, Springer-Verlag, New York, 2001, pp. 278–287.

[19] N. Nisan and A. Ronen, *Algorithmic mechanism design*, in Proceedings of the 31st Annual ACM Symposium on Theory of Computation (STOC'99), ACM, New York, 1999, pp. 129–140.

[20] S. Pettie, *A new approach to all-pairs shortest paths on real-weighted graphs*, Theoret. Comput. Sci., 312 (2003), pp. 47–74.

[21] L. Roditty and U. Zwick, *Replacement paths and k simple shortest paths in unweighted directed graphs*, in Proceedings of the 32nd International Colloquium on Automata, Languages, and Programming (ICALP'05), Lisboa, Portugal, Lecture Notes in Comput. Sci. 3580, Springer-Verlag, New York, 2005, pp. 249–260.

[22] T. Takaoka, *Subcubic cost algorithms for the all pairs shortest path problem*, Algorithmica, 30 (1998), pp. 309–318.

[23] M. Thorup, *Undirected single source shortest paths with positive integer weights in linear time*, J. ACM, 46 (1999), pp. 362–394.

[24] M. Thorup, *Fortifying OSPF/IS-IS Against Link-Failure*, manuscript, 2001.

[25] M. Thorup, *Fully dynamic all-pairs shortest paths: Faster and allowing negative cycles*, in Proceedings of the 9th Scandinavian Workshop on Algorithm Theory (SWAT'04), Humlebaek, Denmark, 2004, pp. 384–396.

[26] M. Thorup, *Integer priority queues with decrease key in constant time and the single source shortest paths problem*, J. Comput. System Sci., 69 (2004), pp. 330–353.

[27] U. Zwick, *A slightly improved sub-cubic algorithm for the all pairs shortest paths problem with real edge lengths*, in Proceedings of the 15th International Symposium on Algorithms and Computation (ISAAC'04), Hong Kong, China, Lecture Notes in Comput. Sci. 3341, Springer-Verlag, 2004, pp. 921–932.

SIAM J. Comput.
Vol. 37, No. 5, pp. 1319–1341

A GROUP-STRATEGYPROOF COST SHARING MECHANISM FOR THE STEINER FOREST GAME[*]

JOCHEN KÖNEMANN[†], STEFANO LEONARDI[‡], GUIDO SCHÄFER[§], AND STEFAN H. M. VAN ZWAM[¶]

Abstract. We consider a game-theoretical variant of the Steiner forest problem in which each player j, out of a set of k players, strives to connect his terminal pair (s_j, t_j) of vertices in an undirected, edge-weighted graph G. In this paper we show that a natural adaptation of the primal-dual Steiner forest algorithm of Agrawal, Klein, and Ravi [*SIAM J. Comput.*, 24 (1995), pp. 445–456] yields a 2-budget balanced and cross-monotonic cost sharing method for this game. We also present a negative result, arguing that no cross-monotonic cost sharing method can achieve a budget balance factor of less than 2 for the Steiner tree game. This shows that our result is tight. Our algorithm gives rise to a new linear programming relaxation for the Steiner forest problem which we term the *lifted-cut relaxation*. We show that this new relaxation is stronger than the standard undirected cut relaxation for the Steiner forest problem.

Key words. approximation algorithms, mechanism design, primal-dual algorithms, Steiner forests, group-strategyproofness

AMS subject classifications. 68Q25, 68R10, 68W25, 90C59, 91A43

DOI. 10.1137/050646408

1. Introduction.

We consider the problem of devising a cost sharing mechanism that is group-strategyproof and satisfies approximate budget balance for a natural game-theoretical variant of the Steiner forest problem. In its most general form, the game-theoretical setting that we consider in this paper can be described as follows.

We are given a service provider and a set R of potential players (or customers, or agents) that are interested in a service offered by the provider. Each player j in R has a *utility* u_j for receiving this service. We assume that u_j is kept private, i.e., that it is known only to player j. The service provider now solicits bids $\{b_j\}_{j \in R}$ from all players and based on these bids (i) determines a set $Q \subseteq R$ of players that receive the service, (ii) computes a solution to service all players in Q, and (iii) for each $j \in Q$ fixes a *price* x_j that j has to pay for receiving the service. A *cost sharing mechanism* is simply a strategy that the service provider uses to make these decisions. We assume that the mechanism complies with the following three natural assumptions: (i) a player is not charged more than his bid, (ii) a player is charged only if he receives service, and (iii) a player is guaranteed to receive service only if his bid is large enough.

[*] Received by the editors December 1, 2005; accepted for publication (in revised form) August 2, 2007; published electronically January 4, 2008. This work was partially supported by the EU within the 6th Framework Programme under contract 001907 (DELIS). Parts of this work appeared previously in *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, and *Proceedings of the* 32*nd International Colloquium on Automata, Languages and Programming.*
http://www.siam.org/journals/sicomp/37-5/64640.html

[†] Department of Combinatorics and Optimization, University of Waterloo, 200 University Avenue West, Waterloo, ON N2L 3G1, Canada (jochen@uwaterloo.ca).

[‡] Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza," Via Salaria 113, 00198 Roma, Italy (leon@dis.uniroma1.it).

[§] Institut für Mathematik, Technische Universität Berlin, Straße des 17. Juni 136, 10623 Berlin, Germany (schaefer@math.tu-berlin.de).

[¶] Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB, Eindhoven, The Netherlands (svzwam@win.tue.nl).

The total cost that is incurred to establish service for a player set $Q \subseteq R$ is denoted by $c(Q)$. One objective that we wish to achieve is *approximate budget balance.* We say that a cost sharing mechanism is *α-budget balanced* if

$$\frac{1}{\alpha} \cdot c(Q) \leq \sum_{j \in Q} x_j \leq \texttt{opt}_Q. \tag{1.1}$$

The first inequality states that at least a fraction $1/\alpha$ of the total cost $c(Q)$ of servicing the players in Q is recovered by the sum of the prices of the players in Q. The second inequality establishes fairness in that the sum of all prices is not allowed to exceed the optimal cost of servicing the players in Q, denoted $\texttt{opt}_Q$. This second inequality is often referred to as *competitiveness.* A cost sharing mechanism is *budget balanced* if $\alpha = 1$. Ideally we obtain cost sharing mechanisms that compute prices in polynomial time and are budget balanced. However, this is clearly impossible if the underlying problem is NP-hard, and we therefore resort to cost sharing mechanisms that are approximately budget balanced.

Define the *benefit* of a player j to be $u_j - x_j$ if $j \in Q$, and zero otherwise. We assume that each player is selfish and may lie about his utility in order to maximize his benefit. The task is to design a cost sharing mechanism that encourages players to bid their true utility; that is, no player or group of players should be able to benefit from lying about their utilities. A cost sharing mechanism is *strategyproof* if the dominant strategy of each player is to bid his utility; it is said to be *group-strategyproof* if this holds even if players are permitted to collude. More precisely, if, for any choice of $i \in Q' \subseteq Q$, the utility of a player i increases as a result of nontruthful behavior of the players in Q', then there is at least one other player $j \in Q'$ whose utility strictly decreases.

In [14], Moulin and Shenker presented a powerful framework that reduces the task of designing a group-strategyproof cost sharing mechanism for a game to that of giving a *cross-monotonic cost sharing method.* In fact, Immorlica, Mahdian, and Mirrokni [9] showed that all group-strategyproof mechanisms that satisfy a certain technical fairness condition can be obtained using Moulin and Shenker's framework. A *cost sharing method* ξ is an algorithm that, given any subset $Q \subseteq R$ of players, computes a solution to service Q and for each $j \in Q$ determines a nonnegative cost share $\xi_Q(j)$. Analogously to the definition in (1.1), we say that ξ is α-budget balanced if

$$\frac{1}{\alpha} \cdot c(Q) \leq \sum_{j \in Q} \xi_Q(j) \leq \texttt{opt}_Q.$$

A cost sharing method ξ is *cross-monotonic* if, for any two sets Q and S such that $Q \subseteq S$ and any player $j \in Q$, we have

$$\xi_S(j) \leq \xi_Q(j).$$

In other words, the cost share of any player under the given cost sharing method does not increase if the size of the player set increases.

Given a budget balanced and cross-monotonic cost sharing method ξ for a game, the following cost sharing mechanism from [14] satisfies budget balance and group-strategyproofness: Initially, let $Q = R$. If, for every player $j \in Q$, the cost share $\xi_Q(j)$ is less than or equal to his bid b_j, stop. Otherwise, remove from Q all players whose cost shares in Q are larger than their bids, and repeat. Eventually, let $x_j = \xi_Q(j)$

be the prices that are charged to players in the final set Q. Jain and Vazirani [10] later proved that the result of Moulin and Shenker also holds true if one considers cross-monotonic cost sharing methods that are approximately budget balanced.

The underlying optimization problem that we consider in this context is the *Steiner forest problem.* In this problem, we are given an undirected graph $G = (V, E)$, a nonnegative cost function $c : E \rightarrow \mathbb{R}^+$ on the edges of G, and a set of $k > 0$ terminal pairs $R = \{(s_1, t_1), \ldots, (s_k, t_k)\} \subseteq V \times V$. Each terminal pair (s_j, t_j), $1 \leq j \leq k$, is associated with a player j that wants to establish a connection between vertices s_j and t_j. A feasible solution for terminal set R is a forest $F \subseteq E$ such that vertices s_j and t_j are in the same tree of F for all $1 \leq j \leq k$. The objective is to find a feasible solution of smallest total cost.

The *Steiner tree* problem is a special case of the Steiner forest problem in which there is a root vertex $r \in V$ and $r \in \{s, t\}$ for all terminal pairs $(s, t) \in R$. In other words, the problem consists of a set of terminals $R \subseteq V$, and a root vertex $r \in V$ and the goal is to connect the terminals in R to r in the cheapest possible way.

Previous work. Computing minimum-cost Steiner trees and forests is NP-hard [7] and APX-complete [4, 5], and therefore, neither of the two problems admits a polynomial time approximation scheme unless P = NP. The best known algorithm for the Steiner forest problem, due to Agrawal, Klein, and Ravi [2] and generalized by Goemans and Williamson [8], uses the primal-dual schema. The algorithms in [2, 8] achieve an approximation ratio of $(2 - 1/k)$ and are both based on the classical *undirected cut formulation* for the Steiner forest problem [3]. The integrality gap of this relaxation is known to be $(2 - 1/k)$, and the results in [2, 8] are therefore tight.

Despite the recent interest in computational game theory, examples of combinatorial optimization problems that possess cross-monotonic cost sharing methods are scarce: Moulin and Shenker [14] gave a cross-monotonic cost sharing method for problems whose optimal cost function is a submodular function of the set Q. However, this condition does not hold for many important network design problems such as Steiner trees and facility location.

The first cross-monotonic cost sharing method for the minimum-cost spanning tree game is due to Kent and Skorin-Kapov [11]. Jain and Vazirani [10] presented an alternative method that is based on the primal-dual spanning tree algorithm due to Edmonds [6]; the authors then used this result to obtain a 2-budget balanced, cross-monotonic cost sharing method for the Steiner tree game. Pál and Tardos [15] developed a 3-budget balanced cross-monotonic cost sharing method for the facility location problem and a 15-budget balanced cross-monotonic cost sharing method for the single-source rent-or-buy network design problem.

In most of the methods proposed so far to obtain approximate budget balanced cross-monotonic cost sharing methods, the cost shares are closely related to a feasible dual solution generated by the algorithm, and therefore approximate budget balance is an immediate consequence of the approximation guarantee achieved by the algorithm.

Immorlica, Mahdian, and Mirrokni [9] showed that NP-hardness of the underlying combinatorial optimization problem is not the only obstruction in achieving budget balance. The authors provide lower bounds on the budget balance factor α of cross-monotonic cost sharing methods for several problems. Among other results they prove (maybe most surprisingly) lower bounds of $\Omega(n)$ and $\Omega(n^{1/3})$ for the budget balance factor of the set cover and the vertex cover problems, respectively. The authors left open the issue of finding a lower bound on the budget balance factor for the Steiner tree problem.

Edmonds [6] proposed the *bidirected cut relaxation* for the Steiner tree problem. It is a well-known fact that an integrality gap of α for this formulation implies that no $(\alpha - \epsilon)$-budget balanced cross-monotonic cost sharing method can exist for any $\epsilon > 0$ (see also [9]). The worst example known for the integrality gap of the bidirected-cut relaxation is due to Goemans (cf. [1]) and shows a gap of 8/7.

Our contribution. While the performance guarantee of two of primal-dual approximation algorithms for the Steiner tree problem is matched by a 2-budget balanced cross-monotonic cost sharing method [10], a similar result for the Steiner forest problem was elusive so far. This contrasts the optimization version of the problem, where primal-dual $(2 - 1/k)$-approximation algorithms exist for both problems [2, 8].

In this paper we present a cross-monotonic cost sharing method for the Steiner forest problem that is 2-budget balanced. Our algorithm is a natural adaptation of the primal-dual algorithm for computing Steiner forests due to Agrawal, Klein, and Ravi [2], which we will review in section 2. We then show how a modification of this algorithm turns it into a cross-monotonic cost sharing method in sections 3 and 4.

An interesting byproduct of the work in this paper is that our Steiner forest algorithm is $(2 - 1/k)$-approximate despite the fact that the forest computed by our method is usually costlier than those computed by known primal-dual algorithms in [2, 8]. Although we are able to prove that the cost shares computed by our algorithm are 2-budget balanced, they do not correspond to a feasible dual solution for any of the known linear programming (LP) formulations of the Steiner forest problem. Obvious questions that arise are: Is there an alternate Steiner forest LP formulation such that the cost shares computed by our algorithm correspond to a feasible dual solution? If so, how does this new LP relaxation relate to the standard undirected-cut LP relaxation?

We answer these questions by presenting in section 5 a new LP relaxation for the Steiner forest problem, the *lifted-cut relaxation*. The dual solution computed by our algorithm is feasible for the dual linear program of the lifted-cut relaxation. We prove that our new relaxation is stronger than the well-studied undirected-cut relaxation for the Steiner forest problem. There are instances for which the optimal objective function value of our lifted-cut relaxation provides a much better approximation of the optimal cost of a Steiner forest than the undirected cut-relaxation.

Unfortunately, there exist instances that show that the LP/integer programming (IP) gap of the lifted-cut relaxation is still nearly 2. For instances in which the lifted-cut relaxation is stronger than the undirected-cut relaxation we can, however, show that the additional strength can be used to improve the performance guarantee of the existing primal-dual Steiner forest algorithms in [2, 8]. The details are presented in section 6.

A natural question is whether there is a cross-monotonic cost sharing method for the Steiner tree and forest games that achieves a budget balance factor smaller than 2. We provide a negative answer to this question by showing (cf. section 7) that there is no $(2 - \epsilon)$-budget balanced cross-monotonic cost sharing method for Steiner trees for any $\epsilon > 0$. This proves that the cross-monotonic cost sharing method for the Steiner tree game [10, 11] as well as our cross-monotonic cost sharing method for the Steiner forest game are tight with respect to the budget balance factor. We remark that our lower bound holds for *any* cross-monotonic cost sharing method for the Steiner tree game, including those taking exponential time.

2. A primal-dual algorithm for the Steiner forest problem. We review the algorithm of Agrawal, Klein, and Ravi [2]. Subsequently, we use AKR to refer to this algorithm. AKR is a *primal-dual* algorithm; that is, the algorithm constructs both a feasible and integral primal solution and a feasible dual solution for an LP formulation of the Steiner forest.

A standard IP formulation for the Steiner forest problem has a binary variable x_e for all edges $e \in E$: x_e has value 1 if edge e is part of the resulting forest and 0 otherwise. A subset $U \subseteq V$ is a *Steiner cut* if it separates at least one terminal pair in R. In other words, U is a Steiner cut iff there is a pair $(s,t) \in R$ with $|\{s,t\} \cap U| = 1$. We use $\mathcal{S}$ to refer to the set of all Steiner cuts. For a subset $U \subseteq V$ we define $\delta(U)$ to be the set of all edges that have exactly one endpoint in U. Consider a Steiner cut $U \in \mathcal{S}$. Any feasible solution F for a given Steiner forest instance must *cross* this cut at least once, i.e., $|\delta(U) \cap F| \geq 1$. This gives rise to the following IP formulation for the Steiner forest problem, which we refer to as the *undirected-cut formulation*:

$$\text{(IP)} \qquad \mathtt{opt}_{\mathrm{IP}} = \min \quad \sum_{e \in E} c(e) \cdot x_e$$

$$\text{(2.1)} \qquad \text{s.t.} \quad \sum_{e \in \delta(U)} x_e \geq 1 \qquad \forall U \in \mathcal{S},$$

$$x_e \in \{0,1\} \qquad \forall e \in E.$$

The dual of the LP relaxation of (IP) has a variable y_U for each Steiner cut $U \in \mathcal{S}$. There is a constraint for each edge $e \in E$ that limits the total dual assigned to sets $U \in \mathcal{S}$ that contain exactly one endpoint of e to be at most the cost $c(e)$ of the edge.

$$\text{(D)} \qquad \mathtt{opt}_{\mathrm{D}} = \max \quad \sum_{U \in \mathcal{S}} y_U$$

$$\text{(2.2)} \qquad \text{s.t.} \quad \sum_{U \in \mathcal{S}\colon e \in \delta(U)} y_U \leq c(e) \qquad \forall e \in E,$$

$$y_U \geq 0 \qquad \forall U \in \mathcal{S}.$$

Algorithm AKR constructs a primal feasible solution for (IP) and a dual feasible solution for (D). The algorithm starts with an infeasible primal solution and reduces the degree of infeasibility as it progresses. At the same time, it greedily creates a feasible dual packing of subsets of large total value. The algorithm raises dual variables of a laminar family of vertex subsets. The final dual solution is maximal in the sense that no single set can be raised without violating a constraint of type (2.2).

We can think of an execution of AKR as a process over time. Let x^τ and y^τ, respectively, be the primal incidence vector and feasible dual solution at time τ. We use F^τ to denote the forest corresponding to x^τ. Initially, $x_e^0 = 0$ for all $e \in E$ and $y_U^0 = 0$ for all $U \in \mathcal{S}$. In the following, we say that an edge $e \in E$ is *tight* if the corresponding constraint (2.2) holds with equality. Assume that the forest F^τ at time τ is infeasible. We use $\bar{F}^\tau$ to denote the subgraph of G that is induced by the tight edges for dual y^τ. In the following, we will also use the term *moat* to refer to a connected component U of $\bar{F}^\tau$. A connected component U of $\bar{F}^\tau$ is *active* at time τ iff it separates at least one terminal pair, i.e., iff $U \in \mathcal{S}$. Let $\mathcal{A}^\tau$ be the set of all connected components of $\bar{F}^\tau$ that are active at time τ.

AKR raises the dual variables for all sets in $\mathcal{A}^\tau$ uniformly at all times $\tau \geq 0$. We say that moats U_1 and U_2 *collide* at time τ if

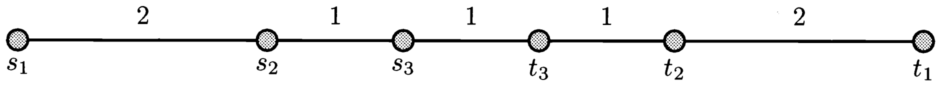


FIG. 3.1. *Distributing the dual growth of each moat U in* AKR *uniformly among U's active terminals does not lead to a cross-monotonic cost sharing method.*

1. U_1 and U_2 are moats at some time $\tau' < \tau$, and
2. τ is the first time during the execution of the algorithm at which forest $\bar{F}^\tau$ contains a connected component containing the vertices of both U_1 and U_2.

If this happens, we add the edges on a shortest U_1, U_2-path to F^τ and continue.

The following is the main result of [2].

THEOREM 2.1. *Suppose that algorithm* AKR *outputs a forest F and a feasible dual solution $\{y_U\}_{U \in \mathcal{S}}$. Then*

$$c(F) \leq \left(2 - \frac{1}{k}\right) \cdot \sum_{U \in \mathcal{S}} y_U \leq \left(2 - \frac{1}{k}\right) \cdot \mathtt{opt}_R,$$

where $\mathtt{opt}_R$ is the minimum cost of a Steiner forest for the given input instance with terminal set R.

3. A cross-monotonic algorithm for the Steiner forest game. In this section we use the ideas presented in the last section to develop a cross-monotonic cost sharing method for the Steiner forest problem.

Consider a subset Q of players and let R be the corresponding set of terminal pairs. Running AKR on this instance yields a feasible dual solution for (D) and Theorem 2.1 implies that its value is at least $\frac{1}{2}\mathtt{opt}_R$ and at most $\mathtt{opt}_R$.

Can we distribute the dual computed by AKR as cost shares over the players in Q? A natural strategy goes as follows: at any time τ during the run of the algorithm, and for any active moat $U \in \mathcal{A}^\tau$, distribute the increase in y_U evenly among the players in Q whose terminals are separated by U.

This strategy does not lead to a cross-monotonic cost sharing method as the example instance in Figure 3.1 shows. In the figure, the edges are labeled by their costs. The instance shown has three players and the terminal pair of player $i \in Q = \{1, 2, 3\}$ is (s_i, t_i). Distributing the dual growth uniformly as proposed yields a cost share of $\xi_Q(1) = 3$ for player 1. On the other hand, if player 3 leaves the game and the set of remaining players is $Q' = \{1, 2\}$, we have $\xi_{Q'}(1) = 5/2$, violating cross-monotonicity.

The example above shows that the *activity time* of a terminal in AKR depends on the presence of other terminals. We now present an adaptation of AKR (subsequently referred to as KLS) that overcomes this problem.

Define the *time of death* $\mathtt{d}(s,t)$ for each terminal pair $(s,t) \in R$ as

$$\mathtt{d}(s,t) = \frac{1}{2} \cdot c(s,t), \tag{3.1}$$

where $c(s,t)$ denotes the cost of the minimum-cost s,t-path in G. We assume for ease of presentation that each vertex $v \in V$ has at most one terminal on it. This assumption is without loss of generality since we can replace each vertex in V by a sufficient number of copies and link these copies by zero-cost edges. We extend the death time notion to individual terminals and define $\mathtt{d}(s) = \mathtt{d}(t) = \mathtt{d}(s,t)$ for terminals $s, t \in R$.[1]

Recall that AKR raises the dual variables for all sets in $\mathcal{A}^\tau$. As a consequence, y^τ is a feasible dual solution for (D) at all times $\tau \geq 0$. Using the notation introduced in the previous section, we obtain KLS by modifying the definition of $\mathcal{A}^\tau$. We say that a connected component U of $\bar{F}^\tau$ is *active* at time τ iff it contains at least one terminal $v \in U$ with death time at least τ; i.e., U is active at time τ iff there exists $v \in U$ with $\mathtt{d}(v) \geq \tau$. KLS grows all active connected components in $\mathcal{A}^\tau$ uniformly at all times $\tau \geq 0$. Observe that in this way KLS also raises dual variables of connected components in $\mathcal{A}^\tau$ that do *not* correspond to Steiner cuts. In what follows we denote by $\mathcal{N}$ the set of all *non-Steiner cuts*, i.e.,

$$\mathcal{N} = \{U \subseteq V \,:\, U \notin \mathcal{S},\; U \cap R \neq \emptyset\}.$$

Furthermore, we let $\mathcal{U} = \mathcal{S} \cup \mathcal{N}$ be the set of all Steiner and non-Steiner cuts.

What is the intuition behind this? Consider a terminal pair $(s,t) \in R$ and imagine running the primal-dual Steiner forest algorithm AKR on the instance consisting of this terminal pair only. In this case, AKR grows two moats corresponding to s and t, respectively, at all times $\tau \leq \mathtt{d}(s,t)$. At time $\mathtt{d}(s,t)$ the moats of s and t collide and a shortest path connecting the terminals is added. In KLS a terminal pair (s,t) is active for the time it would take s and t to connect in the absence of any other terminals. Therefore, the activity time of s and t is *independent* of other terminal pairs. This independence is the crucial property leading to cross-monotonicity.

Consider an arbitrary terminal pair $(s,t) \in R$. Observe that our choice of the death time $\mathtt{d}(s,t)$ in (3.1) implies that s and t end up in the same connected component of the final forest F. Therefore, KLS constructs a feasible solution for the given Steiner forest instance.

For a terminal $v \in R$ and for $\tau \leq \mathtt{d}(v)$ we let $U^\tau(v)$ be the connected component in $\bar{F}^\tau$ that contains v. Also let $a^\tau(v)$ be the number of terminals in $U^\tau(v)$ whose death time is at least τ. We then define the cost share of terminal vertex $v \in R$ as

$$\xi_R(v) = \int_{\tau=0}^{\mathtt{d}(v)} \frac{1}{a^\tau(v)}\, d\tau, \tag{3.2}$$

and we let $\xi_R(s,t) = \xi_R(s) + \xi_R(t)$ for all $(s,t) \in R$.

The proof of the following theorem is the subject of section 4.

THEOREM 3.1. *The cost shares ξ computed by* KLS *are cross-monotonic and 2-budget balanced.*

4. Analysis.

We denote the final forest produced by KLS(R) by F and use $\{y_U\}_{U \in \mathcal{U}}$ for the dual computed by our method.

4.1. Proving cross-monotonicity.

In order to prove the cross-monotonicity of KLS we consider an arbitrary terminal pair $(s,t) \in R$ and let $R_0 = R \setminus \{(s,t)\}$. In this section we study the effect of the removal of (s,t) on the cost shares of all other terminal pairs $(s',t') \in R_0$.

Let us first introduce some simplifying notation. Assume that KLS(R) terminates at time τ^* with forest F. Similarly, KLS(R_0) finishes at time τ_0^* with a forest F_0.

[1] Throughout this paper we slightly abuse notation by letting R refer to both the set of terminal pairs and the set of terminal vertices.

Moreover, for all times τ we let $\mathcal{C}^\tau$ and $\mathcal{C}_0^\tau$ be the sets of connected components of $\bar{F}^\tau$ and of $\bar{F}_0^\tau$, respectively. The next lemma shows that $\mathcal{C}_0^\tau$ is a *refinement* of $\mathcal{C}^\tau$.

LEMMA 4.1. *For all times $\tau \le \tau^*$ and for all $U_0 \in \mathcal{C}_0^\tau$ there must be a set $U \in \mathcal{C}^\tau$ such that $U_0 \subseteq U$.*

Proof. The proof is by induction on the time τ. It is clear that the claim is true for $\tau = 0$ since $\mathcal{C}^0 = \mathcal{C}_0^0 = V$. Consider a point in time $0 \le \tau < \tau^*$, and assume that the claim is true at time τ. $\mathtt{KLS}(R_0)$ grows active sets in $\mathcal{C}_0^\tau$, and these are the only sets that can potentially violate the claim at any time $\tau + \epsilon$ for $\epsilon > 0$. Let $U_0 \in \mathcal{C}_0^\tau$ be an active set at time τ in $\mathtt{KLS}(R_0)$; i.e., there exists a terminal $v \in U_0$ with $\mathtt{d}(v) \ge \tau$. From the induction hypothesis we know that there is a connected component U of $\mathcal{C}^\tau$ that contains U_0. Then U must be active in $\mathtt{KLS}(R)$ at time τ, and hence $\mathtt{KLS}(R)$ grows U at time τ. The claim follows. □

Lemma 4.1 immediately implies cross-monotonicity. Let $\xi(v)$ and $\xi_0(v)$ be the cost share of terminal $v \in R_0$ in $\mathtt{KLS}(R)$ and in $\mathtt{KLS}(R_0)$, respectively.

COROLLARY 4.2. *Algorithm* KLS *is cross-monotonic, i.e., for each $v \in R_0$ we have*

$$\xi_0(v) \ge \xi(v).$$

Proof. Let $U^\tau(v)$ and $U_0^\tau(v)$ be the moats containing terminal v at time τ in $\mathtt{KLS}(R)$ and $\mathtt{KLS}(R_0)$, respectively. Similarly, let $a^\tau(v)$ and $a_0^\tau(v)$ be the number of terminals with death time at least τ in $U^\tau(v)$ and $U_0^\tau(v)$. Lemma 4.1 implies that $U_0^\tau(v) \subseteq U^\tau(v)$ and hence $a_0^\tau(v) \le a^\tau(v)$ for all $\tau \le \tau^*$ and for all $v \in R_0$. Hence we obtain

$$\xi(v) = \int_{\tau=0}^{\mathtt{d}(v)} \frac{1}{a^\tau(v)}\, d\tau \le \int_{\tau=0}^{\mathtt{d}(v)} \frac{1}{a_0^\tau(v)}\, d\tau = \xi_0(v)$$

for all $v \in R_0$ and the corollary follows. □

4.2. Proving approximate budget balance. We first prove that the cost shares computed by KLS satisfy approximate cost recovery.

LEMMA 4.3. *Suppose that algorithm* KLS *outputs a forest F and a dual solution $\{y_U\}_{U \in \mathcal{U}}$. We then have*

$$c(F) \le 2 \cdot \sum_{U \in \mathcal{U}} y_U = 2 \cdot \sum_{(s,t) \in R} \xi_R(s,t). \tag{4.1}$$

Proof. Using Definition 3.2 it can then be seen that the cost share sum on the right-hand side of (4.1) increases by ϵ whenever the total dual value increases by ϵ for some $\epsilon > 0$. Hence we must have $\sum_{(s,t)\in R} \xi_R(s,t) = \sum_{U\in\mathcal{U}} y_U$.

We next prove that $c(F) \le 2 \cdot \sum_{U\in\mathcal{U}} y_U$. We construct a new instance of the Steiner forest problem as follows. For each terminal $v \in R$, introduce a new terminal pair $(\tilde{v}, \tilde{v}')$ and edges $(v, \tilde{v})$ with $c(v, \tilde{v}) = 0$ and $(\tilde{v}, \tilde{v}')$ with $c(\tilde{v}, \tilde{v}') = 2\mathtt{d}(v)$. Run the algorithm AKR on the set of terminal pairs $\tilde{R} \cup R$, where $\tilde{R} = \{(\tilde{v}, \tilde{v}') : v \in R\}$. We denote by $\tilde{\mathcal{S}}$ the set of all Steiner cuts in this new problem, and we use $\tilde{E}$ for the set of only the new edges. Since the edge $(v, \tilde{v})$ will go tight at time $\tau = 0$, the component containing v will be active for precisely the same amount of time as in the run of KLS, so we can convert the dual constructed by AKR on the new problem to the dual constructed by KLS, and vice versa. Let $\{y_U^{\mathtt{AKR}}\}_{U\in\tilde{\mathcal{S}}}$ and $\{y_U^{\mathtt{KLS}}\}_{U\in\mathcal{U}}$ be the dual solutions computed by AKR and KLS, respectively. Since the new edges do not become tight before the death time of a vertex v, the solution computed by AKR, when restricted to the original graph, must be equal to the solution computed by KLS. By Theorem 2.1 the solution returned for this new problem is within a factor 2 of the optimal solution for this problem. Using this, we see

$$\begin{aligned}\sum_{e\in E\cup\tilde{E}} c(e)x_e &= \sum_{e\in E} c(e)x_e + \sum_{e\in\tilde{E}} c(e)x_e\\ &\le 2\sum_{U\in\tilde{\mathcal{S}}} y_U^{\mathtt{AKR}} = 2\sum_{U\in\mathcal{S}\cup\mathcal{N}} y_U^{\mathtt{KLS}} + 2\sum_{v\in R} y_{\{\tilde{v}'\}}^{\mathtt{AKR}}.\end{aligned}$$

Furthermore, we know that edge $(\tilde{v},\tilde{v}')$ is added exactly at time $c(\tilde{v},\tilde{v}')/2$. Hence

$$\sum_{e\in\tilde{E}} c(e)x_e = 2\sum_{v\in R} y_{\{\tilde{v}'\}}^{\mathtt{AKR}}.$$

The lemma follows immediately since $c(F)=\sum_{e\in E} c(e)x_e$. □

We remark that Lemma 4.3 does *not* imply that the cost $c(F)$ of forest F produced by our cost sharing method is at most twice that of an optimal Steiner forest. In fact, $\{y_U\}_{U\in\mathcal{U}}$ is not a feasible solution for (D) since our algorithm raises duals for active sets that correspond to non-Steiner cuts $U\in\mathcal{N}$.[2] Surprisingly, however, we can show that the total dual $\sum_{U\in\mathcal{U}} y_U$ is bounded by the cost $\mathtt{opt}_R$ of an optimal Steiner forest for the given instance on terminal set R.

LEMMA 4.4. *Let $\{y_U\}_{U\in\mathcal{U}}$ be the dual computed by* KLS(R)*, and let $\mathtt{opt}_R$ be the minimum cost of any feasible Steiner forest for the given instance. We have*

$$\sum_{U\in\mathcal{U}} y_U \le \mathtt{opt}_R.$$

Lemmas 4.3 and 4.4 imply the following corollary on the approximate budget balance of KLS.

COROLLARY 4.5. *Let F be the Steiner forest computed by* KLS(R). *We then have*

$$\frac{1}{2}\cdot c(F) \le \sum_{(s,t)\in R} \xi_R(s,t) \le \mathtt{opt}_R.$$

It remains to prove Lemma 4.4.

4.3. A proof of Lemma 4.4. Recall the definition of the death time $\mathtt{d}(s,t)$ of a terminal pair $(s,t)\in R$. In the following, let

$$R=\{(s_1,t_1),\cdots,(s_k,t_k)\}$$

such that

$$\mathtt{d}(s_1,t_1)\le\cdots\le\mathtt{d}(s_k,t_k).$$

We define a precedence order $\prec$ on R by letting $(s_i,t_i)\prec(s_j,t_j)$ iff $i\le j$. We extend this order to terminal vertices by letting

$$s_1\prec t_1\prec s_2\prec t_2\prec\cdots\prec s_k\prec t_k.$$

[2] Observe, however, that the projection of y on the set of Steiner cuts is feasible for (D).

For ease of notation we assume that $v \prec v$ for all $v \in R$.

Let U^τ be an active connected component in $\mathtt{KLS}(R)$ at some time $\tau \geq 0$. A terminal vertex $v \in U^\tau$ is *responsible* for the growth of U^τ iff there does not exist a terminal $u \in U^\tau$ different from v with $v \prec u$. This way, each active moat in KLS has a unique responsible terminal vertex. For a terminal vertex $v \in R$ and a time $\tau \geq 0$, let $r^\tau(v) = 1$ if v is responsible at time τ and $r^\tau(v) = 0$ otherwise. We then define the *responsibility time* of a terminal $v \in R$ as

$$r(v) = \int_{\tau=0}^{\mathtt{d}(v)} r^\tau(v)\, d\tau. \tag{4.2}$$

As before, we let $U^\tau(v)$ be the connected component of $\bar{F}^\tau$ containing terminal $v \in R$. We can show that a terminal $v \in R$ is responsible for a unique moat at all times $0 \leq \tau \leq r(v)$.

Claim 4.6. *Let $v \in R$ be a terminal, and let $r(v)$ be its responsibility time. Then, v is responsible for $U^\tau(v)$ in $\mathtt{KLS}(R)$ for all $0 \leq \tau < r(v)$.*

Proof. Assume for the sake of contradiction that there is a point of time $\tau \in [0, r(v))$ such that v is not responsible for $U = U^\tau(v)$. Since U is active, we know that there must be a terminal $u \in U$ that is responsible. We therefore must have $v \prec u$ and also $\mathtt{d}(v) \leq \mathtt{d}(u)$. Since u and v are contained in the same active moat in KLS at time τ, this means that v cannot be responsible after time τ, and hence $r(v) < \tau$, which is a contradiction. □

Definition (4.2) also implies that

$$\sum_{U \in \mathcal{U}} y_U = \sum_{v \in R} r(v), \tag{4.3}$$

and hence it suffices to bound the sum on the right-hand side in order to prove Lemma 4.4.

Let F^* be a minimum-cost Steiner forest for the given instance with terminal set R. Consider a tree T in F^* and suppose that T connects the terminals $R(T) = \{v_1, \ldots, v_p\}$. We let $R^\tau(T)$ be the set of terminal vertices in $R(T)$ that are responsible at time τ, i.e.,

$$R^\tau(T) = \{v \in R(T) \,:\, r^\tau(v) = 1\}.$$

The following claim shows that at any time τ the moats in

$$U^\tau(T) = \{U^\tau(v) \,:\, v \in R^\tau(T)\}$$

are pairwise disjoint.

Claim 4.7. *Consider a point of time τ and two terminal vertices $u, v \in R^\tau(T)$, $u \neq v$. The two moats $U^\tau(u)$ and $U^\tau(v)$ must be disjoint.*

Proof. Assume for the sake of contradiction that $U^\tau(u)$ and $U^\tau(v)$ are not disjoint. Since both $U^\tau(u)$ and $U^\tau(v)$ are connected components of $\bar{F}^\tau$ it must therefore be the case that $U^\tau(u) = U^\tau(v)$. Claim 4.6 implies that both u and v are responsible for this moat, and hence, we must have $u = v$. This contradicts our choice of u and v. □

The example in Figure 4.1 shows three terminal pairs $(s_1, t_1), (s_2, t_2)$, and (s_3, t_3) that are connected by a tree T in an optimal solution F^*. The figure shows a snapshot of algorithm KLS at some time $\tau > 0$. At this time, five of the terminals are responsible:

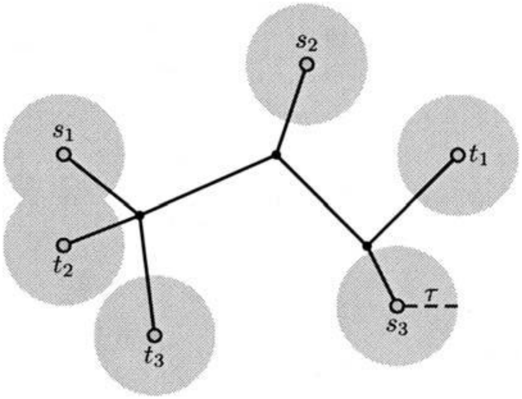


FIG. 4.1. *Snapshot of algorithm* KLS *at some time* $\tau > 0$.

$R^\tau(T) = \{s_1, s_2, s_3, t_1, t_3\}$ (assuming that $t_2 \prec s_1$). Consequently, Claim 4.7 implies that the five moats $U^\tau(s_1), U^\tau(s_2), U^\tau(s_3), U^\tau(t_1)$, and $U^\tau(t_3)$ are pairwise disjoint. But this means that each of the moats has a nonempty intersection with T and therefore, we can *charge* their dual growth in the algorithm to the cost $c(T)$ of tree T.

Let $w \in R(T)$ be the terminal vertex with highest responsibility time among all terminals spanned by tree T. Then, for all terminals $v_i \in R(T) \setminus \{w\}$ and for all $0 \le \tau \le r(v_i)$, Claim 4.7 implies that the moats $U^\tau(w)$ and $U^\tau(v_i)$ are disjoint. Therefore,

$$\sum_{v_i \in R(T)\setminus\{w\}} r(v_i) \le c(T).$$

On the other hand, $r(w)$ must be at most $\mathtt{d}(w)$ which in turn is at most $c(T)/2$, and hence, the last inequality implies that

$$\sum_{i=1}^{p} r(v_i) \le \frac{3}{2} c(T).$$

In the remainder of this section, we will strengthen the above argument in order to prove Lemma 4.4.

LEMMA 4.8. *If* $\delta(U^\tau(w)) \cap T \neq \emptyset$ *for all* $0 \le \tau < r(w)$, *then we must have*

$$\sum_{v \in R(T)} r(v) \le c(T).$$

Proof. Consider any point of time $\tau \ge 0$ where there are at least two terminals in $R(T)$ that are responsible, i.e., $|R^\tau(T)| > 1$. By Claim 4.7 we have that the moats in $U^\tau(T)$ are pairwise disjoint. On the other hand, the vertices in $R^\tau(T)$ are connected by T, and hence, each of the moats in $U^\tau(T)$ loads a distinct part of the edges of T; see Figure 4.1.

Consider now a time τ where $|R^\tau(T)| = 1$. It must be the case that w is the only remaining responsible terminal among the vertices in $R(T)$, i.e., $R^\tau(T) = \{w\}$. By assumption, $U^\tau(w)$ loads at least one edge of T. This concludes the proof of the lemma. □

Recall that T is a tree in an optimal Steiner forest F^* and that T spans terminals $R(T) \subseteq R$. Furthermore, terminal $w \in R(T)$ has the highest responsibility time among all terminals spanned by T. In the following, let $\bar{w}$ be the *mate* of w, i.e., $(w, \bar{w}) \in R$. From now on we will assume that there is a time $\tau_0 \in [0, r(w))$ such that $\delta(U^{\tau_0}(w)) \cap T = \emptyset$ and hence $T \subseteq E(U^{\tau_0}(w))$, where $E(U^{\tau_0}(w))$ denotes the subset of those edges in E that have both endpoints in $U^{\tau_0}(w)$. We also must have $|R^\tau(T)| = 1$ for all $\tau \in [\tau_0, r(w))$ since all vertices of $R(T)$ are in the same connected component of $\bar{F}^\tau$. Furthermore, since w is responsible until time $r(w)$ we must have $R^\tau(T) = \{w\}$ for all $\tau \in [\tau_0, r(w))$, and thus $u \prec w$ and $u \prec \bar{w}$ for all $u \in R(T) \setminus \{w, \bar{w}\}$.

Let $P_{w\bar{w}}$ be the unique $w, \bar{w}$-path in T. We define $I^\tau(T)$ as the set of responsible terminal pairs in $R^\tau(T) \setminus \{w, \bar{w}\}$ that inflict a dual load on path $P_{w\bar{w}}$ in $\texttt{KLS}(R)$ at time τ, i.e.,

$$I^\tau(T) = \{v \in R^\tau(T) \setminus \{w, \bar{w}\} \,:\, \delta(U^\tau(v)) \cap P_{w\bar{w}} \neq \emptyset\}.$$

CLAIM 4.9. *Consider a point in time τ and a terminal $v \in I^\tau(T)$. Then $U^\tau(v)$ contains neither w nor $\bar{w}$.*

Proof. By definition of $I^\tau(T)$, we know that $v \notin \{w, \bar{w}\}$. We also know that $v \prec w$ and $v \prec \bar{w}$. The claim follows as v is responsible for $U^\tau(v)$, and hence $\{w, \bar{w}\} \cap U^\tau(v) = \emptyset$. □

For a time τ and a vertex $v \in I^\tau(T)$, let $p^\tau_{w\bar{w}}(v)$ be the number of intersections of $P_{w\bar{w}}$ and $U^\tau(v)$ at time τ:

$$p^\tau_{w\bar{w}}(v) = |\delta(U^\tau(v)) \cap P_{w\bar{w}}|. \tag{4.4}$$

We use $\texttt{sl}_{w\bar{w}}$ to denote the cost of that part of $P_{w\bar{w}}$ that does not feel any dual load from any of the terminals in $R(T)$. Let l_w and $l_{\bar{w}}$ be the total load on $P_{w\bar{w}}$ coming from terminals w and $\bar{w}$, respectively. We can then express the cost of $P_{w\bar{w}}$ as

$$c(P_{w\bar{w}}) = l_w + l_{\bar{w}} + \texttt{sl}_{w\bar{w}} + \int_0^{\tau_0} \sum_{v \in I^\tau(T)} p^\tau_{w\bar{w}}(v)\, d\tau. \tag{4.5}$$

We obtain the following lemma.

LEMMA 4.10. *If there is a $\tau_0 \in [0, r(w))$ with $\delta(U^{\tau_0}(w)) \cap T = \emptyset$, then we must have*

$$\sum_{v \in R(T)} r(v) \leq c(T).$$

Proof. Similar to the proof of Lemma 4.8, consider a time $\tau < r(w)$ where $R^\tau(T)$ contains more than one terminal. The corresponding moats in $U^\tau(T)$ are pairwise disjoint by Claim 4.7, and the vertices in $R^\tau(T)$ are connected by T. Hence, each of the moats in $U^\tau(T)$ loads a distinct part of T. Moreover, using the definition of $p^\tau_{w\bar{w}}(v)$ in (4.4), for all $\tau \in [0, \tau_0)$ and $v \in I^\tau(T)$ moat $U^\tau(v)$ loads at least $p^\tau_{w\bar{w}}(v)$ edges of T.

Recall that $\texttt{sl}_{w\bar{w}}$ is the cost of the segments of $P_{w\bar{w}}$ that do not feel any load from terminals in $R(T)$. Furthermore, w loads edges of T until time τ_0, and hence we must have

$$c(T) \geq \tau_0 + \texttt{sl}_{w\bar{w}} + \int_0^{\tau_0} \sum_{v \in I^\tau(T)} (p^\tau_{w\bar{w}}(v) - 1)\, d\tau + \sum_{v \in R(T) \setminus \{w\}} r(v). \tag{4.6}$$

Observe that for all $\tau \in [0, \tau_0)$ and $v \in I^\tau(T)$, we account a total contribution of $p^\tau_{w\bar{w}}(v)$: $p^\tau_{w\bar{w}}(v) - 1$ in the first sum and 1 in the second sum, respectively.

The death time of vertex w is at most half of the cost of $P_{w\bar{w}}$. Using (4.5) we therefore obtain

$$r(w) \leq \frac{l_w + l_{\bar{w}}}{2} + \frac{\mathtt{sl}_{w\bar{w}}}{2} + \frac{1}{2} \cdot \int_0^{\tau_0} \sum_{v \in I^\tau(T)} p^\tau_{w\bar{w}}(v)\, d\tau$$

$$\leq \tau_0 + \mathtt{sl}_{w\bar{w}} + \int_0^{\tau_0} \sum_{v \in I^\tau(T)} (p^\tau_{w\bar{w}}(v) - 1)\, d\tau, \tag{4.7}$$

where the second inequality uses the fact that $\max\{l_w, l_{\bar{w}}\} \leq \tau_0$ and that by Claim 4.9, $p^\tau_{w\bar{w}}(v) \geq 2$ for all $v \in I^\tau(T)$. Combining (4.6) and (4.7) yields the lemma. □

We can now sum over all trees T in the forest F^*. Lemmas 4.8 and 4.10 together with (4.3) imply that

$$\sum_{U \in \mathcal{U}} y_U = \sum_{v \in R} r(v) = \sum_{T \in F^*} \sum_{v \in R(T)} r(v) \leq \sum_{T \in F^*} c(T) = \mathtt{opt}_R.$$

This finishes the proof of Lemma 4.4.

5. Lifted-cut LP relaxation for the Steiner forest problem. Recall that without loss of generality we let

$$R = \{(s_1, t_1), \ldots, (s_k, t_k)\}$$

such that

$$\mathtt{d}(s_1, t_1) \leq \cdots \leq \mathtt{d}(s_k, t_k).$$

As before we define a precedence order $\prec$ on R by letting $(s_i, t_i) \prec (s_j, t_j)$ iff $i \leq j$, and we extend this order to terminal vertices by letting

$$s_1 \prec t_1 \prec s_2 \prec t_2 \prec \cdots \prec s_k \prec t_k. \tag{5.1}$$

We assume that $v \prec v$ for all $v \in R$.

Let $R(U)$ be the set of terminal pairs in R that are separated by a Steiner cut $U \in \mathcal{S}$, i.e., $R(U) = \{(s,t) \in R : |\{s,t\} \cap U| = 1\}$. Consider a terminal v and let $\bar{v}$ be v's mate in the Steiner forest instance, i.e., $(v, \bar{v}) \in R$. We let $\mathcal{S}_v \subseteq \mathcal{S}$ be the set of Steiner cuts that separate v and $\bar{v}$ and for which $(v, \bar{v})$ is the highest ranked such terminal pair:

$$\mathcal{S}_v = \{U \in \mathcal{S} : v \in R(U),\ u \prec v \quad \forall\, u \in R(U)\}. \tag{5.2}$$

We also let $\mathcal{N}_v \subseteq \mathcal{N}$ be the set of all non-Steiner cuts containing v and $\bar{v}$ where $(v, \bar{v})$ is the terminal pair of highest rank:

$$\mathcal{N}_v = \{U \in \mathcal{N} : \{v, \bar{v}\} \subseteq U \cap R,\ (u, \bar{u}) \prec (v, \bar{v}) \quad \forall\ (u, \bar{u}) \in U \cap R\}.$$

Recall that we define $\mathcal{U} = \mathcal{S} \cup \mathcal{N}$ as the set of all Steiner and non-Steiner cuts. We then say that a terminal $v \in R$ is *responsible* for a cut $U \in \mathcal{U}$ if $U \in \mathcal{S}_v \cup \mathcal{N}_v$. Observe that for a non-Steiner cut $U \in \mathcal{N}$ two terminals are responsible. Also note that the responsibility notion introduced here differs from the one that was used in section 4 in that a terminal can only be responsible for a Steiner cut if the cut separates it from its mate.

The dual of the lifted-cut relaxation for the Steiner forest problem is as follows:

$$\text{(LC-D)} \qquad \mathtt{opt}_{\text{LC-D}} = \max \quad \sum_{U \in \mathcal{U}} y_U$$

$$\text{(5.3)} \qquad \text{s.t.} \quad \sum_{U \in \mathcal{U} \colon e \in \delta(U)} y_U \leq c(e) \qquad \forall e \in E,$$

$$\text{(5.4)} \qquad \sum_{U \in \mathcal{S}_v} y_U + \sum_{U \in \mathcal{N}_v} y_U \leq \mathtt{d}(v) \qquad \forall v \in R,$$

$$y_U \geq 0 \qquad \forall U \in \mathcal{U}.$$

Notice that a feasible solution to (LC-D) may assign positive values to non-Steiner cuts $U \in \mathcal{N}$. The constraints of type (5.4) are necessary as the objective function value of (LC-D) would be unbounded in their absence.

The LP dual of (LC-D) has variables x_e for every edge $e \in E$ and variables x_v for every terminal $v \in R$:

$$\text{(LC-P)} \qquad \mathtt{opt}_{\text{LC-P}} = \min \quad \sum_{e \in E} c(e) \cdot x_e + \sum_{v \in R} \mathtt{d}(v) \cdot x_v$$

$$\text{(5.5)} \qquad \text{s.t.} \quad \sum_{e \in \delta(U)} x_e + x_v \geq 1 \qquad \forall U \in \mathcal{S}_v, \ \forall v \in R,$$

$$\text{(5.6)} \qquad \sum_{e \in \delta(U)} x_e + x_v + x_{\bar{v}} \geq 1 \qquad \forall U \in \mathcal{N}_v, \ \forall v \in R,$$

$$x_e, \ x_v \geq 0 \qquad \forall e \in E, \ \forall v \in R.$$

LEMMA 5.1. *Let $\{x_e, x_v\}_{e \in E, v \in R}$ be an integral solution that is feasible for* (LC-P)*. Then there is a feasible Steiner forest of cost at most*

$$\sum_{e \in E} c(e) \cdot x_e + \sum_{v \in R} \mathtt{d}(v) \cdot x_v.$$

Proof. Given $\{x_e, x_v\}_{e \in E, v \in R}$, define $F = \{e \in E : x_e = 1\}$. The total cost $c(F)$ of F is $\sum_{e \in E} c(e) \cdot x_e$. F is not necessarily a feasible Steiner forest since there might exist a Steiner cut $U \in \mathcal{S}$ with no crossing edge, i.e., $\delta(U) \cap F = \emptyset$. Let $U \in \mathcal{S}_v$ be such a set and let $\bar{v}$ be the mate of v. Constraint (5.5) for U and v implies that $x_v = 1$ in this case. Next consider the complement $\bar{U} = V \setminus U$. It can be seen that $\bar{v}$ is responsible for $\bar{U}$ and hence, $\bar{U} \in \mathcal{S}_{\bar{v}}$. As no edge crosses $\bar{U}$, constraint (5.5) for $\bar{U}$ and $\bar{v}$ implies that $x_{\bar{v}} = 1$. Therefore, we can add all edges along the shortest $v, \bar{v}$-path to F at a cost of at most $2\mathtt{d}(v, \bar{v})$. Observe that this addition is sufficient to satisfy all Steiner cuts in $\mathcal{S}_v$, so we need only add this path once for v and $\bar{v}$. We can therefore repeat this procedure for all remaining terminals $v \in R$ for which there exists a Steiner cut $U \in \mathcal{S}_v$ that is not crossed by F. The total cost in this solution incurred by the additional paths is not more than $\sum_{v \in R} \mathtt{d}(v) \cdot x_v$, which completes the proof. □

In the remainder of this section we prove the following theorem.

Theorem 5.2. *The objective value of an optimal solution to the lifted-cut relaxation* (LC-P) *is at most the cost of any feasible Steiner forest for the given instance. The dual linear program* (LC-D) *is stronger than the well-known undirected-cut relaxation for the Steiner forest problem. Moreover, the dual solution computed by* KLS *is feasible for* (LC-D). *There exist instances for which the IP/LP gap is about* 2.

The following lemma relates the cost of any feasible solution for the given Steiner forest instance to the objective function value of an optimal solution for (LC-P).

Lemma 5.3. *Let F be a feasible solution for the underlying Steiner forest instance. We can then construct a half-integral solution $\{x_e, x_v\}_{e \in E, v \in R}$ that is feasible for* (LC-P) *and satisfies*

$$\sum_{e \in E} c(e) \cdot x_e + \sum_{v \in R} \mathtt{d}(v) \cdot x_v \leq c(F).$$

In particular, this implies that $\mathtt{opt}_{LC\text{-}P} \leq \mathtt{opt}_R$.

Proof. Let T be a tree in F. We use $E(T)$ and $V(T)$ to refer to the edges and vertices of T, respectively. We construct a solution $\{x_e, x_v\}_{e \in E, v \in R}$ that is feasible for (LC-P) and show that for each tree $T \in F$

$$\sum_{e \in E(T)} c(e) \cdot x_e + \sum_{v \in R \cap V(T)} \mathtt{d}(v) \cdot x_v \leq c(T).$$

The lemma then follows by summing over all trees in F.

Consider a tree $T \in F$. Let $(w, \bar{w})$ be the terminal pair such that w and $\bar{w}$ are responsible for the non-Steiner cut $V(T)$. Moreover, let $P_{w\bar{w}}$ denote the unique $w, \bar{w}$-path in T. We set $x_e = 1/2$ for each edge $e \in E(P_{w\bar{w}})$ and $x_e = 1$ for each edge $e \in E(T) \setminus E(P_{w\bar{w}})$. Moreover, we assign $x_w = x_{\bar{w}} = 1/2$ and $x_v = 0$ for all terminals $v \in (R \cap V(T)) \setminus \{w, \bar{w}\}$. By definition (3.1) of death time, $\mathtt{d}(w, \bar{w}) \leq c(P_{w\bar{w}})/2$. Thus, the objective value for x on T is

$$\sum_{e \in E(T)} c(e) \cdot x_e + \sum_{v \in R \cap V(T)} \mathtt{d}(v) \cdot x_v = c(T) - \frac{c(P_{w\bar{w}})}{2} + \mathtt{d}(w, \bar{w}) \leq c(T).$$

It remains to be shown that x is feasible for (LC-P). We show for each tree T in F and for all $v \in R \cap V(T)$ that x satisfies the cut requirements of constraints (5.5) and (5.6) for sets $U \in \mathcal{S}_v \cup \mathcal{N}_v$.

Consider a cut $U \in \mathcal{S}_v$ for some $v \in R \cap V(T)$. If $v \in \{w, \bar{w}\}$, constraint (5.5) holds since U intersects $P_{w\bar{w}}$ and $x_v = 1/2$. Now let $v \notin \{w, \bar{w}\}$. As $U \in \mathcal{S}_v$ and $v \prec w$, by assumption, it follows that either $\{w, \bar{w}\} \subseteq U$ or $\{w, \bar{w}\} \cap U = \emptyset$. We also have $\bar{v} \notin U$. As T connects v and $\bar{v}$, it can be seen that U either intersects at least one edge e of T that is not on $P_{w\bar{w}}$ (and hence $x_e = 1$) or intersects at least two edges e_1 and e_2 on $P_{w\bar{w}}$ (and therefore $x_{e_1} = x_{e_2} = 1/2$). Thus, constraint (5.5) holds in this case as well.

Next consider a non-Steiner cut $U \in \mathcal{N}_v$ for terminal $v \in R \cap V(T)$. If $v \notin \{w, \bar{w}\}$, then $\{w, \bar{w}\} \cap U = \emptyset$ and U crosses at least one edge of T that is not on $P_{w\bar{w}}$ or at least two edges of $P_{w\bar{w}}$. Hence constraint (5.6) holds. Otherwise, U may cross no edge of T but $x_w + x_{\bar{w}} = 1$ and thus (5.6) is satisfied. □

Running algorithm KLS on terminal set R yields a cost share $\xi_R(s, t)$ for all $(s, t) \in R$. It also returns a dual solution $\{y_U\}_{U \in \mathcal{U}}$ such that $\sum_{(s,t) \in R} \xi_R(s, t) = \sum_{U \in \mathcal{U}} y_U$. It is easy to verify that y is feasible for (LC-D). Lemma 5.3 therefore yields an alternate proof of the competitiveness of KLS.

Corollary 5.4. *ξ satisfies competitiveness; i.e.,*

$$\sum_{(s,t)\in R} \xi_R(s,t) = \sum_{U\in\mathcal{U}} y_U \leq \mathtt{opt}_{\mathrm{LC-D}} = \mathtt{opt}_{\mathrm{LC-P}} \leq \mathtt{opt}_R.$$

The next lemma shows that (LC-D) is at least as strong as the standard LP dual (D).

Lemma 5.5. *Let $\{y_U\}_{U\in\mathcal{S}}$ be a feasible dual solution for* (D). *Then there is a feasible dual solution $\{y'_U\}_{U\in\mathcal{U}}$ for* (LC-D) *with*

$$\sum_{U\in\mathcal{S}} y_U \leq \sum_{U\in\mathcal{U}} y'_U.$$

This implies that $\mathtt{opt}_{\mathrm{D}} \leq \mathtt{opt}_{\mathrm{LC-D}}$.

Proof. Let y be a feasible solution for (D). The sets $\mathcal{S}_v$ for terminals $v \in R$ form a partition of $\mathcal{S}$: $\mathcal{S} = \bigcup_{v\in R} \mathcal{S}_v$. We define a candidate dual solution y' for (LC-D) as follows: for a set $U \in \mathcal{S}$, let $\bar{U} \in \mathcal{S}$ be its complement and define

$$y'_U = y'_{\bar{U}} = \frac{y_U + y_{\bar{U}}}{2}.$$

Let $y'_U = 0$ for all non-Steiner cuts $U \in \mathcal{N}$.

We claim that y' satisfies all constraints of type (5.3). To see this, consider an edge $e \in E$ and observe that

$$\sum_{U\in\mathcal{S}:e\in\delta(U)} y'_U = \sum_{U\in\mathcal{S}:e\in\delta(U)} \frac{y_U + y_{\bar{U}}}{2} = \sum_{U\in\mathcal{S}:e\in\delta(U)} y_U,$$

where the last equality uses the fact that U is a Steiner cut iff its complement is. The dual feasibility of y for (D) shows that y' satisfies (5.3).

We will now show that y' also satisfies all constraints of type (5.4). Assume for the sake of contradiction that y' violates constraint (5.4) for some terminal $v \in R$. We then have

$$\sum_{U\in\mathcal{S}_v} y'_U + \sum_{U\in\mathcal{N}_v} y'_U = \sum_{U\in\mathcal{S}_v} y'_U > \mathtt{d}(v) = c(P_{v\bar{v}})/2, \tag{5.7}$$

where $c(P_{v\bar{v}})$ is the cost of a minimum-cost $v,\bar{v}$-path in G.

Consider a Steiner cut $U \in \mathcal{S}$ and observe that U and its complement $\bar{U}$ separate the same set of terminal pairs. Therefore, $U \in \mathcal{S}_v$ iff $\bar{U} \in \mathcal{S}_{\bar{v}}$ for a terminal pair $(v,\bar{v}) \in R$, and thus,

$$\sum_{U\in\mathcal{S}_v} y'_U = \sum_{U\in\mathcal{S}_v} \frac{y_U + y_{\bar{U}}}{2} = \sum_{U\in\mathcal{S}_{\bar{v}}} y'_U. \tag{5.8}$$

Together with (5.7), this implies that

$$\sum_{U\in\mathcal{S}_v} y'_U + \sum_{U\in\mathcal{S}_{\bar{v}}} y'_U > c(P_{v\bar{v}}).$$

On the other hand, adding the constraints of type (2.2) for all edges $e \in E(P_{v\bar{v}})$ yields

$$\sum_{U\in\mathcal{S}_v} y'_U + \sum_{U\in\mathcal{S}_{\bar{v}}} y'_U \leq \sum_{U\in\mathcal{S}} |\delta(U)\cap P_{v\bar{v}}| \cdot y_U = \sum_{e\in E(P_{v\bar{v}})} \sum_{U\in\mathcal{S}:\, e\in\delta(U)} y_U \leq c(P_{v\bar{v}}),$$

and this is a contradiction. ∎

The dual of the lifted-cut relaxation is stronger than the standard LP dual (D).

LEMMA 5.6. *There exist instances for which* $\mathtt{opt}_{\mathrm{D}} < \mathtt{opt}_{\mathrm{LC-D}}$.

Proof. Consider a cycle of $2n$ vertices with unit edge costs. Let $V = \{v_1, \dots, v_{2n}\}$ and define $R = \{(v_1, v_j)\}_{2 \le j \le 2n}$. The cost of an optimal solution is $\mathtt{opt}_R = 2n - 1$.

We define a dual solution as follows: $y_{\{v\}} = 1/2$ for each $v \in V$ and $y_U = 0$ for all other sets $U \in \mathcal{S}$. Clearly, $\{y_U\}_{U \in \mathcal{S}}$ is a feasible solution to (D). It can easily be verified that this is an optimal solution for (D): If we set $x_e = 1/2$ for each edge e of the cycle, we obtain a feasible solution for the LP relaxation (LP) having the same objective function value. Thus, $\mathtt{opt}_{\mathrm{D}} = n$.

For (LC-D), on the other hand, we can define a dual solution $y'_{\{v\}} = 1/2$ for each $v \in V$, $y'_V = n/2 - 1/2$, and $y'_U = 0$ for all other sets $U \in \mathcal{U}$. It is easy to verify that y' is a feasible solution for (LC-D). We conclude that

$$\mathtt{opt}_{\mathrm{LC-D}} \ge \sum_{U \in \mathcal{U}} y'_U = \frac{3n}{2} - \frac{1}{2}.$$

The latter term is strictly larger than n if $n > 1$. ∎

Unfortunately, as with the undirected cut formulation for the Steiner forest problem, the IP/LP gap of the lifted-cut relaxation is close to 2 for certain instances.

LEMMA 5.7. *There exist instances for which* $\mathtt{opt}_R/\mathtt{opt}_{\mathrm{LC-P}} = 2 - 2/(k+1)$, *where k is the number of terminal pairs.*

Proof. Consider a clique K_n with vertices $V = \{v_1, v_2, \dots, v_n\}$ and unit edge costs. Define $R = \{(v_1, v_j)\}_{2 \le j \le n}$. Without loss of generality, let $(w, \bar{w}) = (v_1, v_2)$ be the highest ranked terminal pair among all terminal pairs in R.

Consider path $P = (v_2, v_3, \dots, v_n, v_1)$ spanning all vertices of K_n. The following is a feasible solution for (LC-P): set $x_w = x_{\bar{w}} = 1/2$ and $x_v = 0$ for all $v \in V \setminus \{w, \bar{w}\}$, and set $x_e = 1/2$ for all edges $e \in E(P)$ and $x_e = 0$ for all edges $e \notin E(P)$. This solution satisfies constraints (5.5) and (5.6). The objective function value for x is $n/2$. Next consider the following dual solution. Let $y_{\{v\}} = 1/2$ for all $v \in V$ and $y_U = 0$ for all other $U \in \mathcal{U}$. Then y satisfies constraints (5.3) and (5.4). The objective value of y is $n/2$, and thus x and y are optimal solutions to (LC-P) and (LC-D), respectively.

Clearly, the optimal solution $\mathtt{opt}_R$ has cost $n - 1$. The ratio between $\mathtt{opt}_R$ and $\mathtt{opt}_{\mathrm{LC-D}}$ is $2 - 2/n$. Since $k = n - 1$, the lemma follows. ∎

6. Algorithmic consequences of the lifted-cut relaxation. In this section we show that, for some instances of the Steiner forest problem, we can use the additional strength of the lifted-cut relaxation in order to prove that algorithm AKR returns a Steiner forest of cost strictly less than $(2 - 1/k)\mathtt{opt}_R$.

Consider an instance of the Steiner forest problem with terminal set R. Assume that algorithm AKR, when executed on this instance, finishes at time $\tau^* \ge 0$ with forest F and feasible dual solution $\{y_U\}_{U \in \mathcal{S}}$. Let $U_1, \dots, U_p$ be the connected components of $\bar{F}^{\tau^*}$ and define $R_i \subseteq R$ to be the set of terminal pairs contained in U_i for all $1 \le i \le p$. Further let (s_i, t_i) be the terminal pair in R_i of highest rank according to the precedence order $\prec$ defined in (5.1), i.e.,

$$(s, t) \prec (s_i, t_i)$$

for all $(s, t) \in R_i$ and for all $1 \le i \le p$. For $1 \le i \le p$, we now define the combined *slack* $\mathtt{sl}_i$ of the constraints (5.4) for the terminal vertices s_i and t_i with respect to dual solution y:

$$\mathtt{sl}_i = 2\mathtt{d}(s_i, t_i) - \sum_{U \in \mathcal{S}_{s_i} \cup \mathcal{S}_{t_i}} y_U.$$

Let $\mathtt{sl}_R = \max_{1 \leq i \leq p} \mathtt{sl}_i$ be the slack of the given instance of the Steiner forest problem.

THEOREM 6.1. *The forest F returned by* AKR *for an instance of the Steiner forest problem with terminal pairs R has cost at most*

$$\left(2 - \frac{1}{k}\right)\left(\frac{Y}{Y + \mathtt{sl}_R/2}\right)\mathtt{opt}_R,$$

where Y is the objective function value of the dual computed by AKR.

Proof. From the proof of Lemma 5.5 (see (5.8)) we know that we may assume without loss of generality that y is symmetric; i.e., we may assume that

$$\sum_{U \in \mathcal{S}_s} y_U = \sum_{U \in \mathcal{S}_t} y_U$$

for all $(s, t) \in R$.

We observe that the proof of Lemma 5.5 works for any fixed precedence order $\prec$ on R; in particular, at no point in the proof of this lemma do we use the fact that $(s, t) \prec (s', t')$ implies $\mathtt{d}(s, t) \leq \mathtt{d}(s', t')$.

Choose $1 \leq q \leq p$ such that $\mathtt{sl}_q = \max_{1 \leq i \leq p} \mathtt{sl}_i$. We will now define an alternative order $\prec'$ on R in which the terminal pairs in R_q have highest rank. The order on terminal pairs in $R \setminus R_q$ and the order within R_q is that induced by $\prec$. Formally, consider two terminal pairs $(s, t), (s', t') \in R$. We let $(s, t) \prec' (s', t')$ iff

- $(s, t) \prec (s', t')$ and either $\{(s, t), (s', t')\} \subseteq R \setminus R_q$ or $\{(s, t), (s', t')\} \subseteq R_q$, or
- $(s, t) \in R \setminus R_q$ and $(s', t') \in R_q$.

Similar to the definition of $\mathcal{S}_v$ in (5.2), we let $\mathcal{S}'_v$ be the set of Steiner cuts that separate v and its mate $\bar{v}$ and for which $(v, \bar{v})$ has highest $\prec'$-rank among all such terminal pairs. The definition of $\prec'$ implies that $(s, t) \prec (s', t')$ iff $(s, t) \prec' (s', t')$ for all $\{(s, t), (s', t')\} \subseteq R_i$ for all $1 \leq i \leq p$. Therefore, we also must have

$$\sum_{U \in \mathcal{S}_v} y_U = \sum_{U \in \mathcal{S}'_v} y_U$$

for all terminals $v \in R$. Specifically, this and the symmetry of y imply that

$$\sum_{U \in \mathcal{S}'_{s_q}} y_U + \sum_{U \in \mathcal{N}_{s_q}} y_U \leq \mathtt{d}(s_q, t_q) - \frac{\mathtt{sl}_q}{2},$$

$$\sum_{U \in \mathcal{S}'_{t_q}} y_U + \sum_{U \in \mathcal{N}_{t_q}} y_U \leq \mathtt{d}(s_q, t_q) - \frac{\mathtt{sl}_q}{2},$$

where $\mathcal{N}_{s_q} = \mathcal{N}_{t_q}$ and $y_U = 0$ for all $U \in \mathcal{N}_{s_q}$. Finally notice that $V \in \mathcal{N}_{s_q}$ as (s_q, t_q) is the highest ranked terminal pair in R under $\prec'$. We now let $y'_U = y_U$ for all Steiner cuts $U \in \mathcal{S}$ and we define $y'_V = \mathtt{sl}_q/2$. It is not hard to see that y' is feasible for the lifted-cut dual (LC-D) for the given instance of the Steiner forest problem.

In the following, we use Y as a short for $\sum_{U\in\mathcal{S}} y_U$. We then have

$$\left(1+\frac{\mathtt{sl}_R}{2Y}\right)\cdot Y = y'_V + \sum_{U\in\mathcal{S}} y'_U \leq \mathtt{opt}_R,$$

and this together with Theorem 2.1 implies

$$c(F) \leq \left(2-\frac{1}{k}\right)\cdot Y \leq \left(2-\frac{1}{k}\right)\left(\frac{Y}{Y+\mathtt{sl}_R/2}\right)\mathtt{opt}_R. \qquad \square$$

Suppose now that we are given an instance of the Steiner tree problem with terminal set R and root vertex r. Let Δ_R be the maximum distance among any two terminals in $R\cup\{r\}$. We call Δ_R the *diameter* of the given instance. Let r' be an arbitrary terminal in $R\cup\{r\}$ such that there exists a terminal $u\in R\cup\{r\}$ with $c(r',u)=\Delta_R$. The Steiner forest instance with terminal pairs

$$R' = \{(u,r') \,:\, u\in R\cup\{r\}\}$$

is easily seen to be equivalent to the given instance of the Steiner tree problem. Suppose again that AKR finishes at time τ^* when run on this instance. It is not hard to convince oneself that the slack $\mathtt{sl}_{R'}$ of this instance is

$$\mathtt{sl}_{R'} = \Delta_R - \tau^*.$$

We therefore obtain the following corollary of Theorem 6.1.

COROLLARY 6.2. *Given an instance of the Steiner tree problem with terminal set R,* AKR *returns a tree T of cost at most*

$$\left(2-\frac{1}{|R|}\right)\left(\frac{Y}{Y+(\Delta_R-\tau^*)/2}\right)\mathtt{opt}_R,$$

where Y is the objective function value of the dual computed by AKR.

7. A lower bound for the Steiner tree game. We next prove that no cross-monotonic cost sharing method for the Steiner tree game can achieve a budget balance factor better than 2.

THEOREM 7.1. *There is no $(2-\epsilon)$-budget balanced, cross-monotonic cost sharing method for the Steiner tree game for any $\epsilon>0$.*

The tools used in this section are adaptations of those used in [9]. In particular, we consider any given cross-monotonic cost sharing method ξ for the Steiner tree game and show that there is an instance of the game where the sum of the cost shares of all players is considerably smaller than the cost of an optimal solution. Instead of using a probabilistic argument similar to the one described in [9], we use a more direct (but ultimately equivalent) proof based on convex combinations.

The family of instances used in our proof resembles the one used for the facility location lower bound in [9]. We construct an undirected graph $G=(V,E)$. First we describe the vertex set. There are k pairwise disjoint sets A_i, $i=1,\ldots,k$, each of which contains m vertices. Every one of these vertices corresponds to a player who wants to connect this vertex with a root vertex (which is different from the vertices in A_i). The set of all players that have a vertex associated with them in A_i is denoted by $\mathcal{A}_i$. The set of all players is $\mathcal{R}=\bigcup_{i=1}^{k}\mathcal{A}_i$.

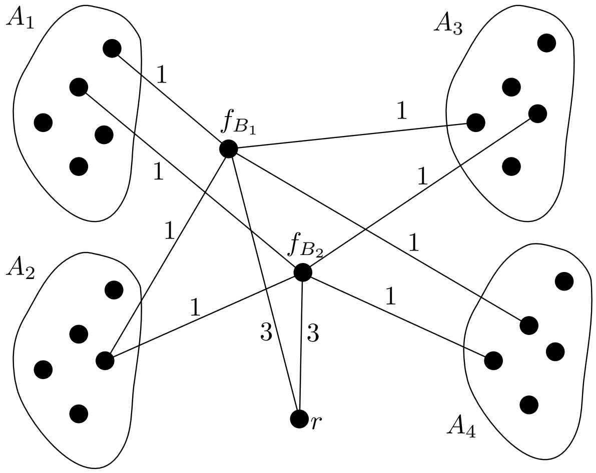


FIG. 7.1. *Example of G in which $k = 4$, $m = 5$, and only two of the f_B are drawn.*

Let $\mathcal{B}$ be the collection of all sets with exactly one element from each of the A_i, i.e.,

$$\mathcal{B} = \big\{\{a_1, \ldots, a_k\} \,:\, a_i \in A_i,\ i = 1, \ldots, k\big\}.$$

For each set $B \in \mathcal{B}$, we introduce a unique vertex f_B and edges (b, f_B) of cost 1 for all vertices $b \in B$. The distance to the vertices not in B is, by the triangle inequality, equal to 3. Finally, there is, for each B, an edge (f_B, r) of cost 3. See Figure 7.1.

The following lemma argues that we may assume that ξ is *symmetric*, i.e., that it does not differentiate between players from the same set $\mathcal{A}_i$.

LEMMA 7.2. *Suppose that there is an α-budget balanced cost sharing method for the Steiner tree game. Then there is also an α-budget balanced cost sharing method that satisfies, for every subset $\mathcal{Q} \subseteq \mathcal{R}$ of players,*

$$\xi_{\mathcal{Q}}(c) = \xi_{\mathcal{Q}}(d)$$

for all $c, d \in \mathcal{Q} \cap \mathcal{A}_i$ and for all $1 \leq i \leq k$. Moreover, for all $c \in \mathcal{Q} \cap \mathcal{A}_i$ and for all $d \in \mathcal{A}_i \setminus \mathcal{Q}$,

$$\xi_{\mathcal{Q}}(c) = \xi_{(\mathcal{Q}\setminus\{c\})\cup\{d\}}(d).$$

Proof. Let $\tilde{\xi}$ be an α-budget balanced cost sharing method for the Steiner tree game. Let Π be the set of permutations of $\mathcal{R}$ that leave the $\mathcal{A}_i$ invariant; i.e., if $\pi \in \Pi$ and $c \in \mathcal{A}_i$, then $\pi(c) \in \mathcal{A}_i$. Then $|\Pi| = (m!)^k$. Write $\pi(\mathcal{Q}) := \{\pi(c) \,:\, c \in \mathcal{Q}\}$. Define, for $c \in \mathcal{R}$,

$$\xi_{\mathcal{Q}}(c) := \sum_{\pi \in \Pi} \frac{1}{(m!)^k} \tilde{\xi}_{\pi(\mathcal{Q})}\big(\pi(c)\big).$$

Notice that, for a player $c \notin \mathcal{Q}$, the value $\xi_{\mathcal{Q}}(c)$ is 0 as $\pi(c) \notin \pi(\mathcal{Q})$ for all $\pi \in \Pi$. Since we average over all player permutations, for all $1 \leq i \leq k$ and for any two players $c, d \in \mathcal{A}_i \cap \mathcal{Q}$, we have $\xi_{\mathcal{Q}}(c) = \xi_{\mathcal{Q}}(d)$. It remains to show that ξ is cross-monotonic and α-budget balanced.

Consider adding a player d to set $\mathcal{Q}$. We have to argue that the cost share of an individual player cannot increase. For a player $c \in \mathcal{Q}$ we see that

$$\xi_{\mathcal{Q}\cup\{d\}}(c) = \sum_{\pi\in\Pi} \frac{1}{(m!)^k} \tilde{\xi}_{\pi(\mathcal{Q}\cup\{d\})}(\pi(c)) \leq \sum_{\pi\in\Pi} \frac{1}{(m!)^k} \tilde{\xi}_{\pi(\mathcal{Q})}(\pi(c)) = \xi_{\mathcal{Q}}(c).$$

This follows since $\pi(\mathcal{Q}\cup\{d\}) = \pi(\mathcal{Q}) \cup \{\pi(d)\}$, and hence the cross-monotonicity of $\tilde{\xi}$ can be applied to each term.

Now we show α-budget balance. To this end we must specify which solution is returned by the algorithm. If we denote with S^π the solution returned by cost sharing method $\tilde{\xi}$ when run on set $\pi(\mathcal{Q})$, we return the solution $S \in \{S^\pi : \pi \in \Pi\}$ with cost $c(S) = \min_{\pi\in\Pi} c(S^\pi)$.

Of course this solution is not necessarily feasible for the original player set, but because of the symmetry of the instance there is a graph isomorphism that maps the solution back to a feasible one without changing the cost.

Now we can write

$$\begin{aligned}\sum_{c\in\mathcal{Q}} \xi_{\mathcal{Q}}(c) &= \sum_{c\in\mathcal{Q}} \sum_{\pi\in\Pi} \frac{1}{(m!)^k} \tilde{\xi}_{\pi(\mathcal{Q})}(\pi(c)) = \sum_{\pi\in\Pi} \frac{1}{(m!)^k} \sum_{c\in\mathcal{Q}} \tilde{\xi}_{\pi(\mathcal{Q})}(\pi(c)) \\ &\geq \sum_{\pi\in\Pi} \frac{1}{(m!)^k} \frac{1}{\alpha} \cdot c(S^\pi) \geq \sum_{\pi\in\Pi} \frac{1}{(m!)^k} \frac{1}{\alpha} \cdot c(S) = \frac{1}{\alpha} \cdot c(S).\end{aligned}$$

Competitiveness can be proved using a similar line of reasoning: the cost of the optimal solution must be the same in any permutation. With that, the proof is complete. □

Now suppose we are given a symmetric cost sharing method ξ. From this point on we will identify players and vertices to avoid complication of notation. Ask the algorithm for cost shares for a subset of players $\{a_1, \ldots, a_k\}$, where $a_i \in A_i$. By construction of the graph, all these terminals can connect to vertex $f_{\{a_1,\ldots,a_k\}}$ at cost 1, at which point they are only 3 units away from the root. Hence there is a solution of cost $k+3$ for this subset. Competitiveness states that

$$\sum_{j=1}^{k} \xi_{\{a_1,\ldots,a_k\}}(a_j) \leq \mathtt{opt}_{\{a_1,\ldots,a_k\}} \leq k+3.$$

Therefore, there must be at least one index i such that $\xi_{\{a_1,\ldots,a_k\}}(a_i) \leq (k+3)/k$, and Lemma 7.2 implies that

$$\xi_{\{a_1,\ldots,a_{i-1},c,a_{i+1},\ldots,a_k\}}(c) \leq (k+3)/k \tag{7.1}$$

for all $c \in A_i$.

For this index i we consider the instance with subset $Q = \{a_1, \ldots, a_k\} \cup A_i$. We bound the sum of the cost shares for this set as follows:

$$\sum_{c\in Q} \xi_Q(c) = \sum_{c\in A_i} \xi_Q(c) + \sum_{j\neq i} \xi_Q(a_j)$$

$$\leq \sum_{c\in A_i} \xi_{\{a_1,\ldots,a_{i-1},c,a_{i+1},\ldots,a_k\}}(c) + \sum_{j\neq i} \xi_{\{a_1,\ldots,a_{i-1},a_{i+1},\ldots,a_k\}}(a_j) \tag{7.2}$$

$$\leq m \cdot \frac{k+3}{k} + k + 2. \tag{7.3}$$

The first inequality is an application of cross-monotonicity; the second follows from (7.1) and the fact that there is a solution of cost $k+2$ for a set

$$\{a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_k\}$$

of players where $a_j \in A_j$.

Due to the large amount of symmetry in this instance, we can in fact describe the optimal solution.

LEMMA 7.3. *The optimal solution for connecting the players in a set Q, as defined above, to the root has cost $2m + k + 1$.*

Proof. We observed above that connecting all terminals $\{a_1, \dots, a_k\}$ via $f_{\{a_1,\dots,a_k\}}$ to the root has cost $k+3$. Fix a terminal $a_j \in Q$ with $a_j \notin A_i$. Each of the remaining $m-1$ terminals in $A_i \setminus \{a_i\}$ can connect to a_j at cost 2. Thus, $\texttt{opt}_Q \leq k+3+2(m-1) = 2m + k + 1$.

We next show that $2m+k+1$ is a lower bound on the optimal cost. Suppose F is the set of vertices f_B, $B \in \mathcal{B}$, that are used to connect all terminals in Q to the root r, and define $f = |F|$. Clearly, $1 \leq f \leq m$. The cost of connecting all vertices in F to the root is $3f$. Moreover, connecting all $k-1$ terminals in $Q \setminus A_i$ to F has cost at least $k-1$. At most f terminals in A_i are adjacent to a vertex in F, and the total cost of connecting these terminals to F is f. The remaining $m-f$ terminals in A_i are not adjacent to any of the f vertices in F, and therefore the cost of connecting these terminals to F is at least $2(m-f)$. Hence, the cost of connecting all terminals in Q via vertices in F is at least

$$3f + k - 1 + f + 2(m-f) = 2m + k + 2f - 1 \geq 2m + k + 1. \qquad \square$$

Combining Lemma 7.3 with inequality (7.3), we can now prove Theorem 7.1.

Proof Theorem 7.1. The ratio between the cost shares of players in the subset Q as defined above and the cost of the network they use can be bounded as follows:

$$\frac{\sum_{c \in Q} \xi_Q(c)}{c(Q)} \leq \frac{\sum_{c \in Q} \xi_Q(c)}{\texttt{opt}_Q} \leq \frac{m\frac{k+3}{k} + k + 2}{2m + k + 1} = \frac{k^2 + 4k + 2}{2k^2 + k + 1},$$

where the last equality holds if we choose $m = k^2$. This ratio tends to $1/2$ as $k \to \infty$, which completes the proof. $\square$

REFERENCES

[1] A. AGARWAL AND M. CHARIKAR, *On the advantage of network coding for improving network throughput*, in Proceedings of the IEEE Information Theory Workshop, San Antonio, TX, 2004, pp. 247–249.

[2] A. AGRAWAL, P. KLEIN, AND R. RAVI, *When trees collide: An approximation algorithm for the generalized Steiner problem on networks*, SIAM J. Comput., 24 (1995), pp. 440–456.

[3] Y. P. ANEJA, *An integer linear programming approach to the Steiner problem in graphs*, Networks, 10 (1980), pp. 167–178.

[4] S. ARORA, C. LUND, R. MOTWANI, M. SUDAN, AND M. SZEGEDY, *Proof verification and the hardness of approximation problems*, J. ACM, 45 (1998), pp. 501–555.

[5] M. BERN AND P. PLASSMANN, *The Steiner problem with edge lengths* 1 *and* 2, Inform. Process. Lett., 32 (1989), pp. 171–176.

[6] J. EDMONDS, *Optimum branchings*, J. Res. Nat. Bur. Standards Sect. B, 71B (1967), pp. 233–240.

[7] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability*, W. H. Freeman and Co., San Francisco, CA, 1979.

[8] M. X. Goemans and D. P. Williamson, *A general approximation technique for constrained forest problems*, SIAM J. Comput., 24 (1995), pp. 296–317.

[9] N. Immorlica, M. Mahdian, and V. S. Mirrokni, *Limitations of cross-monotonic cost sharing schemes*, in Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, SIAM, Philadelphia, 2005, pp. 602–611.

[10] K. Jain and V. V. Vazirani, *Applications of approximation algorithms to cooperative games*, in Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, ACM, New York, 2001, pp. 364–372.

[11] K. Kent and D. Skorin-Kapov, *Population monotonic cost allocation on MST's*, in Proceedings of the 6th International Conference on Operational Research, Croatian Oper. Res. Soc., Zagreb, 1996, pp. 43–48.

[12] J. Könemann, S. Leonardi, and G. Schäfer, *A group-strategyproof mechanism for Steiner forests*, in Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, SIAM, Philadelphia, 2005, pp. 612 – 619.

[13] J. Könemann, S. Leonardi, G. Schäfer, and S. van Zwam, *From primal-dual to cost shares and back: A stronger LP relaxation for the Steiner forest problem*, in Proceedings of the 32nd International Colloquium on Automata, Languages and Programming, Lecture Notes in Comput. Sci. 3580, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 930–942.

[14] H. Moulin and S. Shenker, *Strategyproof sharing of submodular costs: Budget balance versus efficiency*, Econom. Theory, 18 (2001), pp. 511–533.

[15] M. Pál and É. Tardos, *Group strategyproof mechanisms via primal-dual algorithms*, in Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 2003, pp. 584–593.

SIAM J. COMPUT.
Vol. 37, No. 5, pp. 1342–1372

THE FORGETRON: A KERNEL-BASED PERCEPTRON ON A BUDGET[*]

OFER DEKEL[†], SHAI SHALEV-SHWARTZ[†], AND YORAM SINGER[‡]

Abstract. The Perceptron algorithm, despite its simplicity, often performs well in online classification tasks. The Perceptron becomes especially effective when it is used in conjunction with kernel functions. However, a common difficulty encountered when implementing kernel-based online algorithms is the amount of memory required to store the online hypothesis, which may grow unboundedly as the algorithm progresses. Moreover, the running time of each online round grows linearly with the amount of memory used to store the hypothesis. In this paper, we present the *Forgetron* family of kernel-based online classification algorithms, which overcome this problem by restricting themselves to a predefined memory budget. We obtain different members of this family by modifying the kernel-based Perceptron in various ways. We also prove a unified mistake bound for all of the Forgetron algorithms. To our knowledge, this is the first online kernel-based learning paradigm which, on one hand, maintains a *strict* limit on the amount of memory it uses and, on the other hand, entertains a relative mistake bound. We conclude with experiments using real datasets, which underscore the merits of our approach.

Key words. online classification, kernel methods, the Perceptron algorithm, learning theory

AMS subject classifications. 68T05, 68Q32

DOI. 10.1137/060666998

1. Introduction.

The introduction of the Support Vector Machine (SVM) [11] sparked a widespread interest in kernel methods as a means of solving binary classification problems. Although SVM was initially stated as a batch-learning technique, it significantly influenced the development of kernel methods in the online-learning setting. Online classification algorithms that can incorporate kernels include the Perceptron [10], ROMMA [9], ALMA [5], NORMA [7], and the Passive-Aggressive family of algorithms [2]. Each of these algorithms observes examples in a sequence of rounds and constructs its classification function incrementally by storing a subset of the observed examples in its internal memory. The classification function is then defined by a kernel-dependent combination of the stored examples. This set of stored examples is the online equivalent of the *support set* in SVMs; however, in contrast to the support, it constantly changes as learning progresses. In this paper, we call this set the *active set*, as it includes those examples that actively define the current classifier. Typically, an example is added to the active set every time the online algorithm makes a prediction mistake, or when its confidence in a prediction is inadequately low. Under certain circumstances, the active set often grows to be very big, and this can lead to significant computational difficulties. Naturally, since computing devices have bounded memory resources, there is the danger that an online algorithm would require more memory than is physically available. This problem becomes especially eminent in cases where the online algorithm is implemented as part of a specialized hardware system with a small memory, such as a mobile telephone or an autonomous robot. Moreover, the growth of the active set can lead to unacceptably long running

[*]Received by the editors August 7, 2006; accepted for publication (in revised form) June 5, 2007; published electronically January 16, 2008.
http://www.siam.org/journals/sicomp/37-5/66699.html

[†]School of Computer Science and Engineering, The Hebrew University, Jerusalem, 91904, Israel (oferd@cs.huji.ac.il, shais@cs.huji.ac.il).

[‡]Google Inc., 1600 Amphitheatre Parkway, Moutain View, CA 94043 (singer@google.com).

times, as the time-complexity of each online round scales linearly with the size of the active set.

Crammer, Kandola, and Singer [3] first addressed this problem by describing an online kernel-based modification of the Perceptron algorithm in which the active set does not exceed a predefined *budget*. Their algorithm removes redundant examples from the active set in an attempt to make the best use of the limited memory resource. Weston, Bordes, and Bottou [12] followed with their own online kernel machine on a budget. Both techniques work relatively well in practice; however, they both lack formal guarantees on prediction accuracy.

In this paper we present an online kernel-based classifier which is restricted to a fixed budget of active examples and for which we derive a formal learning-theoretic analysis. To the best of our knowledge, this is the first online algorithm on a budget for which a rigorous mistake bound has been proven. Like [3], our approach also uses the kernel-based Perceptron as a starting point and enforces the budget constraint by removing an example from the active set whenever the size of this set exceeds the predefined limit. We name our algorithm *Forgetron*, since it is a variation of the Perceptron algorithm which *forgets* active examples as necessary.

Besides forgetting active examples, the Forgetron algorithm also shrinks the online hypothesis every time it performs an update. This repeated shrinking technique is the key ingredient that makes our theoretical analysis possible. Every time a new example is added to the active set, the entire hypothesis is multiplied by a positive scalar which is at most 1, and often smaller than 1. This causes the weight of each active example to diminish from update to update. If this scaling procedure is done correctly, it ensures that there always exists an active example with a small weight and a minor influence on the current hypothesis. This example can be safely removed from the active set without causing serious damage to the accuracy of our online classifier. The scaling step should be performed carefully, since an overaggressive scaling policy could significantly impair the algorithm's prediction abilities. The delicate balance between safe removal of active examples and overaggressive scaling is the main accomplishment of this paper.

Following the preliminary presentation of the Forgetron algorithm [4], Cesa-Bianchi and Gentile devised a *randomized* online classification algorithm on a budget [1]. They also proved an upper bound on the expected number of prediction mistakes made by their algorithm. We revisit the algorithm of Cesa-Bianchi and Gentile in section 8.

This paper is organized as follows. In section 2 we begin with a more formal presentation of our problem and discuss a profound difficulty in proving mistake bounds for kernel methods on a budget. In sections 3 and 4 we lay the groundwork for our algorithm by analyzing two possible modifications to the Perceptron algorithm. In section 5 we derive the basic Forgetron algorithm, and in sections 6 and 7 we present two possible improvements to the basic algorithm. We conclude with an empirical evaluation of our algorithms in section 8 and a discussion in section 9.

2. Problem setting.

Online learning is performed in a sequence of consecutive rounds. On round t, the online algorithm observes an instance $\mathbf{x}_t$, which is drawn from some predefined instance domain $\mathcal{X}$. The algorithm predicts the binary label associated with that instance and is then given the correct label $y_t \in \{-1, +1\}$. At this point, the algorithm may use the new example $(\mathbf{x}_t, y_t)$ to improve its prediction mechanism for future rounds. We make no assumptions about the way in which the sequence of examples is generated. The goal of the algorithm is to correctly predict as many labels as possible.

The predictions of the online algorithm are determined by a function which is stored in its internal memory and is updated from round to round. We refer to this function as the *hypothesis* of the algorithm and denote the hypothesis used on round t by f_t. Our focus in this paper is on margin-based hypotheses, namely, f_t is a function from $\mathcal{X}$ to $\mathbb{R}$ where $\mathrm{sign}(f_t(\mathbf{x}_t))$ constitutes the actual binary prediction and $|f_t(\mathbf{x}_t)|$ is the confidence in this prediction. The term $yf(\mathbf{x})$ is called the *margin* of the prediction and is positive whenever y and $\mathrm{sign}(f(\mathbf{x}))$ agree. We can evaluate the performance of a hypothesis on a given example $(\mathbf{x}, y)$ in one of two ways. First, we can check whether the hypothesis makes a prediction mistake, namely, determine whether $y = \mathrm{sign}(f(\mathbf{x}))$ or not. Throughout this paper, we use M to denote the number of prediction mistakes made by an online algorithm on a sequence of examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$. The second way we can evaluate the predictions of a hypothesis is by using the *hinge-loss* function, defined as

$$\ell\big(f; (\mathbf{x}, y)\big) \;=\; \begin{cases} 0 & \text{if } \; yf(\mathbf{x}) \geq 1, \\ 1 - yf(\mathbf{x}) & \text{otherwise.} \end{cases} \tag{1}$$

The hinge-loss penalizes a hypothesis for any margin less than 1. Additionally, if $y \neq \mathrm{sign}(f(\mathbf{x}))$ then $\ell(f, (\mathbf{x}, y)) \geq 1$, and therefore the *cumulative hinge-loss* suffered over a sequence of examples upper bounds M. The algorithms discussed in this paper use kernel-based hypotheses, namely, they are defined with respect to a symmetric positive semidefinite kernel operator $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. A kernel-based hypothesis takes the form

$$f(\mathbf{x}) = \sum_{i=1}^{k} \alpha_i K(\mathbf{x}_i, \mathbf{x}), \tag{2}$$

where $\mathbf{x}_1, \ldots, \mathbf{x}_k$ are members of $\mathcal{X}$ and $\alpha_1, \ldots, \alpha_k$ are real valued weights. To facilitate the derivation of our algorithms and their analysis, we associate a reproducing kernel Hilbert space (RKHS) with K in the standard way common to all kernel-based learning methods. First, we define the inner product between the functions $f(\mathbf{x}) = \sum_{i=1}^{k} \alpha_i K(\mathbf{x}_i, \mathbf{x})$ and $g(\mathbf{x}) = \sum_{j=1}^{l} \beta_j K(\mathbf{z}_j, \mathbf{x})$ to be

$$\langle f, g \rangle = \sum_{i=1}^{k} \sum_{j=1}^{l} \alpha_i \beta_j K(\mathbf{x}_i, \mathbf{z}_j).$$

This inner product naturally induces a norm defined by $\|f\| = \langle f, f\rangle^{1/2}$ and a metric $\|f - g\| = (\langle f, f\rangle - 2\langle f, g\rangle + \langle g, g\rangle)^{1/2}$. Next, we let $\mathcal{H}_K$ denote the closure of the set of all hypotheses of the form given in (2), with respect to this metric. These definitions play an important role in the analysis of our algorithms.

Online kernel methods typically restrict themselves to hypotheses that are defined by a subset of the examples observed on previous rounds. That is, the hypothesis used on round t takes the form

$$f_t(\mathbf{x}) = \sum_{i \in I_t} \alpha_i K(\mathbf{x}_i, \mathbf{x}), \tag{3}$$

where I_t is a subset of $\{1, \ldots, (t-1)\}$ and $\mathbf{x}_i$ is the instance observed on round i. As stated above, I_t is called the active set, and we say that example $(\mathbf{x}_i, y_i)$ is *active* on round t if $i \in I_t$.

Perhaps the most well-known online algorithm for binary classification is the Perceptron [10]. Stated as a kernel method, the hypotheses generated by the Perceptron take the form $f_t(\mathbf{x}) = \sum_{i \in I_t} y_i K(\mathbf{x}_i, \mathbf{x})$. Namely, the weight assigned to each active example is either $+1$ or -1, depending on the label of that example. The Perceptron initializes I_1 to be the empty set, which implicitly sets f_1 to be the zero function. It then updates its hypothesis only on rounds where a prediction mistake is made. Concretely, if on round t the margin $y_t f_t(\mathbf{x}_t)$ is nonpositive, then the index t is inserted into the active set. As a consequence, the size of the active set on round t equals the number of prediction mistakes made on previous rounds. A relative mistake bound can be proven for the Perceptron algorithm. The bound holds for any sequence of examples and compares the number of mistakes made by the Perceptron with the cumulative hinge-loss of any fixed hypothesis $g \in \mathcal{H}_K$, even one defined with prior knowledge of the sequence.

THEOREM 1. *Let K be a kernel and let $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_T, y_T)$ be a sequence of examples such that $K(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ for all t. Let g be an arbitrary function in $\mathcal{H}_K$ and define $\ell_t^\star = \ell\big(g; (\mathbf{x}_t, y_t)\big)$. Then the number of prediction mistakes made by the Perceptron on this sequence is bounded by*

$$M \;\leq\; \|g\|^2 + 2 \sum_{t=1}^{T} \ell_t^\star .$$

The proof of this theorem is given in the next section and serves as the basis of the analysis in this paper. Although the Perceptron is guaranteed to be competitive with any fixed hypothesis $g \in \mathcal{H}_K$, the fact that its active set grows with every mistake may pose a serious computational problem, as already noted in the introduction. In fact, this problem is common to most kernel-based online methods which do not explicitly monitor the size of I_t.

On the limitation of algorithms on a memory budget. Our goal is to derive and analyze an online prediction algorithm which resolves the problems discussed above by enforcing a *fixed* bound on the size of the active set. Formally, let B be a positive integer which we refer to as the *budget parameter*. We would like to devise an algorithm which enforces the constraint $|I_t| \leq B$ on every round t. Furthermore, we would like to prove a relative mistake bound for this algorithm along the lines of Theorem 1. Regretfully, it turns out that this goal cannot be reached without making additional assumptions. We show this inherent limitation by presenting a simple counterexample. That is, for any kernel-based algorithm that uses a prediction function of the form given in (3) and which adheres to the constraint $|I_t| \leq B$, we can find a kernel K, a hypothesis $g \in \mathcal{H}_K$, and an arbitrarily long sequence of examples such that the algorithm makes a prediction mistake on every single round while g suffers no loss at all. Our counterexample is constructed as follows. We choose $\mathcal{X}$ to be the set of $B+1$ standard unit vectors in $\mathbb{R}^{B+1}$, namely, $\mathcal{X} = \{e_i\}_{i=1}^{B+1}$, where e_i is the vector with 1 in its ith coordinate and zeros elsewhere. The kernel function K is set to be the standard dot product in $\mathbb{R}^{B+1}$; thus $K(\mathbf{x}, \mathbf{x}') = \mathbf{x} \cdot \mathbf{x}'$. On every round t, the online hypothesis, f_t, is a linear combination of at most B vectors from $\mathcal{X}$. Since $|\mathcal{X}| = B + 1$, there exists a vector $\mathbf{x}_t \in \mathcal{X}$ which is not currently active. Furthermore, by construction, $\mathbf{x}_t$ is orthogonal to all of the active vectors, and therefore $f_t(\mathbf{x}_t) = 0$. Assume without loss of generality that the online algorithm we are using predicts y_t to be -1 when $f_t(\mathbf{x}) = 0$. If on every round we were to present the online algorithm with the example $(\mathbf{x}_t, +1)$, then the online algorithm would make a prediction mistake on every round. On the other hand, the hypothesis $\bar{g} = \sum_{i=1}^{B+1} e_i$ is a member of H_K and attains a hinge-loss of 0 on every round. We have found a sequence of examples and a fixed hypothesis (which is indeed defined by more than B vectors from $\mathcal{X}$) that attains a cumulative loss of zero on this sequence, while the number of mistakes made by our online algorithm equals the number of rounds. Clearly, a general theorem along the lines of Theorem 1 cannot be proven.

One way to resolve the problem illustrated above is to limit the set of competing hypotheses to a subset of $\mathcal{H}_K$ in a way that would naturally exclude $\bar{g}$ in the example above. In this paper, we limit the set of competitors to hypotheses with a bounded norm. Formally, we wish to devise an online algorithm which is competitive with every hypothesis $g \in \mathcal{H}_K$ for which $\|g\| \leq U$, where U is a predefined positive constant. The counterexample above indicates that we cannot prove a relative mistake bound with $U \geq \sqrt{B+1}$, since the norm of $\bar{g}$ in our counterexample is $\sqrt{B+1}$. In this paper we come close to this upper bound by proving that our algorithms can compete with any hypothesis whose norm is bounded from above by $\frac{1}{4}\sqrt{(B+1)/\log(B+1)}$. Limiting the set of competing hypotheses to a ball of norm U about the origin of $\mathcal{H}_K$ is one possible way to overcome the problem exposed by our counterexample. It seems plausible that other restrictions on the general problem setting, such as specific choices of the instance domain $\mathcal{X}$ or the kernel function K, could resolve this problem equally well.

As mentioned in the previous section, Cesa-Bianchi and Gentile devised a randomized online classification algorithm on a budget [1]. Their analysis shows that their algorithm is competitive with any hypothesis whose norm is bounded from above by $O(\sqrt{B+1})$. However, in contrast to our analysis, which bounds the actual number of prediction mistakes (M), the analysis of Cesa-Bianchi and Gentile bounds the *expected* number of mistakes ($\mathbb{E}[M]$), where expectation is taken over the internal randomization of their algorithm. Namely, the actual performance of their randomized algorithm varies from run to run. We illustrate this phenomenon empirically in section 8.

3. The Remove-Oldest Perceptron.

The Perceptron algorithm and its mistake bound (Theorem 1) serve as our starting point. Therefore, it is important to understand the proof of Theorem 1 before proceeding. The key to proving Theorem 1 is the observation that the hypothesis of the Perceptron is drawn toward good hypotheses in $\mathcal{H}_K$. Specifically, whenever the Perceptron makes a prediction mistake, its hypothesis moves closer to every hypothesis $g \in \mathcal{H}_K$ which attains a margin of at least $\frac{1}{2}$ on the current example. This fact is formally stated and proven below.

Lemma 1. *Let $(\mathbf{x}, y)$ be an example, where $\mathbf{x} \in \mathcal{X}$, $K(\mathbf{x},\mathbf{x}) \leq 1$, and $y \in \{-1,+1\}$. Let $f \in \mathcal{H}_K$ be a function such that $yf(\mathbf{x}) \leq 0$, and define $f' = f + yK(\mathbf{x},\cdot)$. Then for any function $g \in \mathcal{H}_K$ it holds that*

$$\|f-g\|^2 - \|f'-g\|^2 \;\geq\; 2yg(\mathbf{x}) - 1.$$

Proof. Using the definition of f', we can write

$$\begin{aligned}
&\|f-g\|^2 - \|f'-g\|^2 \\
&= \|f-g\|^2 - \|(f-g) + yK(\mathbf{x},\cdot)\|^2 \\
&= \|f-g\|^2 - \|f-g\|^2 - 2y\langle (f-g), K(\mathbf{x},\cdot)\rangle - K(\mathbf{x},\mathbf{x}) \\
&= -2y\langle f, K(\mathbf{x},\cdot)\rangle + 2y\langle g, K(\mathbf{x},\cdot)\rangle - K(\mathbf{x},\mathbf{x}).
\end{aligned}$$

Using the reproducing property of $\mathcal{H}_K$, we know that $\langle f, K(\mathbf{x}, \cdot)\rangle = f(\mathbf{x})$ and that $\langle g, K(\mathbf{x}, \cdot)\rangle = g(\mathbf{x})$, and thus we get

$$\|f - g\|^2 - \|f' - g\|^2 \ = \ - 2yf(\mathbf{x}) + 2yg(\mathbf{x}) - K(\mathbf{x}, \mathbf{x}). \tag{4}$$

Using our assumption that $yf(\mathbf{x}) \leq 0$, it follows that $-2yf(\mathbf{x}) \geq 0$. Additionally, recall that we made the assumption that $K(\mathbf{x}, \mathbf{x}) \leq 1$. Plugging these facts back into (4) gives

$$\|f - g\|^2 - \|f' - g\|^2 \ \geq \ 2yg(\mathbf{x}) - 1. \tag{5}$$

This concludes the proof. □

The term $\|f_t - g\|^2 - \|f_{t+1} - g\|^2$ measures how much the hypothesis of the Perceptron gets closer to g, as a result of the update on round t. This term plays an important role in our paper and we therefore denote it by Δ_t. It is worth noting that Δ_t also plays an important role in the analysis of other online algorithms [8, 6, 2]. The proof of Theorem 1 is a simple corollary of Lemma 1.

Proof of Theorem 1. We prove the theorem by bounding $\sum_{t=1}^{T} \Delta_t$ from above and from below. First note that $\sum_t \Delta_t$ is a telescopic sum which reduces to

$$\sum_{t=1}^{T} \Delta_t \ = \ \|f_1 - g\|^2 - \|f_{T+1} - g\|^2.$$

Using the facts that $\|f_{T+1} - g\|^2 \geq 0$ and that f_1 is the zero function, we can upper bound $\sum_t \Delta_t$ by $\|g\|^2$. Next we show a lower bound on $\sum_t \Delta_t$. For rounds on which the Perceptron makes a correct prediction, we have that $f_{t+1} = f_t$, and thus $\Delta_t = 0$. For rounds on which the Perceptron makes a mistake, Lemma 1 tells us that $\Delta_t \geq 2y_t g(\mathbf{x}_t) - 1$. The definition of the hinge-loss in (1) implies that $\ell_t^\star \geq 1 - y_t g(\mathbf{x}_t)$ and therefore $2y_t g(\mathbf{x}_t) \geq 2 - 2\ell_t^\star$. Therefore, we have that

$$\|f_t - g\|^2 - \|f_{t+1} - g\|^2 \geq \ 1 - 2\ell_t^\star.$$

Recalling that M denotes the total number of prediction mistakes made on the entire sequence of examples, we obtain that

$$\sum_{t=1}^{T} \Delta_t \ \geq \ M - 2 \sum_{t: y_t f_t(\mathbf{x}_t) \leq 0} \ell_t^\star.$$

Since the hinge-loss is nonnegative, it holds that

$$\sum_{t=1}^{T} \Delta_t \ \geq \ M - 2 \sum_{t=1}^{T} \ell_t^\star. \tag{6}$$

Comparing this lower bound with the upper bound $\sum_t \Delta_t \leq \|g\|^2$ and rearranging terms proves the theorem. □

We now present the *Remove-Oldest* Perceptron, a simple modification of the kernel Perceptron which conforms with a fixed budget constraint. As long as the active set is smaller than the budget parameter B, the Remove-Oldest Perceptron behaves exactly like the standard kernel Perceptron. The active set therefore grows with every mistake and eventually contains B examples. Once the active set reaches

B examples, the online update is performed in two steps. Whenever the algorithm makes a mistake, it first adds an example to the active set by performing the standard Perceptron update, and then it reduces the size of the active set back to B by removing the oldest active example. More formally, for all $1 \le t \le T$, let I'_t define the active set obtained on round t after applying the standard Perceptron update. That is,

$$I'_t \;=\; \begin{cases} I_t & \text{if } y_t f_t(\mathbf{x}_t) > 0, \\ I_t \cup \{t\} & \text{if } y_t f_t(\mathbf{x}_t) \le 0. \end{cases} \tag{7}$$

Also, let f'_t denote the hypothesis defined by I'_t, namely,

$$f'_t \;=\; \sum_{t \in I'_t} y_t K(\mathbf{x}_t, \cdot). \tag{8}$$

Now, define I_{t+1} to be

$$I_{t+1} \;=\; \begin{cases} I'_t \setminus \{r_t\} & \text{if } |I'_t| = B+1, \\ I'_t & \text{if } |I'_t| \le B, \end{cases} \tag{9}$$

where $r_t = \min I'_t$. Besides being an interesting algorithm, the Remove-Oldest Perceptron is an important intermediate step toward the Forgetron algorithm.

We are unable to prove a mistake bound for the Remove-Oldest Perceptron; however, we are able to quantify the damage due to the second step of the update, defined in (9). Assume that the algorithm is run for T rounds. Let J denote the set of rounds on which a prediction mistake is made, namely, $J = \{1 \le t \le T : y_t f_t(\mathbf{x}_t) \le 0\}$ and $M = |J|$. Note that I_{T+1}, the active set at the end of T rounds, is a subset of J. To analyze the Remove-Oldest Perceptron, we again assume that g is an arbitrary function in $\mathcal{H}_K$ and define $\Delta_t = \|f_t - g\|^2 - \|f_{t+1} - g\|^2$. As in the proof of Theorem 1, the sum $\sum_{t=1}^{T} \Delta_t$ can be upper bounded by $\|g\|^2$, and we concentrate on bounding it from below. Using the notation f'_t, defined in (8), we can rewrite Δ_t as follows:

$$\Delta_t = \|f_t - g\|^2 - \|f'_t - g\|^2 + \|f'_t - g\|^2 - \|f_{t+1} - g\|^2.$$

For brevity, let us define

$$\alpha_t = \|f_t - g\|^2 - \|f'_t - g\|^2 \qquad \text{and} \qquad \gamma_t = \|f'_t - g\|^2 - \|f_{t+1} - g\|^2,$$

and thus $\sum_t \Delta_t = \sum_t \alpha_t + \sum_t \gamma_t$. Since $\Delta_t \ne 0$ only for $t \in J$, we can rewrite

$$\sum_{t=1}^{T} \Delta_t \;=\; \sum_{t \in J} (\alpha_t + \gamma_t) \;=\; \sum_{t \in I_{T+1}} \alpha_t \;+\; \sum_{t \in J \setminus I_{T+1}} \alpha_t \;+\; \sum_{t \in J} \gamma_t. \tag{10}$$

The summands in $\sum_{t \in J} \gamma_t$ which are equal to zero can be omitted from the sum. Specifically, note that γ_t is nonzero only on rounds for which $|I_t| = B$, and therefore,

$$\sum_{t \in J} \gamma_t \;=\; \sum_{t \in J \,:\, |I_t| = B} \gamma_t. \tag{11}$$

The set $J \setminus I_{T+1}$ consists of the indices of the examples that were inserted into the active set and later removed from it. Another way to write this set, using the notation r_t defined above, is $\{r_t \;:\; t \in J \;\wedge\; |I_t| = B\}$, since active examples are removed precisely on rounds on which a mistake occurs and the active set is full. Therefore, it holds that

$$\sum_{t \in J \setminus I_{T+1}} \alpha_t \;=\; \sum_{t \in J\,:\,|I_t|=B} \alpha_{r_t}. \tag{12}$$

Using (11) and (12), we can rewrite (10) as

$$\sum_{t=1}^{T} \Delta_t \;=\; \sum_{t \in I_{T+1}} \alpha_t \;+\; \sum_{t \in J\,:\,|I_t|=B} (\alpha_{r_t} + \gamma_t). \tag{13}$$

We have rewritten $\sum_t \Delta_t$ as the sum of two terms. Next, we lower bound each term individually. The first term deals with examples that were added to the active set and never removed. For these examples, only the effect of the standard Perceptron update (α_t) must be taken into account. The second term deals with examples that were added and then later removed from the active set. Decomposing $\sum \Delta_t$ in this way, and dealing with the two terms separately, is an important technique which we reuse in our main formal result, namely, the proof of Theorem 3.

We first consider the first term on the right-hand side of (13). For every $t \in I_{T+1}$ we can use Lemma 1 to bound $\alpha_t \geq 2y_t g(\mathbf{x}_t) - 1$. Using the definition of the hinge-loss in (1), we know that $2y_t g(\mathbf{x}_t) - 1 \geq 1 - 2\ell_t^\star$, and therefore,

$$\alpha_t \;\geq\; 1 - 2\ell_t^\star. \tag{14}$$

Moving on to the second term on the right-hand side of (13), we note that for every round t on which an example was removed from the active set, α_{r_t} measures the benefit of initially adding the example r_t to the active set, whereas γ_t measures the damage caused by removing this example later. We will actually analyze a more general case where instead of entirely removing example r_t on round t, we may only decrease its weight. In other words, instead of subtracting $y_{r_t} K(\mathbf{x}_{r_t}, \cdot)$ from the current hypothesis, we subtract $\lambda y_{r_t} K(\mathbf{x}_{r_t}, \cdot)$ for some $0 < \lambda \leq 1$. For the purpose of lower bounding (13), we can simply assume that $\lambda = 1$. However, the more general form of our analysis will prove useful later, as we make further progress toward an algorithm with a budget constraint and a mistake bound.

Next, we show that our lower bound on $\alpha_{r_t} + \gamma_t$ is influenced by two factors: the parameter λ, which determines what portion of the example $\mathbf{x}_{r_t}$ is removed, and the term $y_{r_t} f_t'(\mathbf{x}_{r_t})$, which is the margin attained by the current hypothesis on the example being removed. More precisely, we show that the lower bound on $\alpha_{r_t} + \gamma_t$ is similar to the lower bound in (14) minus the additional penalty

$$\Psi(\lambda, \mu) \;=\; \lambda^2 + 2\lambda - 2\lambda\mu, \tag{15}$$

where μ is an abbreviation for $y_{r_t} f_t'(\mathbf{x}_{r_t})$.

Lemma 2. *Let f, f', and g be arbitrary functions in $\mathcal{H}_K$, let $(\mathbf{x}, y)$ be an example such that $\mathbf{x} \in \mathcal{X}$, $K(\mathbf{x}, \mathbf{x}) \leq 1$, and $y \in \{-1, +1\}$, and define $\ell^\star = \ell(g; (\mathbf{x}, y))$. Assume that $yf(\mathbf{x}) \leq 0$. Then for any $\lambda \in (0, 1]$ it holds that*

$$\Big(\|f - g\|^2 - \|(f + yK(\mathbf{x}, \cdot)) - g\|^2\Big) + \Big(\|f' - g\|^2 - \|(f' - \lambda y K(\mathbf{x}, \cdot)) - g\|^2\Big)$$
$$\geq\; 1 - 2\ell^\star - \Psi(\lambda,\, yf'(\mathbf{x})).$$

Proof. We rewrite $\|f' - g\|^2 - \|(f' - \lambda y K(\mathbf{x}, \cdot)) - g\|^2$ as

$$\begin{aligned}
&\|f' - g\|^2 - \|f' - \lambda y K(\mathbf{x}, \cdot) - g\|^2 \\
&\quad = \|f' - g\|^2 - \|f' - g\|^2 + 2\lambda y \langle f' - g, K(\mathbf{x}, \cdot)\rangle - \lambda^2 \|K(\mathbf{x}, \cdot)\|^2 \\
&\quad = 2\lambda y \langle f', K(\mathbf{x}, \cdot)\rangle - 2\lambda y \langle g, K(\mathbf{x}, \cdot)\rangle - \lambda^2 \|K(\mathbf{x}, \cdot)\|^2.
\end{aligned}$$

Using the reproducing property of $\mathcal{H}_K$, it holds that $\langle f', K(\mathbf{x}, \cdot)\rangle = f'(\mathbf{x})$, $\langle g, K(\mathbf{x}, \cdot)\rangle = g(\mathbf{x})$, and $\|K(\mathbf{x}, \cdot)\|^2 = K(\mathbf{x}, \mathbf{x})$. Plugging these equalities into the above, and using our assumption that $K(\mathbf{x}, \mathbf{x}) \leq 1$, we have

$$\|f' - g\|^2 - \|f' - \lambda y K(\mathbf{x}, \cdot) - g\|^2 \;\geq\; 2\lambda y f'(\mathbf{x}) - 2\lambda y g(\mathbf{x}) - \lambda^2. \tag{16}$$

Using Lemma 1 and denoting $f' = f + yK(\mathbf{x}, \cdot)$ we get the bound

$$\|f - g\|^2 - \|f' - g\|^2 \;\geq\; 2yg(\mathbf{x}) - 1. \tag{17}$$

For brevity, let us denote the term on the left-hand side of the statement of the lemma by δ. Summing (17) with (16), we have

$$\delta \;\geq\; 1 - 2(1-\lambda)(1 - yg(\mathbf{x})) - \Big(2\lambda + \lambda^2 - 2\lambda y f'(\mathbf{x})\Big).$$

Using the definition of the hinge-loss, it holds that $\ell^\star \geq 1 - yg(\mathbf{x})$, and since $(1-\lambda) \geq 0$, we get

$$\delta \;\geq\; 1 - 2(1-\lambda)\ell^\star - \Big(2\lambda + \lambda^2 - 2\lambda y f'(\mathbf{x})\Big).$$

Finally, we neglect the nonnegative term $2\lambda\ell^\star$, and the lemma is proven. □

Using Lemma 2 with λ set to 1 and f' set to f'_t, we get

$$\alpha_{r_t} + \gamma_t \;\geq\; 1 - 2\ell^\star_{r_t} - \Psi(1,\, y_{r_t} f'_t(\mathbf{x}_{r_t})). \tag{18}$$

Combining (13) with (14) and (18), we obtain the lower bound

$$\sum_{t=1}^{T} \Delta_t \;\geq\; M - \; 2\sum_{t \in J} \ell^\star_t \; - \sum_{t \in J \,:\, |I_t| = B} \Psi(1,\, y_{r_t} f'_t(\mathbf{x}_{r_t})).$$

Comparing this bound to the upper bound $\sum_t \Delta_t \leq \|g\|^2$, using the fact that the hinge loss is always nonnegative, yields the following corollary.

COROLLARY 1. *Let K be a symmetric positive semidefinite kernel and let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of examples such that $K(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ for all t. Let g be an arbitrary function in $\mathcal{H}_K$, define $\ell^\star_t = \ell\big(g; (\mathbf{x}_t, y_t)\big)$, and let Ψ be as defined in* (15)*. Then the number of prediction mistakes made by the Remove-Oldest Perceptron on this sequence is bounded by*

$$M \;\leq\; \|g\|^2 + 2\sum_{t=1}^{T} \ell^\star_t \; + \sum_{t \in J \,:\, |I_t| = B} \Psi(1,\, y_{r_t} f'_t(\mathbf{x}_{r_t})).$$

This corollary does not constitute a relative mistake bound since we cannot provide any guarantee of the value of $\Psi(1,\, y_{r_t} f'_t(\mathbf{x}_{r_t}))$. The magnitude of this term depends on how well the classifier f'_t classifies the example being removed, $\mathbf{x}_{r_t}$. Referring back to the definition of Ψ in (15), we get that $\Psi(1,\, y_{r_t} f'_t(\mathbf{x}_{r_t})) = 3 - 2y_{r_t} f'_t(\mathbf{x}_{r_t})$. Therefore, every time $y_{r_t} f'_t(\mathbf{x}_{r_t}) \geq \frac{3}{2}$, the bound in Corollary 1 is actually strengthened, whereas every time $y_{r_t} f'_t(\mathbf{x}_{r_t}) < \frac{3}{2}$, it is weakened. Clearly, the term $y_{r_t} f'_t(\mathbf{x}_{r_t})$ plays an important role in determining whether or not $\mathbf{x}_{r_t}$ can be safely removed from the active set on round t. In order to obtain a concrete mistake bound, we must modify the Remove-Oldest Perceptron in a way which controls the damage caused by the removal step. Lemma 2, in its general form $(0 < \lambda \leq 1)$, helps us gain this control. Namely, we can control the magnitude of the term $\Psi(\lambda,\, y_{r_t} f'_t(\mathbf{x}_{r_t}))$ by ensuring that $|f'_{r_t}(\mathbf{x}_t)|$ is sufficiently small, and by setting λ to a value smaller than 1. Both tasks can be achieved by repeatedly shrinking the online hypothesis on every update. The details of this modification are discussed in the next section.

4. Repeatedly shrinking the Perceptron hypothesis. In the previous section, we discussed the damage caused by removing the oldest active example from the active set. The key to controlling the extent of this damage is to ensure that the example being removed has a sufficiently small influence on the current hypothesis. One way to achieve this is by shrinking the norm of the online hypothesis following each update. Namely, on each round t where an update is performed, the online hypothesis is multiplied by a scalar $0 < \phi_t \leq 1$ (the concrete value of ϕ_t is specified in the next section). To study the effect of the shrinking step on the accuracy of the online algorithm, let us momentarily forget about the removal step introduced in the previous section and focus only on the shrinking step. The two techniques, removal and shrinking, are combined in the next section.

To facilitate the analysis of the shrinking technique, we introduce a new online algorithm, the *Shrinking* Perceptron. This algorithm is a variation of the standard kernel-based Perceptron and constructs an online hypothesis which is a *weighted* combination of functions in $\mathcal{H}_K$,

$$f_t \;=\; \sum_{i \in I_t} y_i \sigma_{i,t} K(\mathbf{x}_i, \cdot),$$

where $\sigma_{i,t} \in [0,1]$. The update procedure starts with the standard Perceptron update. Specifically, if a correct prediction is made then $f_{t+1} = f_t$. Otherwise, I_{t+1} is set to $I_t \cup \{t\}$ and $\sigma_{t,t}$ is set to 1. We use the notation f'_t to denote the intermediate hypothesis which results from this update, namely,

$$f'_t(\mathbf{x}) \;=\; f_t(\mathbf{x}) + y_t \sigma_{t,t} K(\mathbf{x}_t, \mathbf{x}). \tag{19}$$

The second step of the update is the shrinking step, which sets f_{t+1} to be $\phi_t f'_t$, where ϕ_t is a shrinking coefficient in $(0,1]$. Setting $\sigma_{i,t+1} = \phi_t \sigma_{i,t}$ for all $1 \leq i \leq t$, we can write

$$f_{t+1} \;=\; \sum_{i \in I_{t+1}} y_i \sigma_{i,t+1} K(\mathbf{x}_i, \cdot).$$

The recursive definition of each weight $\sigma_{i,t}$ can be unraveled to give the following explicit form:

$$\sigma_{i,t} \;=\; \prod_{j \in I_{t-1} \,\wedge\, j \geq i} \phi_j .$$

By choosing sufficiently small shrinking coefficients ϕ_t, we can make the weights $\sigma_{i,t}$ decrease as quickly as we like. If these weights indeed decrease rapidly enough, the contribution of older active examples to the online hypothesis becomes negligible. This demonstrates the potential of shrinking as a means of controlling the effect of old active examples on the current hypothesis. However, this benefit comes at a price. Repeatedly shrinking the norm of the Perceptron hypothesis takes a toll on the accuracy of the online algorithm. A good choice of ϕ_t should balance the need to attenuate the influence of older active examples with the damage caused by the shrinking step. In the remainder of this section, we prove a bound on the damage caused by the shrinking step.

To remind the reader, our goal, as stated in section 2, is to find an algorithm which is competitive with any $g \in \mathcal{H}_K$ whose norm $\|g\|$ is bounded above by U, where $U = \frac{1}{4}\sqrt{(B+1)/\log(B+1)}$. The term $\Delta_t = \|f_t - g\|^2 - \|f_{t+1} - g\|^2$, which played a major role in the proof of Theorem 1, again appears in our analysis. We now show how this term is affected by the shrinking step. As before, $\Delta_t = 0$ on rounds where a correct prediction was made, and we can focus on rounds where $\Delta_t \neq 0$. As before, we denote the set of indices t for which $\Delta_t > 0$ by J. Using the notation f'_t defined above, we can rewrite Δ_t as

$$\Delta_t = \|f_t - g\|^2 - \|f'_t - g\|^2 + \|f'_t - g\|^2 - \|f_{t+1} - g\|^2.$$

For brevity, define

$$\alpha_t = \|f_t - g\|^2 - \|f'_t - g\|^2 \qquad \text{and} \qquad \beta_t = \|f'_t - g\|^2 - \|f_{t+1} - g\|^2, \tag{20}$$

and so $\sum_t \Delta_t = \sum_{t \in J} \alpha_t + \sum_{t \in J} \beta_t$. For each t, α_t measures the progress made by the Perceptron update on round t, while β_t measures the damage caused by the shrinking step which follows the Perceptron update. Our first task is to lower bound $\sum_{t \in J} \beta_t$. In order to do so, we partition the set J into the following three subsets:

$$\begin{aligned} J_1 &= \{t \in J \ : \ \phi_t \, \|f'_t\| \geq U\}, \\ J_2 &= \{t \in J \ : \ \|f'_t\| \leq U \ \wedge \ \phi_t \, \|f'_t\| < U\}, \\ J_3 &= \{t \in J \ : \ \|f'_t\| > U \ \wedge \ \phi_t \, \|f'_t\| < U\}. \end{aligned} \tag{21}$$

To gain some insight, we can think of the shrinking step in geometric terms. On round t, we first apply the Perceptron update and obtain f'_t. The function f'_t is a point in the Hilbert space $\mathcal{H}_K$. Then, we perform the shrinking step which moves f'_t toward the origin of $\mathcal{H}_K$, resulting in f_{t+1}. Now let $\mathcal{B}_U \subset \mathcal{H}_K$ be a ball of radius U, centered at the origin of $\mathcal{H}_K$. The set J_1 represents those rounds where both f'_t and f_{t+1} lie outside or on the surface of $\mathcal{B}_U$. The set J_2 represents the rounds where $f'_t \in \mathcal{B}_U$ and f_{t+1} lies in the interior of $\mathcal{B}_U$. Finally, J_3 represents rounds where $f'_t \notin \mathcal{B}_U$ and the shrinking step moves f_{t+1} into the interior of $\mathcal{B}_U$. This geometric interpretation is illustrated in Figure 1. We now deal with each of the three cases individually, beginning with the set J_1. The following lemma builds on our assumption that $\|g\| \leq U$.

LEMMA 3. *Let $U > 0$ and $0 < \phi \leq 1$ be scalars, and let g and f be two functions in $\mathcal{H}_K$ such that $\|g\| \leq U \leq \phi\|f\|$. Then,*

$$\|f - g\|^2 - \|\phi f - g\|^2 \geq 0.$$

Proof. We begin by noting that $\phi\|f\|^2 \geq U\|f\| \geq \|g\|\|f\|$. Using the Cauchy–Schwarz inequality, we have that $\|g\|\|f\| \geq \langle f, g\rangle$, and therefore

$$\phi\|f\|^2 \ \geq \ \langle f, g\rangle. \tag{22}$$

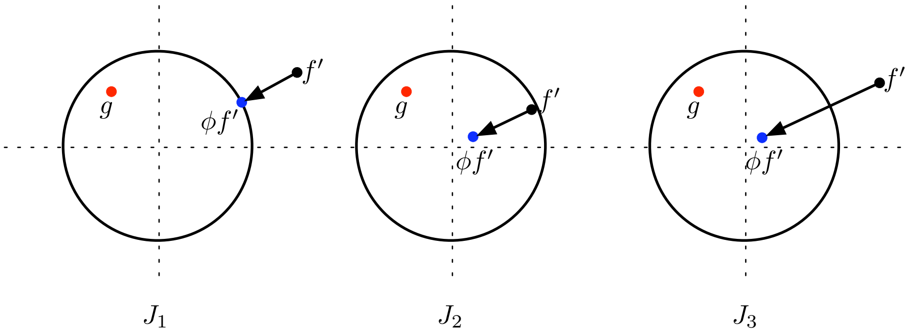


FIG. 1. *A geometrical interpretation of the three hypothesis-shrinking cases.*

The term $\|f-g\|^2 - \|\phi f - g\|^2$ can now be rewritten as

$$\|f-g\|^2 - \|\phi f - g\|^2 = \left(\|f\|^2 - 2\langle f,g\rangle + \|g\|^2\right) - \left(\phi^2\|f\|^2 - 2\phi\langle f,g\rangle + \|g\|^2\right)$$
$$= (1-\phi^2)\|f\|^2 - 2(1-\phi)\langle f,g\rangle. \tag{23}$$

Since $(1-\phi)$ is nonnegative, we can plug (22) into the right-hand side above and get the bound

$$\|f-g\|^2 - \|\phi f - g\|^2 \;\geq\; (1-\phi^2)\|f\|^2 - 2(1-\phi)\phi\|f\|^2 \;=\; (1-\phi)^2\,\|f\|^2 \;\geq\; 0. \qquad \square$$

To recap, the geometric implication of Lemma 3 is that the shrinking step does not have an adverse effect on Δ_t so long as f_{t+1} remains outside the interior of $\mathcal{B}_U$.

Next, we prove a looser bound, compared to the bound provided by Lemma 3, which holds for all $t \in J$ and in particular for $t \in J_2$.

LEMMA 4. *Let g and f be two functions in $\mathcal{H}_K$. Then, for any ϕ in $(0,1]$ the following bound holds:*

$$\|f-g\|^2 - \|\phi f - g\|^2 \geq \|g\|^2(\phi - 1).$$

Proof. As in (23), the left-hand side in the statement of the lemma can be rewritten as

$$\|f-g\|^2 - \|\phi f - g\|^2 \;=\; (1-\phi^2)\|f\|^2 - 2(1-\phi)\langle f,g\rangle.$$

We now use the elementary fact that for any $u, v \in \mathcal{H}_K$, $\|u-v\|^2 \geq 0$, which can be rewritten as $\|u\|^2 - 2\langle u,v\rangle \geq -\|v\|^2$. Setting $u = \sqrt{1-\phi^2}\, f$ and $v = \sqrt{\frac{1-\phi}{1+\phi}}\, g$, this inequality becomes

$$(1-\phi^2)\|f\|^2 - 2(1-\phi)\langle f,g\rangle \;\geq\; -\frac{1-\phi}{1+\phi}\|g\|^2.$$

Combining the above inequality with the fact that $1+\phi \geq 1$ proves the bound. $\square$

Finally, we focus on rounds from J_3.

LEMMA 5. *Let $U > 0$ and $0 < \phi \leq 1$ be scalars, and let g and f be two functions in $\mathcal{H}_K$ such that $\|g\| \leq U$, $\|f\| > U$, and $\|\phi f\| < U$. Then,*

$$\|f-g\|^2 - \|\phi f - g\|^2 \;\geq\; \|g\|^2\left(\frac{\phi\|f\|}{U} - 1\right).$$

Proof. Defining $\nu = U/\|f\|$, we can rewrite the left-hand side of our claim as

$$\left(\|f-g\|^2 - \|\nu f - g\|^2\right) + \left(\|\nu f - g\|^2 - \|\phi f - g\|^2\right).$$

Since $0 < \nu < 1$ and $\|\nu f\| = U$, we can use Lemma 3 to lower bound the first term above by 0. Similarly, we can use Lemma 4 to lower bound the second term above by $-\|g\|^2(1-\frac{\phi}{\nu})$. Summing the two bounds, we get

$$\|f-g\|^2 - \|\phi f - g\|^2 \;\geq\; -\|g\|^2\left(1 - \frac{\phi}{\nu}\right) \;=\; \|g\|^2\left(\frac{\phi\|f\|}{U} - 1\right),$$

which proves the lemma. □

Combining Lemmas 3, 4, and 5, and recalling that $\beta_t = \|f'_t - g\|^2 - \|f_{t+1} - g\|^2$, we obtain the following lower bound:

$$\sum_{t\in J} \beta_t \;\geq \|g\|^2 \left(\sum_{t \in J_2} (\phi_t - 1) \;+\; \sum_{t\in J_3}\Big(\frac{\phi_t\|f'_t\|}{U} - 1\Big)\right).$$

This bound can be restated as follows:

$$(24) \qquad \sum_{t\in J}\beta_t \;\geq \|g\|^2 \sum_{t\in J}(\Phi_t - 1), \quad \text{where} \quad \Phi_t \;=\; \begin{cases} 1 & \text{if } t \in J_1, \\ \phi_t & \text{if } t \in J_2, \\ \frac{\phi_t \|f'_t\|}{U} & \text{if } t \in J_3. \end{cases}$$

Using the inequality $x - 1 \geq \log(x)$, we obtain the following corollary.

Corollary 2. *Let g be a function in $\mathcal{H}_K$ such that $\|g\| \leq U$, where $U \geq 0$. Let β_t be as defined in* (20) *and Φ_t be as defined in* (24)*. Then it holds that*

$$\sum_{t\in J}\beta_t \;\geq\; \|g\|^2 \log\left(\prod_{t\in J}\Phi_t\right).$$

Repeating the analysis of the kernel Perceptron, we can lower bound $\sum_{t\in J}\alpha_t \geq M - 2\sum_{t=1}^{T}\ell^\star_t$ (as in (6)) and upper bound $\sum_{t\in J}(\alpha_t + \beta_t) \leq \|g\|^2$. Combining these two inequalities with the result from Corollary 2 gives

$$M \;\leq\; \|g\|^2\left(1 - \log\left(\prod_{t\in J}\Phi_t\right)\right) + 2\sum_{t=1}^{T}\ell^\star_t.$$

The above mistake bound can be applied to any concrete strategy of choosing the shrinking coefficient ϕ_t. In the next section, we combine elements from the analysis of the Remove-Oldest Perceptron and the Shrinking Perceptron to derive the Forgetron algorithm, our first online algorithm on a budget for which we prove a mistake bound.

5. The Forgetron algorithm.

5. The Forgetron algorithm. In this section we present the Forgetron algorithm, which combines the removal and shrinking techniques presented in the previous sections. The result is a provably correct online learning algorithm on a fixed budget. The main challenge in combining the two techniques revolves around the choice of the shrinking coefficients $\phi_1, \ldots, \phi_T$. On one hand, the shrinking step must be aggressive enough to attenuate the contribution of old active examples to the online hypothesis. On the other hand, an overly aggressive shrinking policy could damage the accuracy of our algorithm. Concretely, we show that the following choice of ϕ_t successfully balances this tradeoff:

$$\phi_t \;=\; \min\left\{ (B+1)^{-\frac{1}{2(B+1)}} \,,\, \frac{U}{\|f'_t\|} \right\}, \tag{25}$$

where $f'_t = f_t + y_t K(\mathbf{x}_t, \cdot)$ and

$$U \;=\; \frac{1}{4}\sqrt{\frac{B+1}{\log(B+1)}}. \tag{26}$$

Although this simple choice of ϕ_t enables us to prove a formal mistake bound, we note that it has some deficiencies, which we discuss at the end of this section. In the next section, we describe a refined mechanism for choosing ϕ_t, which overcomes these deficiencies.

The Forgetron algorithm initializes I_1 to be the empty set, which implicitly sets f_1 to be the zero function. If a prediction mistake occurs on round t, namely, $y_t f_t(\mathbf{x}_t) \leq 0$, a three step update is performed. The first step is the Perceptron update, which inserts the index t into the active set. We denote the resulting active set by I'_t and the resulting hypothesis by

$$f'_t \;=\; f_t(\mathbf{x}) + y_t K(\mathbf{x}_t, \cdot). \tag{27}$$

The second step is the shrinking step, which sets

$$f''_t \;=\; \phi_t f'_t, \tag{28}$$

where $\phi_t \in (0, 1]$ is the shrinking coefficient. The last step of the update is the removal step: if the budget constraint is violated, we remove the oldest element from the active set. Put more formally, if $|I'_t| > B$, we set $I_{t+1} = I'_t \setminus \{r_t\}$, where $r_t = \min I'_t$, and otherwise, if $|I'_t| \leq B$, we set $I_{t+1} = I_t$. Following the notation established in the previous section, we can rewrite f_t as

$$f_t \;=\; \sum_{i \in I_t} y_i \, \sigma_{i,t} \, K(\mathbf{x}_i, \cdot), \qquad \text{where} \qquad \sigma_{i,t} \;=\; \prod_{j \in I_{t-1} \,\wedge\, j \geq i} \phi_j .$$

The pseudocode of the Forgetron algorithm is given in Figure 2.

We now turn to the analysis of the Forgetron algorithm. Recall that our goal is to prove a mistake bound similar to that of the Perceptron (see Theorem 1), relative to any competitor g from the set $\{g \in \mathcal{H}_K : \|g\| \leq U\}$. To gain some insight into our proof technique, assume that g attains a zero loss on every example from the input sequence, that is, $y_t \, g(\mathbf{x}_t) \geq 1$ for all t. As in the proof of Theorem 1, we prove a mistake bound for the Forgetron by tracking the dynamics of $\|f_t - g\|^2$. We informally refer to $\|f_t - g\|^2$ as our instantaneous distance from the competitor g. Initially, $f_1 \equiv 0$, and therefore $\|f_1 - g\|^2 = \|g\|^2$. For rounds on which the Forgetron makes a prediction mistake, we first perform the Perceptron update and obtain f'_t. From Lemma 1 we know that $\|f_t - g\|^2 - \|f'_t - g\|^2 \geq 2y_t g(\mathbf{x}_t) - 1 \geq 1$, namely, the Perceptron update moves our classifier closer to g by at least one unit. Next, we perform the shrinking and removal steps. These steps might increase the distance between our classifier and g. Suppose that we could show that the deviation caused by these two steps is at most a half. Then overall, after performing the three step update, the distance between our

```
INPUT: symmetric positive semidefinite kernel $K(\cdot,\cdot)$ ; budget $B > 0$
INITIALIZE: $I_1 = \emptyset$ ; $f_1 \equiv 0$ ; $U = \frac{1}{4}\sqrt{\frac{B+1}{\log(B+1)}}$
For $t = 1, 2, \dots$
    receive an instance $\mathbf{x}_t$
    predict $\mathrm{sign}(f_t(\mathbf{x}_t))$
    receive correct label $y_t$
    If $y_t f_t(\mathbf{x}_t) > 0$
      set $I_{t+1} = I_t$
          and for all $(i \in I_t)$ set $\sigma_{i,t+1} = \sigma_{i,t}$
    Else
      (1) set $I'_t = I_t \cup \{t\}$
          // define $f'_t = f_t + y_t K(\mathbf{x}, \cdot)$
      (2) set $\phi_t = \min\{ (B+1)^{-\frac{1}{2(B+1)}} , U/\|f'_t\| \}$
          set $\sigma_{t,t+1} = \phi_t$ and for all $(i \in I_t)$ set $\sigma_{i,t+1} = \phi_t \, \sigma_{i,t}$
          // define $f''_t = \phi_t f'_t$
      (3) If $|I'_t| \le B$
            set $I_{t+1} = I'_t$
          Else
            define $r_t = \min I_t$
            set $I_{t+1} = I'_t \setminus \{r_t\}$
    define $f_{t+1} = \sum_{i \in I_{t+1}} \sigma_{i,t+1} \, y_i \, K(\mathbf{x}_i, \cdot)$
```

FIG. 2. *The basic Forgetron algorithm.*

classifier and g decreases by at least a half. Therefore, after M prediction mistakes, the distance to g decreases by at least $\frac{1}{2}M$. Using the facts that the initial distance to g is $\|g\|^2$ and the final distance cannot be negative, we conclude that $\|g\|^2 - \frac{1}{2}M \ge 0$, which gives us a bound on M. Therefore, to obtain a mistake bound, we must bound the total amount by which the shrinking and removal steps increase our distance to g. We now formalize this intuition and prove the following relative mistake bound.

THEOREM 2. *Let $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_T, y_T)$ be a sequence of examples such that $K(\mathbf{x}_t, \mathbf{x}_t) \le 1$ for all t. Assume that this sequence is presented to the Forgetron algorithm with a budget parameter $B \ge 83$ and with ϕ_t defined as in* (25)*. Let g be a function in $\mathcal{H}_K$ such that $\|g\| \le U$, where U is given by* (26)*, and define $\ell^\star_t = \ell\big(g; (\mathbf{x}_t, y_t)\big)$. Then, the number of prediction mistakes made by the Forgetron on this sequence is at most*

$$M \;\le\; 2\,\|g\|^2 \;+\; 4\sum_{t=1}^{T} \ell^\star_t .$$

Before proving this theorem, we must quantify the negative effect of the shrinking and removal steps on our mistake bound. As before, let J denote the set of rounds on which the Forgetron makes a prediction mistake, and for every $t \in J$ define Φ_t as in (24). The role played by Φ_t in our analysis below is similar to its role in the analysis of the shrinking Perceptron in section 4. Namely, Φ_t bounds the effect of the shrinking step on our mistake bound. Furthermore, let t be a round in J on which $|I_t| = B$, and let r_t denote the index of the example which is removed from the active set on that round. Recall the definition of the function Ψ in (15) and define

$$\Psi_t \;=\; \begin{cases} \Psi\Big(\sigma_{r_t,t+1}\ ,\ y_{r_t} f''_t(\mathbf{x}_{r_t})\Big) & \text{if } t \in J \ \wedge\ |I_t| = B, \\ 0 & \text{otherwise.} \end{cases} \tag{29}$$

It should come as no surprise that the function Ψ plays a role in the analysis of the removal step of the Forgetron update similar to the role it played in our analysis of the Remove-Oldest Perceptron in section 3. The following lemma formalizes the relationship between Φ_t, Ψ_t, and the number of mistakes made by the Forgetron.

LEMMA 6. *Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of examples such that $K(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ for all t and assume that this sequence is presented to the Forgetron algorithm. Let Φ_t and Ψ_t be as defined in* (24) *and* (29)*, respectively. Then, the following bound holds for any $g \in \mathcal{H}_K$:*

$$M - \left(\|g\|^2 \sum_{t\in J} \log(1/\Phi_t) + \sum_{t \in J} \Psi_t \right) \;\leq\; \|g\|^2 + 2 \sum_{t \in J} \ell^\star_t .$$

Proof. For each t define $\Delta_t = \|f_t - g\|^2 - \|f_{t+1} - g\|^2$. As in our previous proofs, we prove the lemma by bounding $\sum_{t=1}^{T} \Delta_t$ from above and from below. First note again that $\sum_t \Delta_t$ is a telescopic sum which collapses to $\|f_1 - g\|^2 - \|f_{T+1} - g\|^2$. Using the facts that $\|f_{T+1} - g\|^2 \geq 0$ and that $f_1 \equiv 0$, we obtain the upper bound

$$\sum_{t=1}^{T} \Delta_t \;\leq\; \|g\|^2. \tag{30}$$

Next we show a lower bound on $\sum_t \Delta_t$. On rounds where the Forgetron makes a correct prediction, we have that $f_{t+1} = f_t$ and thus

$$\sum_{t=1}^{T} \Delta_t = \sum_{t \in J} \Delta_t. \tag{31}$$

Next, we rewrite Δ_t as a sum of three terms for rounds on which the Forgetron makes a mistake,

$$\Delta_t \;=\; \underbrace{\|f_t\ - g\|^2 - \|f'_t\ - g\|^2}_{\alpha_t} + \underbrace{\|f'_t\ - g\|^2 - \|f''_t\ \ g\|^2}_{\beta_t} + \underbrace{\|f''_t\ - g\|^2 - \|f_{t+1}\ - g\|^2}_{\gamma_t}, \tag{32}$$

where f'_t and f''_t are defined in (27) and (28), respectively. Summing over $t \in J$ and using (13) we get that

$$\sum_{t \in J} \Delta_t \;=\; \sum_{t \in J} \alpha_t + \sum_{t \in J} \beta_t + \sum_{t \in J} \gamma_t \;=\; \sum_{t \in I_{T+1}} \alpha_t + \sum_{t \in J : |I_t| = B} (\alpha_{r_t} + \gamma_t) \;+\; \sum_{t \in J} \beta_t. \tag{33}$$

We now bound each of the summands in the above equation. First, we use Lemma 1 and (14) to get that

$$\sum_{t \in I_{T+1}} \alpha_t \;\geq\; \sum_{t \in I_{T+1}} (1 - 2\ell^\star_t)\,. \tag{34}$$

Recall that $f'_{r_t} = f_{r_t} + y_{r_t} K(\mathbf{x}_{r_t}, \cdot)$, and in addition we can rewrite f_{t+1} as $f''_t - \sigma_{r_t,t+1} y_{r_t} K(\mathbf{x}_{r_t}, \cdot)$. Using Lemma 2 with $f = f_{r_t}$, $f' = f''_t$, and $\lambda = \sigma_{r_t,t+1}$, we get that for any $t \in J$ for which $|I_t| = B$ we have that

$$\alpha_{r_t} + \gamma_t \ \geq \ 1 - 2\ell^\star_{r_t} - \Psi_t.$$

Combining the above with (34) gives

$$\sum_{t \in I_{T+1}} \alpha_t + \sum_{t \in J : |I_t| = B} (\alpha_{r_t} + \gamma_t) \ \geq \ \sum_{t \in I_{T+1}} (1 - 2\ell^\star_t) \ + \sum_{t \in J : |I_t| = B} \left(1 - 2\ell^\star_{r_t} - \Psi_t\right).$$

Note that for each $t \in J$ we have that either $t \in I_{T+1}$ or there exists $i \in J$ for which $|I_i| = B$ and $r_i = t$. In addition, Ψ_t is defined to be zero if on round t we do not remove any element from the active set. Therefore, we can further write

$$\sum_{t \in I_{T+1}} \alpha_t + \sum_{t \in J : |I_t| = B} (\alpha_{r_t} + \gamma_t) \ \geq \ M - 2 \sum_{t \in J} \ell^\star_t - \sum_{t \in J} \Psi_t. \tag{35}$$

Next, we bound $\sum_t \beta_t$ using Corollary 2:

$$\sum_{t \in J} \beta_t \ \geq \ \|g\|^2 \sum_{t \in J} \log(\Phi_t) \ = \ -\|g\|^2 \sum_{t \in J} \log(1/\Phi_t). \tag{36}$$

Using (35) and (36) in (33) yields

$$\sum_{t \in J} \Delta_t \ \geq \ M - 2 \sum_{t \in J} \ell^\star_t - \|g\|^2 \sum_{t \in J} \log(1/\Phi_t) - \sum_{t \in J} \Psi_t.$$

Combining the above with (30) and (31) gives

$$M - \left(\|g\|^2 \sum_{t \in J} \log(1/\Phi_t) + \sum_{t \in J} \Psi_t \right) \ \leq \ \|g\|^2 + 2 \sum_{t \in J} \ell^\star_t.$$

This concludes the proof. □

Lemma 6 bounds the total damage to our mistake bound due to the shrinking and removal steps. To prove Theorem 2, we show that our choice of the shrinking coefficient in (25) ensures that the term $\left(\|g\|^2 \sum_{t \in J} \log(1/\Phi_t) + \sum_{t \in J} \Psi_t\right)$ is well behaved. First, we prove an upper bound on $\|g\|^2 \sum_{t \in J} \log(1/\Phi_t)$, the negative effect due to the shrinking step of the Forgetron update.

Lemma 7. *Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of examples presented to the Forgetron algorithm with ϕ_t defined as in* (25). *Let J denote the online iterations on which the Forgetron algorithm makes a prediction mistake, and let $M = |J|$. Let g be any function in $\mathcal{H}_K$ with $\|g\| \leq U$, where U is given in* (26), *and let Φ_t be as defined in* (24). *Then*

$$\|g\|^2 \sum_{t \in J} \log(1/\Phi_t) \ \leq \ \frac{M}{32}.$$

Proof. We begin the proof by showing that

$$\Phi_t \ \geq \ (B+1)^{-\frac{1}{2(B+1)}} \tag{37}$$

for all $t \in J$. If $t \in J_1$ then $\Phi_t = 1$, which is clearly greater than $(B+1)^{-\frac{1}{2(B+1)}}$. If $t \in J_2$ then Φ_t is defined to equal ϕ_t. It follows from the definition of J_2 in (21) that $U \geq \|f'_t\|$, and therefore

$$(B+1)^{-\frac{1}{2(B+1)}} \;\leq\; 1 \;\leq\; \frac{U}{\|f'_t\|}.$$

Referring back to (25), we get that $\phi_t = (B+1)^{-\frac{1}{2(B+1)}}$, and therefore (37) holds in this case as well. Finally, if $t \in J_3$ then Φ_t is defined to equal $\phi\|f'_t\|/U$. From the definition of J_3 in (21), we have that $U < \|f'_t\|$, and therefore $\Phi_t > \phi_t$. If $\phi_t = (B+1)^{-\frac{1}{2(B+1)}}$ then (37) holds trivially. Otherwise, $\phi_t = U/\|f'_t\|$, $\Phi_t > 1$, and once again (37) holds. We can now rewrite (37) as

$$\log\left(\frac{1}{\Phi_t}\right) \;\leq\; \frac{\log(B+1)}{2(B+1)}.$$

Combining the above with the assumption that $\|g\|^2 \leq U^2$ and using the definition of U in (26) results in

$$\|g\|^2 \log(1/\Phi_t) \;\leq\; \frac{B+1}{16\log(B+1)}\,\frac{\log(B+1)}{2(B+1)} \;=\; \frac{1}{32}.$$

Summing both sides of the above over all $t \in J$ proves the lemma. □

Next, we prove that our choice of ϕ_t in (25) guarantees an upper bound on $\sum_{t\in J}\Psi_t$, the negative effect due to the removal step of the Forgetron update.

LEMMA 8. *Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of examples such that $K(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ for all t. Assume that this sequence is presented to the Forgetron algorithm with a budget parameter $B \geq 83$ and with ϕ_t defined as in* (25)*. Let J denote the online iterations on which the Forgetron algorithm makes a prediction mistake and let $M = |J|$. Let g be a function in $\mathcal{H}_K$ such that $\|g\| \leq U$, where U is given in* (26)*, and let Ψ_t be as defined in* (15)*. Then,*

$$\sum_{t\in J}\Psi_t \;\leq\; \frac{15M}{32}.$$

Proof. Let t be an online round in J, and recall that I_t is the active set of the Forgetron algorithm on round t and that B is the predefined memory budget. If $|I_t| < B$ then $\Psi_t = 0$. Otherwise, Ψ_t equals

$$\Psi_t = \sigma_{r_t,t+1}^2 + 2\,\sigma_{r_t,t+1} - 2\,\sigma_{r_t,t+1}\,y_{r_t}\,f''_t(\mathbf{x}_{r_t}). \tag{38}$$

The definition of ϕ_t given in (25) implies that $\phi_t \leq (B+1)^{-1/(2(B+1))}$ for all $t \in J$. Since the oldest element in the active set, whose index is r_t, is scaled $B+1$ times before it is removed from the active set, we get

$$\sigma_{r_t,t+1} \;\leq\; \left((B+1)^{-\frac{1}{2(B+1)}}\right)^{B+1} \;=\; \frac{1}{\sqrt{B+1}}. \tag{39}$$

Next, we use the Cauchy–Schwarz inequality to bound the term $-y_{r_t}f''_t(\mathbf{x}_{r_t})$ by $\|f''_t\|\,\|K(\mathbf{x}_{r_t},\cdot)\|$. The definition of ϕ_t implies that $\|f''_t\| \leq U$, and we assumed

$K(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ for all t, so $-y_{r_t} f_t''(\mathbf{x}_{r_t}) \leq U$. Plugging this inequality and the inequality in (39) into (38) gives

$$\Psi_t \ \leq \ \frac{1}{B+1} + \frac{2}{\sqrt{B+1}} + \frac{2\,U}{\sqrt{B+1}}.$$

Using the definition of U from (26), we have

$$\Psi_t \ \leq \ \frac{1}{B+1} + \frac{2}{\sqrt{B+1}} + \frac{1}{2\sqrt{\log(B+1)}}. \tag{40}$$

The right-hand side of the above inequality decreases monotonically with B and is at most $15/32$ for $B \geq 83$. Thus, $\sum_{t \in J} \Psi_t \leq \frac{15M}{32}$. □

Proof of Theorem 2. From Lemma 6 we have

$$M - \left(\|g\|^2 \sum_{t \in J} \log(1/\Phi_t) + \sum_{t \in J} \Psi_t \right) \ \leq \ \|g\|^2 + 2 \sum_{t \in J} \ell_t^\star.$$

Plugging the bounds in Lemmas 7 and 8 into the above inequality gives

$$M - \left(\frac{M}{32} + \frac{15M}{32} \right) \ \leq \ \|g\|^2 + 2 \sum_{t \in J} \ell_t^\star.$$

Multiplying both sides of the above inequality by 2 provides the desired mistake bound. □

We have shown that the choice of ϕ_t in (25) indeed results in a provably correct learning algorithm on a budget. However, this definition of ϕ_t suffers from several drawbacks. First and foremost, the resulting algorithm performs poorly in practice. In the next section, we present the self-tuned Forgetron, which uses a refined shrinking mechanism and significantly outperforms the algorithm presented in this section (see experimental results in section 8). Another problem with the definition of ϕ_t in (25) is that it forces $\|f_t''\|$ to be at most U. We used this property in the proof above to bound $-y_{r_t} f_t''(\mathbf{x}_{r_t})$, which in turn provided us with an upper bound on Ψ_t. In practice, it is often the case that r_t can be safely removed from the active set without any shrinking, and the norm of f_t'' can be allowed to grow beyond U. The refined shrinking mechanism of the self-tuned Forgetron uses the actual values of $\Psi_1, \ldots, \Psi_t$ to define ϕ_t and does not explicitly use U.

6. The self-tuned Forgetron.

In section 5 we introduced the Forgetron framework and proposed a simple definition of the shrinking coefficients in (25). Besides constants, which do not change from round to round, the definition in (25) depends solely on $\|f_t'\|$. Moreover, it makes explicit use of the upper bound U. In this section we propose an improved shrinking scheme which does not rely on the knowledge of U. We name the resulting algorithm the *self-tuned Forgetron*. The main principle which we follow in the derivation of the self-tuned Forgetron is to apply the gentlest possible shrinking step. For example, if we are fortunate, and the damage from the removal step happens to be small without applying any shrinking, then our improved shrinking scheme will set $\phi_t = 1$. On such rounds, the self-tuned Forgetron algorithm update reduces back to the Remove-Oldest Perceptron update discussed in section 3.

Recall that in our analysis in section 5, Lemma 7 provided an upper bound on $\|g\|^2 \sum_{t \in J} \log(1/\Phi_t)$ and Lemma 8 provided an upper bound on $\sum_{t \in J} \Psi_t$. Together with Lemma 6, these upper bounds were sufficient to prove the Forgetron mistake bound in Theorem 2. On every update, the self-tuned Forgetron chooses the gentlest shrinking that still ensures that the bounds in Lemmas 7 and 8 still hold and that our mistake bound remains valid. More formally, given an input sequence of examples of length T, define M_t to be the number of prediction mistakes made by our algorithm on rounds $\{1, 2, \ldots, t\}$. On round t, if an online update is invoked, the self-tuned Forgetron chooses the shrinking coefficient ϕ_t to be the largest number in $(0, 1]$ that satisfies the condition

$$\forall t, \quad \sum_{i \in J \,:\, i \leq t} \Psi_i \ \leq\ \frac{15}{32} M_t. \tag{41}$$

More concretely, define

$$Q_t \ =\ \sum_{i \in J \,:\, i < t} \Psi_i.$$

Let $t \in J$ be an index of a round on which the Forgetron makes a prediction mistake and is required to remove an example from the active set ($|I_t| = B$). The tth constraint from (41) can be rewritten as

$$\Psi\big(\sigma_{r_t,t}\,\phi_t\ ,\ y_{r_t}\phi_t f_t'(\mathbf{x}_{r_t})\big)\ +\ Q_t\ \leq\ \frac{15}{32} M_t.$$

The self-tuned Forgetron sets ϕ_t to be the maximal value in $(0, 1]$ for which the above inequality holds, namely,

$$\phi_t = \max\left\{\phi \in (0,1]\ :\ \Psi\big(\sigma_{r_t,t}\,\phi\ ,\ y_{r_t}\phi f_t'(\mathbf{x}_{r_t})\big) + Q_t\ \leq\ \frac{15}{32}M_t\right\}. \tag{42}$$

Note that Ψ is a quadratic function in ϕ, and thus the optimal value of ϕ_t can be found analytically. Simple algebraic manipulations yield that

$$\phi_t\ =\ \begin{cases} \min\left\{1, \frac{-b+\sqrt{d}}{2a}\right\} & \text{if } a > 0\ \vee\ (a < 0\ \wedge\ d > 0\ \wedge\ \frac{-b-\sqrt{d}}{2a} > 1), \\ \min\{1, -c/b\} & \text{if } a = 0, \\ 1 & \text{otherwise,} \end{cases} \tag{43}$$

where

$$\begin{gathered} a = \sigma_{r_t,t}^2 - 2\,\sigma_{r_t,t}\,y_{r_t} f_t'(\mathbf{x}_{r_t}), \quad b = 2\,\sigma_{r_t,t}, \\ c = Q_t - \tfrac{15}{32} M_t, \quad \text{and} \quad d = b^2 - 4ac. \end{gathered} \tag{44}$$

The pseudocode of the self-tuned Forgetron is given in Figure 3.

By construction, the effect of the removal step is upper bounded by

$$\sum_{t \in J} \Psi_t\ \leq\ \frac{15}{32} M. \tag{45}$$

This fact replaces Lemma 8. Therefore, to apply the same proof technique as in the previous section, it now suffices to show that the bound

$$\|g\|^2 \sum_t \log(1/\Phi_t) \leq \frac{1}{32} M \tag{46}$$

```
INPUT: symmetric positive semidefinite kernel $K(\cdot,\cdot)$ ; budget $B > 0$
INITIALIZE: $I_1 = \emptyset$ ; $f_1 \equiv 0$ ; $Q_1 = 0$ ; $M_0 = 0$
For $t = 1, 2, \ldots$
    receive an instance $\mathbf{x}_t$
    predict sign$(f_t(\mathbf{x}_t))$
    receive correct label $y_t$
    If $y_t f_t(\mathbf{x}_t) > 0$
      set $I_{t+1} = I_t$, $Q_{t+1} = Q_t$, $M_t = M_{t-1}$,
          and for all $(i \in I_t)$ set $\sigma_{i,t+1} = \sigma_{i,t}$
    Else
      set $M_t = M_{t-1} + 1$
      (1) set $I'_t = I_t \cup \{t\}$
          // define $f'_t = f_t + y_t K(\mathbf{x}, \cdot)$
      If $|I'_t| \le B$
          set $I_{t+1} = I'_t$, $Q_{t+1} = Q_t$, $\sigma_{t,t} = 1$,
              and for all $(i \in I_{t+1})$ set $\sigma_{i,t+1} = \sigma_{i,t}$
      Else
          (2) define $r_t = \min I_t$
              define $a, b, c, d$ as in (44) and set $\phi_t$ as in (43)
              set $\sigma_{t,t+1} = \phi_t$ and for all $(i \in I_t)$ set $\sigma_{i,t+1} = \phi_t\, \sigma_{i,t}$
              set $Q_{t+1} = \Psi\big(\sigma_{r_t,t+1}\ ,\ y_{r_t} f''_t(\mathbf{x}_{r_t})\big) + Q_t$
              // define $f''_t = \phi_t f'_t$
          (3) set $I_{t+1} = I'_t \setminus \{r_t\}$
    define $f_{t+1} = \sum_{i \in I_{t+1}} \sigma_{i,t+1}\, y_i\, K(\mathbf{x}_i, \cdot)$
```

FIG. 3. *The self-tuned Forgetron algorithm.*

still holds. To prove the above inequality, we require the following lemma.

LEMMA 9. *Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of examples such that $K(\mathbf{x}_t, v\mathbf{x}_t) \le 1$ for all t and assume that this sequence is presented to the self-tuned Forgetron with a budget parameter $B \ge 83$. Let J denote the set of rounds on which the algorithm makes a prediction mistake, let ϕ_t be as in* (42)*, and let Φ_t be as defined in* (24)*. Finally, let t be a round in J such that $\Phi_t < 1$. Then,*

$$\Phi_t\, \sigma_{r_t,t} \ \ge\ \frac{1}{\sqrt{B+1}}.$$

Proof. Define

$$\phi' \ =\ \min\left\{ 1\ ,\ \frac{U}{\|f'_t\|}\ ,\ \frac{1}{\sigma_{r_t,t}\sqrt{B+1}} \right\}.$$

This definition implies the following: (i) $\phi' \in (0, 1]$. (ii) $\phi' \|f'_t\| \le U$, and, therefore, using the Cauchy–Schwarz inequality, $\phi' f'_t(\mathbf{x}_{r_t}) \le U$. (iii) $\sigma_{r_t,t}\, \phi' \le 1/\sqrt{B+1}$. Therefore,

$$(\sigma_{r_t,t}\, \phi')^2 + 2\sigma_{r_t,t}\, \phi'\, (1 - y_{r_t} \phi' f'_t(\mathbf{x}_{r_t})) \ \le\ \frac{1}{B+1} + \frac{2}{\sqrt{B+1}}(1 + U).$$

The left-hand side of the above equals $\Psi\big(\sigma_{r_t,t+1},\ y_{r_t} f''_t(\mathbf{x}_{r_t})\big)$. Using the definition of U we get that

$$\Psi\big(\sigma_{r_t,t}\phi', y_{r_t}\phi' f'_t(\mathbf{x}_{r_t})\big) \ \le\ \frac{1}{B+1} + \frac{2}{\sqrt{B+1}} + \frac{1}{2\sqrt{\log(B+1)}}.$$

The right-hand side of the above inequality is at most $\frac{15}{32}$ for $B \geq 83$. In addition, the definition of the self-tuned Forgetron implies that $Q_t \leq \frac{15}{32} M_{t-1}$ for each t. Therefore,

$$\Psi\big(\sigma_{r_t,t}\phi' \ , \ y_{r_t}\phi' f'_t(\mathbf{x}_{r_t})\big) + Q_t \ \leq \ \frac{15}{32} M_t. \tag{47}$$

Since ϕ' is in $(0,1]$ and satisfies (47), and ϕ_t is the largest value which satisfies (47), we get that $\phi_t \geq \phi'$. By the definition of Φ_t in (24) we have $\Phi_t \geq \phi_t$, and therefore $\Phi_t \geq \phi'$. We have therefore reduced our problem to proving $\phi' \sigma_{r_t,t} \geq 1/\sqrt{B+1}$.

The assumption that $\Phi_t < 1$ implies that $\phi' < 1$ as well. We are left with two possibilities: either $\phi' = U/\|f'_t\|$ or $\phi' = \frac{1}{\sigma_{r_t,t}\sqrt{B+1}}$. If $\phi' = U/\|f'_t\|$, then

$$\phi_t \, \|f'_t\| \ \geq \ \phi' \, \|f'_t\| \ = \ \frac{U}{\|f'_t\|} \, \|f'_t\| \ = \ U.$$

Therefore, $t \in J_1$, that is, the norm of the hypothesis after the shrinking step is still as large as U (see also Figure 1). This immediately implies that $\Phi_t = 1$, which stands in contradiction to the assumption that $\Phi_t < 1$. We have thus shown that ϕ' must equal $1/(\sigma_{r_t,t}\sqrt{B+1})$. It therefore holds that $\phi' \sigma_{r_t,t} \geq 1/\sqrt{B+1}$, and this concludes our proof. ▯

Equipped with the above lemma, we can prove a mistake bound for the self-tuned Forgetron.

THEOREM 3. *Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of examples such that $K(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ for all t. Assume that this sequence is presented to the self-tuned Forgetron of Figure 3 with a budget parameter $B \geq 83$. Let g be a hypothesis in $\mathcal{H}_K$ such that $\|g\| \leq U$, where U is given by (26), and define $\ell^\star_t = \ell\big(g; (\mathbf{x}_t, y_t)\big)$. Then, the number of prediction mistakes made by the self-tuned Forgetron on this sequence is at most*

$$M \ \leq \ 2\,\|g\|^2 \ + \ 4\sum_{t=1}^{T} \ell^\star_t.$$

Proof. We follow the proof of Theorem 2. The bound in (45) holds by construction, and therefore it suffices to show that (46) holds. Since $\|g\| \leq U$, we know that

$$\|g\|^2 \sum_{t \in J} \log(1/\Phi_t) \ \leq \ \frac{B+1}{16\log(B+1)} \sum_{t \in J} \log(1/\Phi_t).$$

Therefore, to prove that (46) holds, it suffices to show that

$$\sum_{t \in J} \log(1/\Phi_t) \ \leq \ \frac{\log(B+1)}{2(B+1)}\, M$$

or, equivalently, that

$$\prod_{t \in J} \Phi_t \ \geq \ (B+1)^{-\frac{M}{2(B+1)}}. \tag{48}$$

We prove the above inequality by strong induction on the number of prediction mistakes made by the self-tuned Forgetron. Once again, J denotes the online rounds on which the algorithm made a prediction mistake. First note that if $|J| < B$ then $\phi_t = 1$

for all $t \in J$, in which case the claim is trivial. Therefore, we assume that $|J| \geq B$. Assume that the claim holds for every $J' \subset J$ (which means that $|J'| < M$), and let us prove the claim for J. That is, we need to show that

$$\prod_{t \in J} \Phi_t \;\geq\; (B+1)^{-\frac{|J|}{2(B+1)}}. \tag{49}$$

Let $j = \max J$ denote the index of the last element that was inserted into J. If $\Phi_j = 1$, then

$$\prod_{t \in J} \Phi_t \;=\; \prod_{t \in J \setminus \{j\}} \Phi_t.$$

Applying the inductive assumption to the set $J' = J \setminus \{j\} \subset J$, we get that

$$\prod_{t \in J} \Phi_t \;=\; \prod_{t \in J'} \Phi_t \;\geq\; (B+1)^{-\frac{|J'|}{2(B+1)}} \;\geq\; (B+1)^{-\frac{|J|}{2(B+1)}}.$$

Therefore, it is left to show that the claim holds for $\Phi_j < 1$. Recall that I'_j denotes the active set after applying the Perceptron update step and before applying the removal step on round j. Using the inductive assumption on the set $J' = J \setminus I'_j$, we have $|J'| = |J| - (B+1)$, and therefore,

$$\prod_{t \in J} \Phi_t \;=\; \prod_{t \in J'} \Phi_t \prod_{t \in I'_j} \Phi_t \;\geq\; (B+1)^{-\frac{|J|-(B+1)}{2(B+1)}} \prod_{t \in I'_j} \Phi_t. \tag{50}$$

Recall that $r_j = \min I'_j$. Using the fact that $\Phi_j \geq \phi_j$ and the definition of $\sigma_{r_j,j}$, we get that

$$\prod_{t \in I'_j} \Phi_t \;\geq\; \Phi_j \prod_{t \in I_j} \phi_t \;=\; \Phi_j \, \sigma_{r_j,j}.$$

From Lemma 9 we know that the right-hand side of the above is at least $1/\sqrt{B+1}$. Using this fact in (50) gives

$$\prod_{t \in J} \Phi_t \;\geq\; (B+1)^{-\frac{|J|-(B+1)}{2(B+1)}} \, \frac{1}{\sqrt{B+1}} \;=\; (B+1)^{-\frac{|J|}{2(B+1)}}.$$

This concludes our proof. □

7. A greedy removal scheme. The variants of the Forgetron algorithm we discussed so far always remove the oldest element from the active set. The accompanying shrinking step controls the damage due to the removal step. Our approach stands in contrast to earlier online learning algorithms on a budget [3, 12] which focus on choosing which example to remove from the active set and do not take any measures to control the damage due to this removal. While earlier work did not provide any mistake bounds, we would like to build on the intuition conveyed in previous work to devise a greedy removal scheme that may skip the shrinking step when possible.

In this section we describe an extension of the Forgetron framework which allows removal of examples other than the oldest one. Our removal criterion is based on the analysis presented in section 5. Specifically, in Lemma 6 we showed that the damage inflicted upon the hypothesis due to the removal step is $\Psi\big(\sigma_{r_t,t}\,\phi_t, y_{r_t}\phi_t f''_t(\mathbf{x}_{r_t})\big)$. The goal of the shrinking step is to ensure that the total damage due to the removal step is at most $\frac{15}{32}$ times the number of prediction mistakes. According to our analysis, if there exists an example $i \in I_t$ for which

$$\Psi\big(\sigma_{r_t,t}, y_{r_t} f'_t(\mathbf{x}_{r_t})\big) \;\leq\; \frac{15}{32}, \tag{51}$$

then this example can be safely removed from the active set without any shrinking. We therefore employ the following two stage approach. If indeed there exists an index $i \in I_t$ for which (51) holds, then we skip the shrinking step and remove this index from the active set. Otherwise, we perform the self-tuned Forgetron update discussed in the previous section. Formally, define

$$j \;=\; \arg\min i \in I_t \Psi(\sigma_{i,t}\, ,\, y_i f'_t(\mathbf{x}_i)).$$

The example to be removed is set to

$$r_t \;=\; \begin{cases} j & \text{if } \;\Psi(\sigma_{j,t}\, ,\, y_j f'_t(\mathbf{x}_j)) \;\leq \frac{15}{32}, \\ \min I_t & \text{otherwise.} \end{cases} \tag{52}$$

The shrinking coefficient ϕ_t is set as before, namely,

$$\phi_t \;=\; \max\left\{\phi \in (0,1] \;:\; \Psi\big(\sigma_{r_t,t}\,\phi\;,\; y_{r_t}\phi f'_t(\mathbf{x}_{r_t})\big) \;+\; Q_t \;\leq\; \frac{15}{32} M_t\right\},$$

where $Q_t = \sum_{i \in J : i < t} \Psi_i$.

The greedy removal scheme entertains the mistake bound proven for the self-tuned Forgetron. To see this, first note that Lemma 6 does not assume that $r_t = \min I_t$. In fact, the lemma holds for any choice of $r_t \in I_t$. In particular, Lemma 6 holds for the example chosen by the greedy removal scheme. In addition, the inequality $\sum_t \Psi_t \leq \frac{15}{32} M$ holds by construction. Thus, it again suffices to show that

$$\|g\|^2 \sum_t \log(1/\Phi_t) \leq \frac{1}{32} M. \tag{53}$$

To prove the above inequality, note that whenever $\Phi_t < 1$ (and thus $\phi_t < 1$) we have that $r_t = \min I_t$. Therefore, the update coincides with the update of the self-tuned Forgetron and the proof of Lemma 9 can be repeated verbatim. Moreover, it is immediate to verify that the same reasoning used to prove Theorem 3 carries over to the greedy removal scheme. In summary, the bound of Theorem 3 also applies to the greedy removal scheme.

8. Experiments. In this section we present experimental results which demonstrate the merits of the Forgetron algorithms. Since the focus of this paper is on the online-learning setting, we ran different online algorithms on various datasets and we report the online error for each experiment. The online error is the number of prediction mistakes an algorithm makes on a sequence of examples, divided by the sequence length. Throughout this section we consistently use the online error to assess the performance of the different algorithms.

We compare the performance of our algorithms with the two methods described in [3] and [1], abbreviated by CKS and CG, respectively, and with the standard kernel-based Perceptron. The CKS algorithm is a variant of the kernel-based Perceptron,

Table 8.1

Each of the three tables corresponds to a different ratio between B, the budget parameter, and p, the size of the active set used by the Perceptron on the respective dataset. For example, the top table sets the B to be a quarter of the number of mistakes suffered by the standard Perceptron algorithm. Each entry in the table gives the average online error attained by the algorithms on each dataset.

$B = p/4$

	Perceptron	F (basic)	F (slf-tuned)	F (greedy)	CKS
MNIST	6.08	35.22	11.25	9.54	17.45
USPS	7.73	40.70	14.88	12.70	18.52
ADULT	20.33	30.44	22.31	24.06	33.48
synth. (5%)	9.56	11.60	9.89	11.84	32.76
synth. (10%)	18.16	20.30	18.38	21.07	41.13

$B = p/2$

	Perceptron	F (basic)	F (slf-tuned)	F (greedy)	CKS
MNIST	6.08	27.05	8.62	7.78	9.02
USPS	7.73	31.95	11.03	9.78	10.26
ADULT	20.33	26.67	21.40	23.70	27.82
synth. (5%)	9.56	10.66	9.70	11.98	20.16
synth. (10%)	18.16	19.10	18.27	21.74	30.05

$B = p$

	Perceptron	F (basic)	F (slf-tuned)	F (greedy)	CKS
MNIST	6.08	16.07	6.08	6.08	6.08
USPS	7.73	20.29	7.73	7.73	7.73
ADULT	20.33	22.06	20.33	20.33	20.33
synth. (5%)	9.56	9.79	9.56	9.56	9.56
synth. (10%)	18.16	18.37	18.16	18.16	18.16

which uses the following heuristic to enforce a strict memory budget. When the budget is exceeded, the algorithm calculates the margin attained by removing each active example from the active set and then applying the resulting hypothesis to the removed example. The removed example is the one which attains the maximal margin. This removal scheme is similar to the removal scheme described in section 7. The CKS algorithm guarantees only that its removal scheme does not damage the accuracy of the hypothesis when the margin attained by the removed example is greater than one. If no such example exists in the active set, no formal guarantees are provided. We therefore anticipate that the CKS algorithm would work well when the examples form a separable dataset but is likely to fail on more difficult, inseparable, datasets.

The CG algorithm is a randomized method for online learning on a budget. When the CG algorithm exceeds its budget, it removes a randomly chosen example from the active set. In all our experiments, we ran the CG algorithm 10 times on each dataset and report the online error averaged over the 10 different runs. A disadvantage of this average-case analysis in the online setting is that in real-world online-learning problems, we typically run the algorithm over the sequence of examples only once. We discuss this disadvantage in our last experiment below.

In all our experiments, we focus on the self-tuned Forgetron described in section 6 and on the greedy removal Forgetron described in section 7. We also conducted experiments with the basic Forgetron algorithm described in section 5; however, its performance was found to be significantly inferior to the other Forgetron variants. This can be attributed to the worst-case definition of the shrinking coefficients employed by the basic Forgetron. Also note that the self-tuned Forgetron and the greedy removal Forgetron are identical to the original Perceptron when the active set used by the Perceptron is less than the budget parameter, while the basic Forgetron is different due to the fixed shrinking coefficient. For clarity, we present the results of

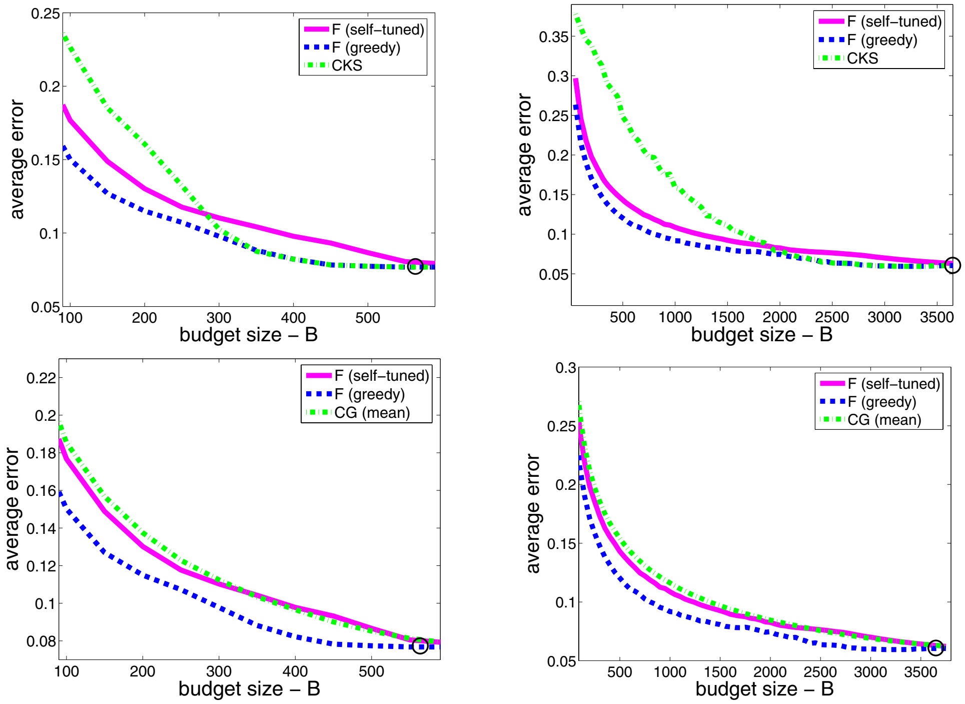


FIG. 4. *The average online error of different budget algorithms as a function of the budget B on the USPS dataset (left) and the MNIST dataset (right). The online error of the Perceptron and its budget requirements for each problem are marked with a circle.*

the basic Forgetron only in Table 8.1 and not in the graphs in Figures 4, 5, 6, and 7.

Our first experiment was performed with two standard datasets: the MNIST dataset, which consists of 60,000 training examples, and the USPS dataset, with 10,000 examples. These two datasets are well known and induce relatively easy classification problems. The instances in both datasets are handwritten images of digits; thus each image corresponds to one of the 10 digit classes. We generated 126 binary problems by splitting the 10 labels into two equal-size sets in all possible ways ($\binom{10}{5}/2 = 126$). We report the online error averaged over these 126 problems. We ran the various algorithms with different values of the budget parameter B, using a Gaussian kernel defined as $K(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\frac{1}{2}\|\mathbf{x}_1 - \mathbf{x}_2\|^2)$. The results of these experiments are summarized in Figure 4 and in Table 8.1. Since the standard Perceptron does not take a budget parameter, we mark its accuracy and active set size in Figure 4 with a small circle. All four algorithms perform quite well on these datasets. It is apparent that for both datasets, the greedy removal Forgetron is slightly better than the alternative methods. Comparing the performance of the self-tuned Forgetron and CKS, we note that the former performs better on small budgets while the latter is better on large budgets. It is also apparent that the average online error of the CG method is similar to the online error of the self-tuned Forgetron.

Our next experiment was performed with the census-income (adult) dataset, which consists of 199,523 examples. This dataset is highly nonbalanced (only 6.21 percent of the labels are positive). We overcame this problem by randomly generating a balanced subset of this dataset. We repeated this process 10 times, generating 10 different balanced datasets. The results we report were obtained by averaging over

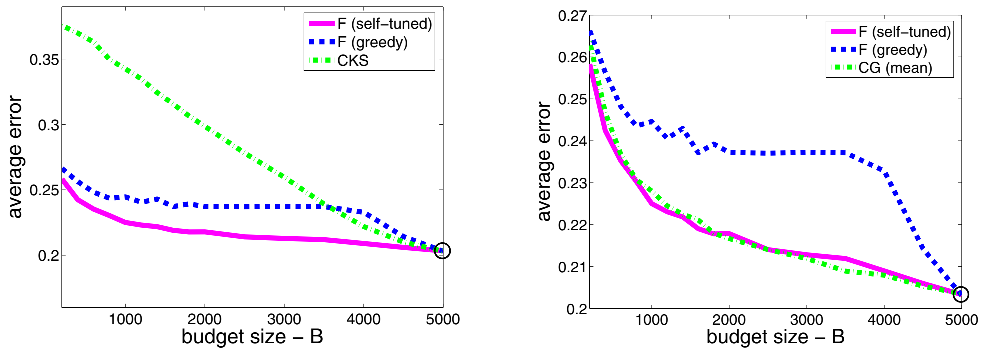


Fig. 5. *The average online error of different budget algorithms as a function of the budget B on the census-income (adult) dataset. The average error of the Perceptron and its budget requirements are marked with a circle.*

the 10 different selections. We first ran the Perceptron algorithm on each dataset with a Gaussian kernel. The online error of the Perceptron was approximately 21 percent. We then ran the various budget algorithms on each dataset with different values of B. The results are given in Figure 5 and in Table 8.1. It is apparent that the self-tuned Forgetron and the CG method perform very well on this dataset and outperform both the greedy removal Forgetron algorithm and the CKS method. The performance of the greedy removal Forgetron is also relatively good for small budgets. The relatively poor performance of CKS on this dataset, when B takes small values, may be due to the difficulty of the classification task. As mentioned above, the analysis of CKS is based on the assumption that there always exists an example whose margin, after its removal, is greater than 1. Whenever we are unfortunate, and there is no such example in the active set, the CKS removal step may significantly damage the accuracy of the current hypothesis.

To further investigate the performance of the various algorithms, our last experiment examines the accuracy of the algorithms in the presence of label noise. Recall that the number of active examples used by the basic Perceptron algorithm grows with each prediction mistake. Therefore, we expect the Perceptron algorithm to require a large active set in the presence of noise. As in our previous experiments, we ran the various budget algorithms with a Gaussian kernel. We generated two synthetic datasets as follows. We randomly sampled 5000 positive examples from a two-dimensional Gaussian with a mean vector of $(1, 1)$ and a diagonal covariance matrix with $(0.2, 2)$ as its diagonal. We then sampled 5000 negative examples from a normal distribution with a mean vector of $(-1, -1)$ and the same covariance as before. Finally, we flipped each label with a probability of 0.1 for the first dataset and with a probability of 0.05 for the second dataset, thus introducing two noise rates. We then presented the data to each of the algorithms. We repeated this process for different values of the budget parameter B, ranging from 10 to 2000. We repeated the entire experiment 100 times and averaged the results. The average online error attained by each algorithm for each choice of B is given in Figure 6 and in Table 8.1. The graphs underscore several interesting phenomena. First note that the self-tuned Forgetron and the CG method outperform both the greedy removal Forgetron and the CKS method. In fact, the self-tuned Forgetron and the CG method achieve almost the same accuracy as the vanilla Perceptron algorithm while requiring less than a fifth of the active set size required by the Perceptron. The ability to obtain a low error

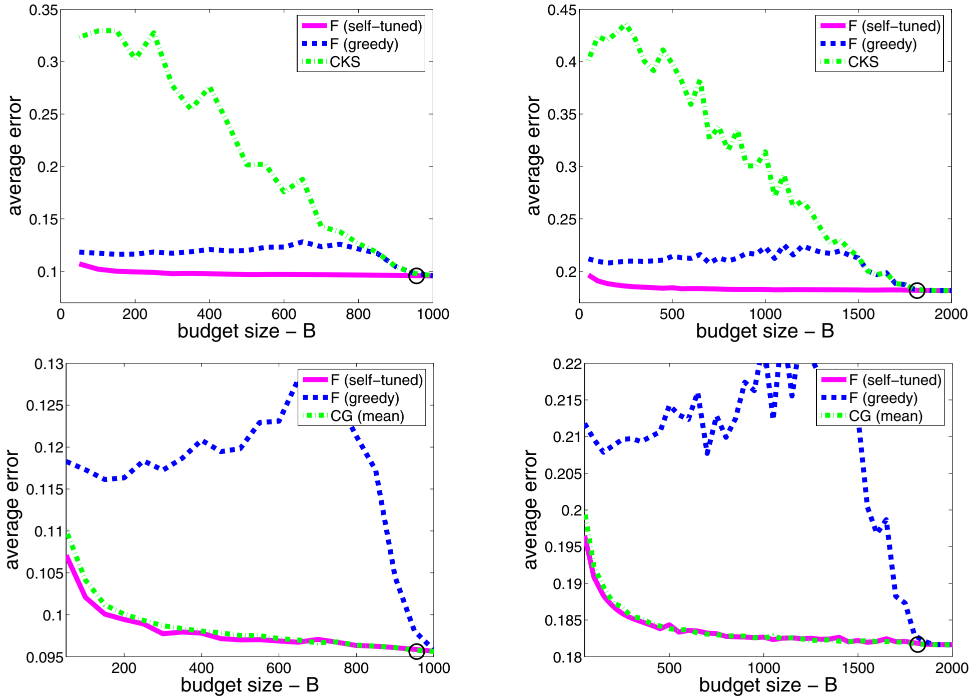


FIG. 6. *The average online error of different budget algorithms as a function of the budget B on synthetic datasets with 5% label noise (left) and 10% label noise (right). The average accuracy of the Perceptron and its budget requirements for each problem are marked by a circle.*

with a small budget on this dataset is not surprising as the decision boundary can be described by a small number of examples. The performance of the greedy removal Forgetron is also reasonable. The inferior performance of the CKS method on these datasets may be attributed to the following observation. A mislabeled example in the active set is likely to decrease the accuracy of the classifier. In addition, if these examples are removed from the active set, they are likely to be incorrectly classified by the resulting classifier. Alas, the removal criterion of the CKS method prefers to leave mislabeled examples in the active set. As mislabeled examples start accumulating in the active set, the damage to the classifier's accuracy becomes more pronounced. In contrast, the Forgetron algorithms demote the weight of each example in the active set on each round, thus ensuring that noisy examples do not remain active for a very long period. The removal criterion of the greedy removal Forgetron algorithm is also affected by the above argument. Indeed, we can see that the performance of the greedy removal Forgetron is rather good with small budgets, it worsens as the budget increases, and finally it improves again when the budget is large. When the budget is very small, the greedy removal Forgetron cannot find an example $r_t \in I_t$ for which $\Psi \leq \frac{15}{32}$. Thus, the example removed is the oldest example in the active set (see (52)). As the budget increases, there are examples whose margins are greater than 1, so the greedy removal Forgetron removes them without further scaling. As with the CKS algorithm, this removal criterion prefers to leave noisy examples in the active set and we can see deterioration in the performance.

So far, we have calculated the average online error of the CG algorithm, and our experiments indicate that it is similar to the online error of the self-tuned Forgetron.

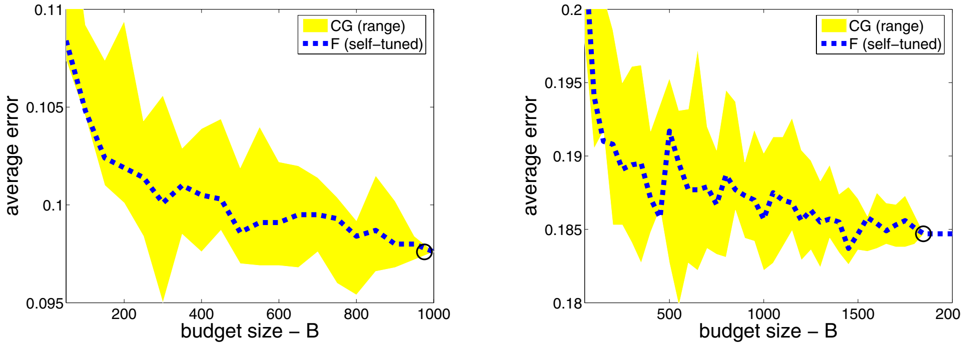


FIG. 7. *The online error of the self-tuned Forgetron and the CG algorithm on a single dataset with* 5% *label noise (left) and on a single dataset with* 10% *label noise. For the CG algorithm, the range of online errors over* 10 *runs is given.*

However, the CG algorithm is a randomized method and its performance on individual runs may vary. Recall that the goal of online learning is to accurately predict a sequence of labels that is revealed incrementally as learning proceeds. Once all of the labels have been revealed, the prediction task becomes vacuous. Therefore, it only makes sense to run the online algorithm over the sequence of examples once. In our last experiment, we compared single runs of the CG algorithm with the self-tuned Forgetron. We ran the self-tuned Forgetron on a single dataset with 5% label noise and on a single dataset with 10% label noise, without averaging the results over several datasets. We also ran the CG algorithm 10 times on each of these datasets. In Figure 7 we give the online error of the self-tuned Forgetron and the range of online errors attained by the CG algorithm. The self-tuned Forgetron outperforms the CG algorithm approximately half of the time, and the performance of the CG algorithm varies significantly from run to run. Therefore, when running the CG algorithm a single time on a given sequence of examples, we can only hope that we are lucky and that its performance is close to average or better. On the other hand, the deterministic self-tuned Forgetron does not suffer from this problem and consistently attains the average accuracy of the CG algorithm.

We conclude this section with a brief discussion of the time complexity of the various algorithms. Let κ denote the time required for a single evaluation of the kernel function. The implementation of the self-tuned Forgetron requires at most B kernel operations on each online round and an additional $O(B)$ operations. Therefore, its total complexity is $O(B\kappa)$ on each round. The complexity of a single round of the CG method is also $O(B\kappa)$. A direct implementation of the greedy removal Forgetron and of the CKS method requires calculating the prediction of the current hypothesis on each example in the active set. The resulting complexity is therefore $O(B^2\kappa)$. A more sophisticated implementation can decrease the number of kernel operations on each online round to be at most B. This can be done by maintaining a matrix with all the kernel evaluations for pairs $\mathbf{x}_i, \mathbf{x}_j$, where $i, j \in I_t$, and updating only a single row and a single column of this matrix on each online round. The resulting complexity of this implementation is $O(B\kappa + B^2)$. However, this implementation requires an additional storage for the $B \times B$ matrix described above.

9. Discussion. We presented a family of kernel-based online classifiers that restrict themselves to a memory of fixed size. The main idea behind our construction is to control the influence that each individual active example has on the online hypothesis. We achieve this control mechanism by repeatedly shrinking the weights that define the online hypothesis. Our shrinking step is done in a way that ensures that an active example can always be removed from the active set without significantly sacrificing classification accuracy.

Our empirical evaluation demonstrates that the gentle shrinking policy employed by the self-tuned Forgetron update significantly outperforms the aggressive shrinking policy of the basic Forgetron algorithm. Moreover, the original Perceptron algorithm, which neither performs any shrinking nor removes active examples, consistently outperforms the Forgetron variants. These observations reinforce our view of the shrinking and removal steps as a type of noise which interferes with the online-learning process. By making this noise as small as possible, we obtain online-learning algorithms that approach the performance of the original Perceptron.

A nice property of this work, also shared by [1], is the way in which theory and practice go hand-in-hand. As mentioned in the introduction of this paper, previous attempts to address the task of online learning on a budget have all lacked a rigorous mathematical justification. In contrast, our algorithm and the algorithm in [1] both entertain formal worst-case guarantees. Our experiments demonstrate that the theoretically motivated algorithms consistently outperform the heuristic approach.

Our experiments suggest that an online kernel method on a memory budget fails when its active set accumulates many noisy active examples. The basic Forgetron and the self-tuned Forgetron avoid this problem by always removing the oldest active example from the active set. This strategy ensures that a noisy active example is removed from the active set after precisely B updates. Even if an adversary creates the sequence of examples, our algorithms cannot be maneuvered into accumulating the noisy examples for a longer number of updates. The CG algorithm [1] exhibits a similar characteristic. Its randomized removal policy always gives an equal probability to removing each active example and therefore cannot be manipulated into accumulating noisy examples. On the other hand, the more sophisticated removal strategies of the CKS algorithm [3] and the greedy Forgetron update can be exploited by an adversary. These algorithms can be tricked into maintaining noisy examples in their memory and discarding informative ones. Our experiments demonstrate that this phenomenon is exhibited even in the case of random label noise, where the input is not controlled by an adversary. This observation sheds a somewhat pessimistic light on the prospects of developing more sophisticated online kernel methods on a budget. It seems that any algorithm that applies a nontrivial removal strategy makes itself vulnerable to manipulation and may be coerced into accumulating noise.

Several interesting open problems remain to be solved. A first challenge is to bridge the gap between the theoretical upper bound of $\sqrt{B+1}$ on the norm of the competitor and

$$U \;=\; \frac{1}{4}\sqrt{\frac{B+1}{\log(B+1)}},$$

achieved by our algorithms. The CG algorithm of [1] managed to close this gap using a randomized algorithm and proving a bound on the *expected* number of mistakes (where expectation is taken over the internal randomization of their algorithm). The question whether there exists a deterministic algorithm which matches the upper bound of $\sqrt{B+1}$ is open.

The intersection of machine learning and computational resource management is a fascinating research field, from both theoretical and practical standpoints. In this paper, we investigated a very simple online-learning scenario, but our construction can be leveraged to solve more complex and realistic problems. For example, one could try to use our framework to devise online algorithms on a memory budget for tasks such as online regression, ranking, and sequence prediction. Another interesting problem is how to train thousands or even millions of online classifiers in parallel, where all of the classifiers share a common global memory of limited size. Rather than just limiting the number of active examples available to each classifier, we would like to dynamically allocate the global memory resource to the various classifiers in a way that would make optimal use of it.

REFERENCES

[1] N. Cesa-Bianchi and C. Gentile, *Tracking the best hyperplane with a simple budget perceptron*, in Learning Theory, Lecture Notes in Comput. Sci. 4005, Springer-Verlag, Berlin, 2006, pp. 483–498.

[2] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, *Online passive aggressive algorithms*, J. Mach. Learn. Res., 7 (2006), pp. 551–585.

[3] K. Crammer, J. Kandola, and Y. Singer, *Online classification on a budget*, in Advances in Neural Information Processing Systems 16, The MIT Press, Cambridge, MA, 2004, pp. 225–232.

[4] O. Dekel, S. Shalev-Shwartz, and Y. Singer, *The Forgetron: A kernel-based perceptron on a fixed budget*, in Advances in Neural Information Processing Systems 18, The MIT Press, Cambridge, MA, 2006, pp. 259–266.

[5] C. Gentile, *A new approximate maximal margin classification algorithm*, J. Mach. Learn. Res., 2 (2001), pp. 213–242.

[6] D. P. Helmbold, J. Kivinen, and M. Warmuth, *Relative loss bounds for single neurons*, IEEE Trans. Neural Networks, 10 (1999), pp. 1291–1304.

[7] J. Kivinen, A. J. Smola, and R. C. Williamson, *Online learning with kernels*, IEEE Trans. Signal Process., 52 (2002), pp. 2165–2176.

[8] J. Kivinen and M. Warmuth, *Exponentiated gradient versus gradient descent for linear predictors*, Inform. and Comput., 132 (1997), pp. 1–64.

[9] Y. Li and P. M. Long, *The relaxed online maximum margin algorithm*, Machine Learning, 46 (2002), pp. 361–387.

[10] F. Rosenblatt, *The perceptron: A probabilistic model for information storage and organization in the brain*, Psychological Review, 65 (1958), pp. 386–407.

[11] V. N. Vapnik, *Statistical Learning Theory*, John Wiley, New York, 1998.

[12] J. Weston, A. Bordes, and L. Bottou, *Online (and offline) on an even tighter budget*, in Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, Barbados, 2005, pp. 413–420.

SIAM J. Comput.
Vol. 37, No. 5, pp. 1373–1386

ON k-D RANGE SEARCH WITH PATRICIA TRIES*

BRADFORD G. NICKERSON[†] AND QINGXIU SHI[†]

Abstract. Patricia tries are explored for indexing combined text and spatial data. A combined text and spatial data range search algorithm is presented for reporting all data from a set of size n intersecting a query hyperrectangle. We also use Patricia tries to answer ϵ-approximate orthogonal range search on a set of n random points and hyperrectangles in k-dimensional data space. ϵ-approximate orthogonal range counting queries can be answered in $O(k \log n/\epsilon^{k-1})$ time, and the number of nodes visited for orthogonal range counting queries is shown to be $O(\log n + k(1+2n^{1/k}\Delta)^{k-1})$ for cubical range of side length Δ. Patricia tries are evaluated experimentally for both orthogonal range search and ϵ-approximate orthogonal range search (for $2 \leq k \leq 14$ and n up to 1,000,000) using uniformly distributed random data. The expected range search time is determined theoretically and found to agree with experimental results.

Key words. orthogonal range search, Patricia trie, approximate range search

AMS subject classifications. 68P05, 68P10, 68W40, 68W25, 68P15, 65D18

DOI. 10.1137/060653780

1. Introduction.

Range search represents an important class of problems that occur in applications of databases, geographical information systems, computer graphics, and computational geometry. Given a collection of n records, each containing multidimensional attributes or keys, a range search asks for all records in the collection with key values each inside specified ranges. Over the past 30 years, more than 60 data structures for the range search problem have been presented [1, 4, 8, 11, 15]. The motivation for this research is to find a dynamic, linear space data structure that supports efficient combined text and spatial data orthogonal range search. Patricia tries are well suited for text indexing and search and seem appropriate for combining text and spatial data in one data structure for efficient range search. This paper explores Patricia tries for a variety of orthogonal range search problems, including those with high dimension k (i.e., $k \approx \log n$).

1.1. Background and previous results.

One of the earliest data structures for solving points in range search problems is the k-dimensional (k-d) tree [3]. Its preprocessing time and storage requirements are $O(n \log n)$ and $O(nk)$, respectively. Lee and Wong [12] have shown that in the worst case its range reporting time is $O(n^{1-1/k}+F)$ (F is the number of points in the range). The range tree was introduced by Bentley and Friedman [4]. It has a good worst-case search time ($O(\log^k n+F)$) but has relatively high preprocessing and storage costs: $O(n \log^{k-1} n)$ and $O(n \log^{k-1} n)$, respectively. Lower bounds for range search were studied by Chazelle [7], who showed that a sequence of n operations for insertion, deletion, and reporting points in a given range costs $\Omega(n(\log n)^k)$, and the worst-case query time for an orthogonal range search data structure, using m units of storage, is $\Omega((\log n/\log(2m/n))^{k-1})$.

To obtain better performance, several researchers turned to an approximate version of the range search problem; instead of counting the points in the exact spec-

*Received by the editors March 6, 2006; accepted for publication (in revised form) June 8, 2007; published electronically January 16, 2008. This research was supported in part by the National Science and Engineering Research Council of Canada.
http://www.siam.org/journals/sicomp/37-5/65378.html

[†]Faculty of Computer Science, University of New Brunswick, Fredericton, NB, E3B 5A3, Canada (bgn@unb.ca, qingxiu.shi@unb.ca).

ified ranges, the data point whose distance to the boundary of the range is within ϵ times the range's diameter may or may not be included in the count. The approximate range search problem was solved optimally by Arya and Mount [2]. With an $O(kn)$-space structure called the balanced box-decomposition tree (which can be constructed in $O(kn \log n)$ time), ϵ-approximate range counting queries can be answered in $O(2^k \log n + (3\sqrt{k}/\epsilon)^k)$ time. If the ranges are convex, the query time can be strengthened to $O(2^k \log n + k^2(3\sqrt{k}/\epsilon)^{k-1})$. They also presented a lower bound of $\Omega(\log n + 1/\epsilon^{k-1})$ for the complexity of answering ϵ-approximate range counting queries assuming a partition tree approach for cubical range in fixed dimension.

The Patricia trie was discovered by Morrison [13]. All nodes in Patricia tries have degree greater than or equal to two by eliminating all one-child internal nodes. Patricia tries are well-balanced trees [18] in the sense that a random shape of Patricia tries resembles the shape of complete balanced trees. Patricia tries can be preprocessed in $O(n \log n)$ time and $O(kn)$ space, and fewer internal nodes are visited for a partial match search of a Patricia trie compared to a k-d trie [10].

1.2. Our results. In section 2 we use binary Patricia tries to represent multidimensional points and hyperrectangles, as well as combined text and spatial data, and we present a range search algorithm for reporting all k-d records from a set of size n intersecting a query hyperrectangle. In section 3 we theoretically analyze the average cost of range search, which is proportional to the number of nodes in the trie visited during the range search. For a cubical range query with side length Δ, the orthogonal range reporting queries visit $O(\log n + k(1 + 2n^{1/k}\Delta)^{k-1})$ nodes. In section 4 we state the problem of ϵ-approximate orthogonal range search and theoretically analyze its cost using Patricia tries. ϵ-approximate orthogonal range counting queries can be answered in $O(k \log n/\epsilon^{k-1})$ time for a cubical query. We show that Patricia tries are appropriate for ϵ-approximate range search and support efficient approximate range search both for k-d points and k-d hyperrectangles. In section 5 we present an extensive experimental study of the practical performance of the Patricia trie using uniform randomly generated spatial data and place names selected from the Canadian geographical names database [19]. We compare the performance of the Patricia trie to the k-d tree, the k-d trie, the $2k$-d trie, the R^*-tree, and the naive method and find that Patricia tries are best when F is small relative to n (e.g., $F \leq \log_2 n$) and $k \leq 10$. The experimental results agree well with the theoretical analysis and show that allowing small errors can significantly improve the query execution time of ϵ-approximate range counting, with a less dramatic effect on the complexity of approximate range reporting.

2. Patricia tries for text and spatial data. Binary tries are data structures which use a binary representation of the key to store keys as a path in the tree. Binary k-d tries use the principle of bit interleaving, e.g., a set of n k-d keys:

$$\begin{gathered} P_1 = (P_{11}, P_{12}, \dots, P_{1k}), \\ \vdots \\ P_n = (P_{n1}, P_{n2}, \dots, P_{nk}), \end{gathered}$$

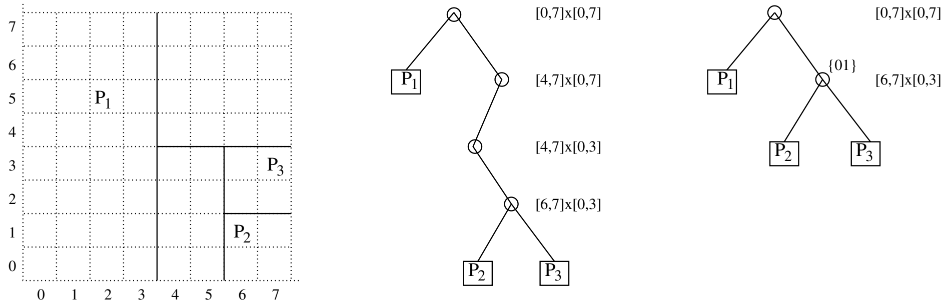
where P_{ij} can be spatial data or text data, $1 \leq i \leq n$ and $1 \leq j \leq k$. Letting $\tilde{P}_{ij}$ be the binary representation of P_{ij}, $\tilde{P}_{ij} \in \{0,1\}^\infty$, we produce one sequence $\tilde{P}_i \in \{0,1\}^\infty$ for each P_i by regular shuffling of the components $\tilde{P}_{i1}, \cdot, \tilde{P}_{ik}$ and use these new composite keys $\tilde{P}_1, \cdot, \tilde{P}_n$ to construct a trie. More precisely, if $\tilde{P_{ij}} = \tilde{P}^0_{ij}\tilde{P}^1_{ij}\tilde{P}^2_{ij}\cdots$, then $\tilde{P_i} = \tilde{P}^0_{i1}\cdots\tilde{P}^0_{ik}\tilde{P}^1_{i1}\cdots\tilde{P}^1_{ik}\tilde{P}^2_{i1}\cdots$, where the superscript indicates the bit position.

We denote by T the Patricia trie constructed by inserting n keys into an initially empty trie. Altogether there are $n-1$ internal nodes and n leaves in T. The skipped bits are stored in an array SKIPSTR, and every leaf is associated with one key.

2.1. Spatial data. We assume our search space is defined on the set of positive integers in k-d space and the space is finite, limited by the number of bits B used to represent an integer in binary, $B = \log_2(\text{MAX - MIN} + 1)$, where MIN and MAX are the whole search space's lower and upper bounds. For example, we have a set of three keys in a 2-dimensional (2-d) space of bits (as shown in Figure 2.1(a)): $P_1 = (2, 5)$, $P_2 = (6, 1)$, and $P_3 = (7, 3)$. We assume $B = 3$, and so we have

$$\begin{aligned} P_1 &= (010, 101) \longrightarrow \tilde{P}_1 = 011001, \\ P_2 &= (110, 001) \longrightarrow \tilde{P}_2 = 101001, \\ P_3 &= (111, 011) \longrightarrow \tilde{P}_3 = 101111. \end{aligned}$$

The thick lines in Figure 2.1(a) represent partitions. The principle of partition is that each partition splits a space into two subspaces of equal size. k-d tries select the attributes to be split cyclically, i.e., $1, \ldots, k, 1, \ldots$. Figure 2.1(b) is a regular 2-d trie built up using the sequences $\tilde{P}_1$, $\tilde{P}_2$, and $\tilde{P}_3$. Removing the one-child internal nodes and storing the skipped information at the internal nodes, we get a 2-d Patricia trie (Figure 2.1(c)). In [5] a binary $2k$-d trie data structure for k-d hyperrectangle range search was investigated.



(a) A 2–d space

(b) Regular 2–d trie

(c) Patricia trie

FIG. 2.1. *A 2-d space with three points and their corresponding tries.*

Each node in k-d tries covers part of the k-d space; that is, every node has a cover space defined as $NC = [\mathcal{L}_1, \mathcal{U}_1] \times [\mathcal{L}_2, \mathcal{U}_2] \times \cdots \times [\mathcal{L}_k, \mathcal{U}_k]$. Arrays $\mathcal{L}$ and $\mathcal{U}$ store the lower and upper bounds of a node's cover space. In Figure 2.1(b) and (c), the list of tuples is the cover space NC of each internal node. The root of k-d tries covers the whole space, and child nodes cover half of the search space volume of their parent. The nodes on level ℓ split attribute $p = (\ell \mod k) + 1$ (at the root, $\ell = 0$). If a node on level ℓ has cover space $[\mathcal{L}_1, \mathcal{U}_1] \times \cdots \times [\mathcal{L}_p, \mathcal{U}_p] \times \cdots \times [\mathcal{L}_k, \mathcal{U}_k]$, then its left child's cover space is $[\mathcal{L}_1, \mathcal{U}_1] \times \cdots \times [\mathcal{L}_p, (\mathcal{L}_p + \mathcal{U}_p)/2] \times \cdots \times [\mathcal{L}_k, \mathcal{U}_k]$, and its right child's cover space is $[\mathcal{L}_1, \mathcal{U}_1] \times \cdots \times ((\mathcal{L}_p + \mathcal{U}_p)/2, \mathcal{U}_p] \times \cdots \times [\mathcal{L}_k, \mathcal{U}_k]$. For k-d Patricia tries, ℓ is not the level of the trie but the length of the path from root to the node plus the length of the skipped bits in the internal nodes along the path. The node cover space must take the skipped bit string stored in the nodes into consideration. For example, in Figure 2.1(c), the node cover space of the root of the 2-d Patricia trie is $[0, 7] \times [0, 7]$ ($\ell = 0$), and the node cover space of its right child (denoted by NC_r) is computed as follows: first, $p = \ell \mod 2 + 1 = 1$, $NC_r = [4, 7] \times [0, 7]$, and $\ell = 1$; then the first bit of the skipped bits string is “0”, which means a left child node has been removed, $p = \ell \mod 2 + 1 = 2$, $NC_r = [4, 7] \times [0, 3]$, and $\ell = 2$; the second and last bit of the skipped bit string is “1”, which means a right child node has been removed, $p = \ell \mod 2 + 1 = 1$, $NC_r = [6, 7] \times [0, 3]$, and $\ell = 3$. So the node cover space of root’s right child is $[6, 7] \times [0, 3]$, as shown in Figure 2.1(c).

2.2. Text data.

Definition 2.1 (numeric mapping [9]). *Assume strings are comprised of symbols drawn from an alphabet of size α, and each symbol is mapped to an integer in the range 0 to $\alpha - 1$. Let a string of length c be $s_1 s_2 \cdots s_c$, with each symbol s_i mapped to an integer t_i. The string s is mapped to $\frac{t_1}{\alpha} + \frac{t_2}{\alpha^2} + \frac{t_3}{\alpha^3} + \cdots + \frac{t_c}{\alpha^c}$, which is a one-to-one mapping.*

For an alphabet of size α, we assign a new decimal value in the range 0 to $\alpha - 1$ for each symbol, different from its decimal value in *ASCII* table, and use $\lceil \log_2 \alpha \rceil$ bits to represent each symbol’s decimal value. For example, assume an alphabet with four characters $\{A, E, I, O\}$. We have $\alpha = 4$ and assign an integer from 0 to 3 to them, respectively, in sequence, and each symbol can be well represented by two bits. Consider the five strings AE, AO, E, I, and O. We use the bit representation of each string to build up the trie. The mapping and the corresponding binary Patricia trie are shown in Figure 2.2.

	binary value	mapping
AE	0001	0/4+1/16=1/16
AO	0011	0/4+3/16=3/16
E	01	1/4
I	10	2/4
O	11	3/4

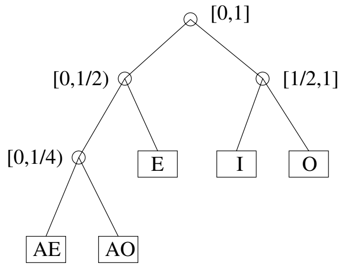


Fig. 2.2. *Mapping strings to rational numbers and the corresponding binary Patricia trie; the tuples beside the internal nodes are the node cover spaces.*

2.3. Combined text and spatial data. We assume each of the coordinate values can be represented in B bits, and the symbols in strings are drawn from an alphabet of size α and each text symbol can be represented in $\lceil \log_2 \alpha \rceil$ bits. For simplicity, we assume the first r dimensions in a k-d key $P = (P_1, P_2, \ldots, P_k)$ are spatial data and the remainder are text data; that is, $P_1, P_2, \ldots, P_r$ are numeric data and $P_{r+1}, \ldots, P_k$ are text data, and $0 \leq r \leq k$. First, we get the bit string of each dimension. If the length of the bit string of the text data P_i is smaller than B, “0” is added at the end of the bit string to extend its length to B, $r + 1 \leq i \leq k$. Then we can use the bit interleaving to get a sequence $\tilde{P}$ of P and insert P using $\tilde{P}$ into the trie T.

Given a query hyperrectangle $W = [L_1, H_1] \times [L_2, H_2] \times \cdots \times [L_k, H_k]$, and $L_i \leq H_i$, a k-d key $P = (P_1, P_2, \ldots, P_k)$ is in range iff $P_i \in [L_i, H_i]\ \forall\, i \in \{1, 2, \ldots, k\}$. We obtain the query hyperrectangles’s cover space $WC = [L_i, H_i]^k$. Each node in T has one of the three color types based on the relation between WC and NC:

1. black: $WC(p) \supseteq NC(p)\ \forall p \in \{1, 2, \dots, k\}$,
2. white: $WC(p) \cap NC(p) = \emptyset\ \exists p \in \{1, 2, \dots, k\}$,
3. grey: all other cases,

where $WC(p)$ is the pth component of the k-d vector WC, and $NC(p)$ is the pth component of the k-d vector NC. The range search algorithm is presented in [17].

2.4. Dynamic operations. Binary Patricia tries support insertion and deletion. Deletion is similar to insertion as described in [17].

3. Orthogonal range search cost. We adapt the approach used in [6] to analyze the range search cost of k-d Patricia tries using Theorem P of [10]. Without loss of generality, the following discussions are all based on unit space $[0, 1]^k$, and we assume the input data and the query hyperrectangle W are drawn from a uniform random distribution.

Given a query $q = (q_1, q_2, \dots, q_k)$ when each q_j can be specified or unspecified (denoted by *), a partial match query returns all records whose attributes coincide with the specified attributes of q. If, e.g., $q = (17, *, *, 30)$, we look for all records whose first attribute is 17 and whose fourth attribute is 30; the second and third attributes are left unspecified. The specification pattern ω of q is a word in $\{S, *\}^k$, where $\omega_j = S$ if q_j is specified and $\omega_j = *$ if q_j is unspecified; in our example we have the specification pattern $S * *S$. Partial match queries make sense if at least one attribute of the query is specified and at least one attribute is not. The analysis of the average cost of partial match queries in k-d Patricia tries was addressed by Kirschenhofer and Prodinger [10]. We restate their theorem as follows.

Theorem 3.1. *Given a Patricia trie T built from n k-d keys and a partial match query of specification pattern ω, let $S \subset \{1, 2, \dots, k\}$ be the set of specified coordinates. The average cost of a partial match query measured by the number of nodes traversed in T is*

$$Q_S(n, k) \sim n^{1-\frac{s}{k}} \left(\frac{(\frac{s}{k}+1)(1-2^{-s/k})}{k \log 2} \frac{\Gamma(\frac{s}{k})}{1-\frac{s}{k}} \sum_{j=0}^{k-1} (\delta_1\, \delta_2\, \cdots\, \delta_j) 2^{-j(1-s/k)} + \beta(\log_2 n^{1/k}) \right),$$

where s is the number of specified attributes in ω, $1 \le s \le k-1$, $\delta_j = 1$ if the jth attribute of ω is specified and $\delta_j = 2$ if it is unspecified, and $\beta(x)$ is a continuous periodic function with mean zero and small amplitude.

The following proposition relates the performance of range searches with the performance of partial match queries.

Proposition 3.2. *Given a Patricia trie T built from n k-d keys and a partial match query of specification pattern ω, let $S \subset \{1, 2, \dots, k\}$ be the set of specified coordinates. The average cost of a partial match query measured by the number of nodes traversed in T is*

$$Q_S(n, k) = E\left\{ \sum_{t=1}^{2n-1} \prod_{p \in S} |NC_t(p)| \right\},$$

where $|NC_t(p)|$, $1 \le p \le k$, are the cover spaces of node t in T.

Proof. If a node is visited, $q_p \in NC(p) = [\mathcal{L}_p, \mathcal{U}_p]\ \forall p \in S$. The probability that a node in trie T will be visited is determined by the volume of every node's cover space in the space $[0, 1]$. □

We use the probabilistic model of random range queries introduced in [6]. A range query is a k-d hyperrectangle $W = [L_1, H_1] \times [L_2, H_2] \times \cdots \times [L_k, H_k]$ with $0 \leq L_i \leq H_i \leq 1$ for $1 \leq i \leq k$. To get the color types for a node in the trie, we compare all the k ranges of WC with NC. In our range search algorithm [14], the range search proceeds from the root to the leaves. On each level, we do at least one comparison of the k ranges and store the color as the node's state. Traversing stops on paths when we meet with black or white nodes and continues when grey nodes are encountered and continues to collect black nodes in the subtree of the black nodes we first meet. The time complexity of range search is proportional to the number of grey nodes (GN) and black nodes (BN) visited in the trie built from the input data. We have the following equation:

$$Q(n,k) = \sum_{t=1}^{2n-1} 1_{[node_t \in GN \cup BN]},$$

where we use $1_{[A]}$ as the indicator variable for the event A.

LEMMA 3.3. $\sum_{t=1}^{2n-1} \prod_{p=1}^{k} |NC_t(p)| \leq 1 + \log_2 n$.

Proof. We denote the volume of the node t in the Patricia trie T as $|NC_t|$, and $|NC_t| = \prod_{p=1}^{k} |NC_t(p)|$. If there are no skipped bits in the root, $|NC_t| = 1$. As the level ℓ visited in T increases, the value of $|NC_t|$ decreases. $\sum_{t=1}^{2n-1} \prod_{p=1}^{k} |NC_t(p)|$ is maximal when every level except possibly the deepest in T is completely filled, and there is no skipped bit string in any internal node. Then we have $\sum_{t=1}^{2n-1} \prod_{p=1}^{k} |NC_t(p)| \leq 1 + 2 \times \frac{1}{2} + 4 \times \frac{1}{4} + 8 \times \frac{1}{8} + \cdots + 2^{\lceil \log_2 n \rceil} \frac{1}{2^{\lceil \log_2 n \rceil}} \leq 1 + \log_2 n$. ∎

THEOREM 3.4. *Given a Patricia trie T built from n random k-d data points uniformly distributed on $[0,1]^k$, and given a query hyperrectangle W with side lengths $\Delta_1 \ldots \Delta_k$ and with a center Z which is uniformly distributed on $[0,1]^k$, the orthogonal range reporting query is expected to visit*

$$O\left(n \prod_{p=1}^{k} \Delta_p + \log n + \sum_{s=1}^{k-1} \left(\sum_{S \subset \{1,\ldots,k\}, |S|=s} \gamma(k,S) \prod_{p \notin S} \Delta_p \right) n^{1-\frac{s}{k}} \right)$$

nodes in T, where $\gamma(d,S) = \frac{(\frac{s}{k}+1)(1-2^{-s/k})}{k \log 2} \frac{\Gamma(\frac{s}{k})}{1-\frac{s}{k}} \sum_{j=0}^{k-1} (\delta_1\, \delta_2 \cdots \delta_j) 2^{-j(1-s/k)}$ *with* $s = |S|$, $\delta_j = 1$ *if* $j \in S$, *and* $\delta_j = 2$ *if* $j \notin S$.

Proof. $E\{Q(n,k)\} = E\{\sum_{t=1}^{2n-1} 1_{[node_t \in GN \cup BN]}\}$. This calculation includes the reporting time for collection of the subtree of black nodes which arises during the traversal. The probability that a node t in the Patricia trie T is black or grey is given as $\Pr(node_t \in GN \cup BN) \leq \prod_{p=1}^{k} (|NC(p)| + \Delta_p)$. The probability for query hyperrectangle W intersecting a node's cover space NC_t is the probability that Z_j, the center of W, is within the distance $\frac{\Delta_j}{2}$ of $NC(j)$. This probability is bounded by the volume of NC_t expanded by Δ_j in the jth dimension $\forall j \in \{1,\ldots,k\}$. We have

$$\begin{aligned} E\{Q(n,k)\} &\leq E\left\{ \sum_{t=1}^{2n-1} \prod_{p=1}^{k} (\Delta_p + |NC_t(p)|) \right\} \\ &= \sum_{S \subseteq \{1,\ldots,k\}} \left(\prod_{p \notin S} \Delta_p \right) E\left\{ \sum_{t=1}^{2n-1} \prod_{p \in S} |NC_t(p)| \right\} \end{aligned}$$

$$\begin{aligned}
&= \sum_{S=\emptyset} \left(\prod_{p \notin S} \Delta_p \right) E \left\{ \sum_{t=1}^{2n-1} \prod_{p \in S} |NC_t(p)| \right\} \\
&\quad + \sum_{S=\{1,\dots,k\}} \left(\prod_{p \notin S} \Delta_p \right) E \left\{ \sum_{t=1}^{2n-1} \prod_{p \in S} |NC_t(p)| \right\} \\
&\quad + \sum_{S \subset \{1,\dots,k\}, 1 \le |S| \le k-1} \left(\prod_{p \notin S} \Delta_p \right) E \left\{ \sum_{t=1}^{2n-1} \prod_{p \in S} |NC_t(p)| \right\} \\
&= (2n-1) \prod_{p=1}^{k} \Delta_p + \ E \left\{ \sum_{t=1}^{2n-1} \prod_{p=1}^{k} |NC_t(p)| \right\} \\
&\quad + \sum_{S \subset \{1,\dots,k\}, 1 \le |S| \le k-1} \left(\prod_{p \notin S} \Delta_p \right) E \left\{ \sum_{t=1}^{2n-1} \prod_{p \in S} |NC_t(p)| \right\}.
\end{aligned}$$

Using Proposition 3.2, we obtain

$$\begin{aligned}
E\{Q(n,k)\} \le (2n-1) \prod_{p=1}^{k} \Delta_p + E \left\{ \sum_{t=1}^{2n-1} \prod_{p=1}^{k} |NC_t(p)| \right\} \\
+ \sum_{S \subset \{1,\dots,k\}, 1 \le |S| \le k-1} Q_S(n,k) \prod_{p \notin S} \Delta_p.
\end{aligned}$$

The results follow by Theorem 3.1 and Lemma 3.3. □

COROLLARY 3.5. *Given a Patricia trie T built from n random k-d data points and a query hypercube W with side length Δ, the orthogonal range reporting query is expected to visit*

$$O(n\Delta^d + \log n + d(1 + 2n^{1/d}\Delta)^{d-1})$$

nodes in T.

Proof. When the query hyperrectangle W is a hypercube, i.e., $\Delta_1 = \Delta_2 = \cdots = \Delta_k = \Delta$, we have

$$\gamma(k,S) = \frac{(\frac{s}{k}+1)(1-2^{-s/k})}{k \log 2} \frac{\Gamma(\frac{s}{k})}{1-\frac{s}{k}} \sum_{j=0}^{k-1} (\delta_1\, \delta_2\, \cdots\, \delta_j) 2^{-j(1-s/k)}.$$

Assuming the worst case where the first $(k-s)$ dimensions are unspecified (i.e., $\delta_i = 2$ $\forall i \in \{1,\dots,k-s\}$), we have

$$\begin{aligned}
\gamma(k,S) &\le \frac{(\frac{s}{k}+1)(1-2^{-s/k})}{k \log 2} \frac{\Gamma(1+\frac{s}{k})}{(1-\frac{s}{k})\frac{s}{k}} \left(\sum_{j=0}^{k-s} 2^j 2^{-j(1-s/k)} + \sum_{j=k-s+1}^{k-1} 2^{k-s} 2^{-j(1-s/k)} \right) \\
&= \frac{(1-2^{-s/k})\Gamma(2+\frac{s}{k})}{s(1-\frac{s}{k})\log 2} \frac{2^{s/k} - 2^{(k-s+1)s/k}}{(2^{s/k}-1)(2^{s/k}-2)} \\
&= \frac{(1-2^{s(1-s/k)})\Gamma(2+\frac{s}{k})}{s(1-\frac{s}{k})(2^{s/k}-2)\log 2}
\end{aligned}$$

$$= \frac{(2^{1-s/k} + 2^{2(1-s/k)} + \cdots + 2^{s(1-s/k)})\Gamma(2+\frac{s}{k})}{s(1-\frac{s}{k})2\log 2}$$
$$\leq \frac{k2^{s(1-s/k)}}{(k-s)\log 2},$$

since $1 \leq \Gamma(2+\frac{s}{k}) < 2$. Now we can write the third part of Theorem 3.4 as

$$\sum_{s=1}^{k-1} \sum_{S\subset\{1,\ldots,k\},|S|=s} \gamma(n,S)n^{1-\frac{s}{k}}\Delta^{k-s}$$
$$\leq \sum_{s=1}^{k-1} C(k,s)\frac{k2^{s(1-s/k)}}{(k-s)\log 2}(n^{1/k}\Delta)^{k-s}$$
$$= \sum_{s=1}^{k-1} C(k-1,s-1)\frac{k^2}{s(k-s)\log 2}(2^{s/k}n^{1/k}\Delta)^{(k-1)-(s-1)}$$
$$\leq k((1+2n^{1/k}\Delta)^{k-1}-1),$$

where $C(k,s)$ denotes the number of ways to choose an s-element subset from a k-element set. The last step in the proof uses the binomial theorem. The result follows from Theorem 3.4. □

For orthogonal range counting queries, the term $n\Delta^k$ in reporting query time disappears, leaving $E\{Q(n,k)\} = O(\log n + k(1+2n^{1/k}\Delta)^{k-1})$.

4. Approximate orthogonal range search. Given n k-d points and a k-d query hyperrectangle $W = [L_1, H_1]\times[L_2, H_2]\times\cdots\times[L_k, H_k]$, and $L_i \leq H_i$ with center $Z = (\frac{L_1+H_1}{2}, \frac{L_2+H_2}{2}, \ldots, \frac{L_k+H_k}{2})$, the ϵ-approximate orthogonal range query counts (or reports) points in the query hyperrectangle, allowing errors near the boundary of the query hyperrectangle. The edges of W have given lengths $\Delta_1, \Delta_2, \ldots, \Delta_k$, where $\Delta_i = H_i - L_i$ $\forall i \in \{1, 2, \ldots, k\}$. We define $[\mathrm{MIN}_i, \mathrm{MAX}_i]$ $\forall i \in \{1, 2, \ldots, k\}$ as the minimum and maximum possible data coordinate values for dimension i. Given $0 \leq \epsilon \leq 0.5$, let $W^- = [L_1 + \Delta_1\epsilon, H_1 - \Delta_1\epsilon] \times [L_2 + \Delta_2\epsilon, H_2 - \Delta_2\epsilon] \times \cdots \times [L_k + \Delta_k\epsilon, H_k - \Delta_k\epsilon]$ be the k-d inner query hyperrectangle with center at Z, and let $W^+ = [L_1 - \Delta_1\epsilon, H_1 + \Delta_1\epsilon] \times [L_2 - \Delta_2\epsilon, H_2 + \Delta_2\epsilon] \times \cdots \times [L_k - \Delta_k\epsilon, H_k + \Delta_k\epsilon]$ be the k-d outer query hyperrectangle with center at Z. (We assume $\mathrm{MIN}_i \leq L_i - \Delta_i\epsilon$ and $H_i + \Delta_i\epsilon \leq \mathrm{MAX}_i$ $\forall i \in \{1, 2, \ldots, k\}$.) For an ϵ-approximate range search query, points inside W^- must be counted, points outside W^+ must not be counted, and points between W^- and W^+ may be miscounted (see Figure 4.1). The approximate orthogonal range search algorithm is presented in [16].

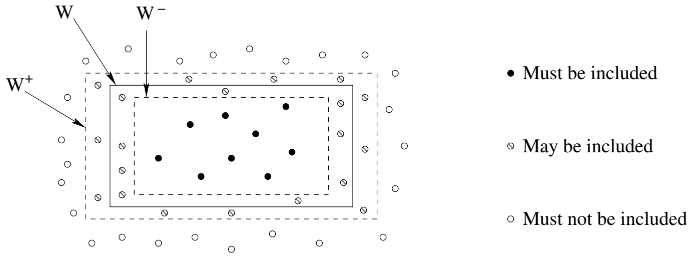


FIG. 4.1. *Approximate orthogonal range search queries.*

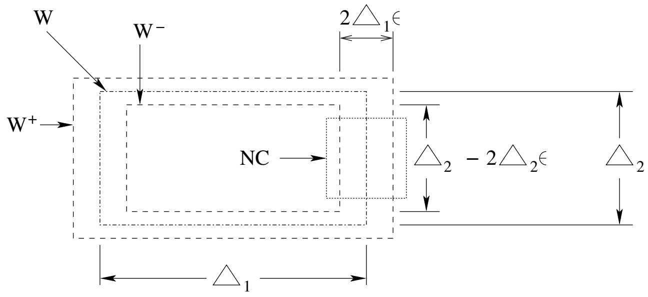


FIG. 4.2. *An illustration of a node in T with cover space NC intersecting both the* 1*-facet of* W^- *and the* 1*-facet of* W^+*;* $\epsilon = 0.1$.

THEOREM 4.1. *Given a Patricia trie T built from n random k-d data points and a random query hyperrectangle W with side lengths $\Delta_1 \dots \Delta_k$, and $0 < \epsilon \leq 0.5$, ϵ-approximate range counting queries visit*

$$O\left(\log n \sum_{p=1}^{k} \left(\prod_{i=1, i\neq p}^{k} \left(2 + \frac{\Delta_i}{\Delta_p}\left(\frac{1}{\epsilon} - 2\right)\right)\right)\right)$$

nodes in T.

Proof. A node is said to be expanded if the algorithm visits the children of this node. For a node to be expanded, its node cover space must intersect with both the inner query hyperrectangle W^- and the outer query hyperrectangle W^+. We call the facet of the query hyperrectangle perpendicular to the pth orthogonal axis the p-facet. According to the definition of W^- and W^+, the p-facets of W^- and W^+ are separated from each other at least by a distance of $2\Delta_p\epsilon\ \forall p \in \{1, \dots, k\}$. So a node in T with $|NC(p)| < 2\Delta_p\epsilon\ \forall p \in \{1, \dots, k\}$ is never expanded in the approximate range search algorithm, where $|NC(p)|$ is the length of the pth side of node cover space NC.

Each partition of T splits a region of the search space into two equal subregions. Each coordinate axis gets cut in turn, in a cyclical fashion of $1, 2, \dots, k, 1, 2, \dots$, which results in regions such that the length of the longest side is equal to or twice that of the smallest side, and $|NC(1)| \leq |NC(2)| \leq \cdots \leq |NC(k)|$. Assume a node in T intersects both the p-facet of W^- and the p-facet of W^+, and $1 \leq p \leq k$; then $|NC(p)| \geq 2\Delta_p\epsilon$, $|NC(i)| \geq \Delta_p\epsilon\ \forall i \in \{1, \dots, p-1\}$, and $|NC(j)| \geq 2\Delta_p\epsilon\ \forall j \in \{p+1, \dots, k\}$ (see Figure 4.2). So there are at most

$$\left(\prod_{i=1}^{p-1}\left(1 + \left\lceil \frac{\Delta_i - 2\Delta_i\epsilon}{\Delta_p\epsilon} \right\rceil\right)\right)\left(\prod_{j=p+1}^{k}\left(1 + \left\lceil \frac{\Delta_j - 2\Delta_j\epsilon}{2\Delta_p\epsilon} \right\rceil\right)\right)$$

regions intersecting with both the p-facet of W^- and the p-facet of W^+. Each k-dimensional hyperrectangle has $2k$ facets, and so altogether there are at most

$$2\sum_{p=1}^{k}\left(\prod_{i=1}^{p-1}\left(1 + \left\lceil \frac{\Delta_i - 2\Delta_i\epsilon}{\Delta_p\epsilon} \right\rceil\right)\right)\left(\prod_{j=p+1}^{k}\left(1 + \left\lceil \frac{\Delta_j - 2\Delta_j\epsilon}{2\Delta_p\epsilon} \right\rceil\right)\right)$$

$$\leq 2\sum_{p=1}^{k}\left(\prod_{i=1}^{p-1}\left(2+\frac{\Delta_i - 2\Delta_i\epsilon}{\Delta_p\epsilon}\right)\right)\left(\prod_{j=p+1}^{k}\left(2+\frac{\Delta_j - 2\Delta_j\epsilon}{2\Delta_p\epsilon}\right)\right)$$
$$\leq 2\sum_{p=1}^{k}\left(\prod_{i=1,i\neq p}^{k}\left(2+\frac{\Delta_i - 2\Delta_i\epsilon}{\Delta_p\epsilon}\right)\right)$$

regions overlapping both W^- and W^+. The depth of the Patricia trie is $O(\log n)$ [18], and so we reach the desired result. □

COROLLARY 4.2. *Given a Patricia trie T built from n random k-d data points and a random query hypercube W, and $0 < \epsilon \leq 0.5$, ϵ-approximate range counting queries visit $O(k \log n/\epsilon^{k-1})$ nodes in T.*

Besides the number of nodes visited in the approximate range counting algorithm, up to $2F$ additional nodes are visited in answering the approximate range search reporting queries [16], where F is the number of points in range.

5. Experiments.

We have conducted a series of experiments to validate the theoretical analysis of sections 3 and 4. Our experiments were performed using uniformly and randomly distributed data from the interval $[0, 1]$ for ϵ ranging from 0 to 0.5, $2 \leq k \leq 14$, and n up to 1,000,000. The programs were run on a Sun Microsystems V880 with four 1.2 GHz UltraSPARC III processors, 16 GB of main memory, and running Solaris 8. Each experimental point in the following graphs was done with an average of 300 test cases.

5.1. *k*-d points.

We rearranged the theoretical result in Theorem 3.4 for query hypercubes with side length Δ and obtained the equation

$$Q = n\Delta^k + \log n + \sum_{s=1}^{k-1}\left(\sum_{S\subset\{1,\ldots,k\},|S|=s}\gamma(k,S)\right)\Delta^{k-s}n^{1-s/k}.$$

We compared the experimental results of range reporting queries using Patricia tries to the value of Q and found that they are consistent when $2 \leq k \leq 14$ and $n = 1{,}000{,}000$ (see Figure 5.1), where $\Delta = vol^{1/k}$.

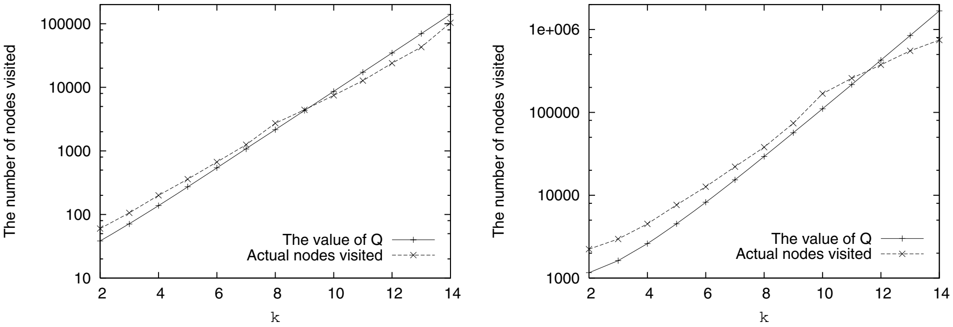
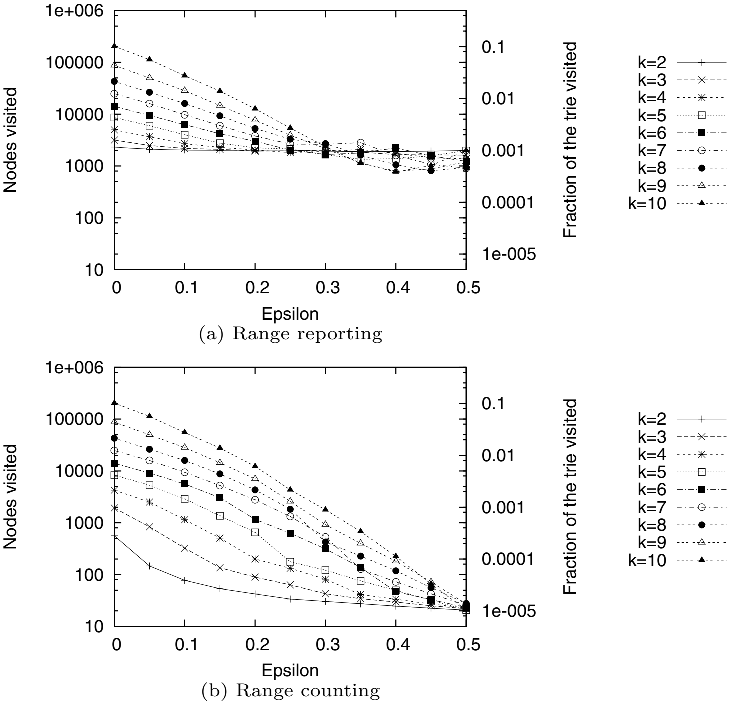


FIG. 5.1. *The experimental and theoretical number of nodes visited for Patricia trie range search for the volume of the query hypercubes (left) $vol = 0.00001$ and (right) $vol = 0.001$, where $n = 1{,}000{,}000$ and $2 \leq k \leq 14$.*

The experimental results for approximate range reporting and counting queries with k-d query hypercube volume of 0.001 for Patricia tries are shown in Figure 5.2.



(a) Range reporting

(b) Range counting

FIG. 5.2. *Number of nodes visited versus ϵ in Patricia trie for k-d points with k-d query hypercube volume of* 0.001 *(*n = 1,000,000*).*

The corresponding time for range search is directly proportional to the number of nodes visited. We find that there are significant improvements when ϵ grows from 0 to 0.5. As ϵ increases, the numbers of nodes visited tend to converge, irrespective of k. Figure 5.2 shows that the average improvement of the approximate range counting queries when $\epsilon = 0.05$ and $k \leq 5$ is more dramatic than that of the range reporting queries.

We define the fraction of points miscounted in approximate range search as $\delta_{\epsilon=x} = \frac{|F_{\epsilon=x}-F_{\epsilon=0}|}{F_{\epsilon=0}}$, where $F_{\epsilon=x}$ is the number of points in range for an ϵ-approximate range query when $\epsilon = x$, and $F_{\epsilon=0}$ is the number of points in the exact range query. The experimental results show that when $k = 2$, $n =$ 1,000,000, and $vol = 0.001$, $\delta_{\epsilon=x}$ ranges from 0.002 ($\epsilon = 0.05$) to 0.05 ($\epsilon = 0.5$) [16], and so relatively few points are miscounted.

5.2. *k*-d hyperrectangles. Data hyperrectangle centers are uniformly distributed in $[0,1]^d$ and the lengths of their sides uniformly and independently distributed between 0 and $maxsize$. Range search reports hyperrectangles that intersect with the query hyperrectangle. We show the results of experiments with k-d query hypercubes with volumes that range from 0.01% to 0.1% of the total space for Patricia tries for k-d rectangles in Figure 5.3, in comparison with the R*-tree (the maximum number of children $M = 10$), the 2k-d trie, and the naive method. When $maxsize = 0.01$ and the query hypercube volume is 0.0001, the Patricia trie has a speed-up factor between 1.2 and 2.6 over the R*-tree and between 3.0 and 4.6 over the 2k-d trie. With increasing volume of the query hypercube, the Patricia trie has a speed-up factor between

4.4 and 5.8 over the $2k$-d trie. The R*-tree is best when $k \geq 9$. The Patricia trie not only needs less space than the $2k$-d trie but spends less time on range search. The experimental results of ϵ-approximate range search are presented in [16]. An average of 40% fewer nodes are visited for the Patricia trie when $0.05 \leq \epsilon \leq 0.5$, compared to the exact range counting queries.

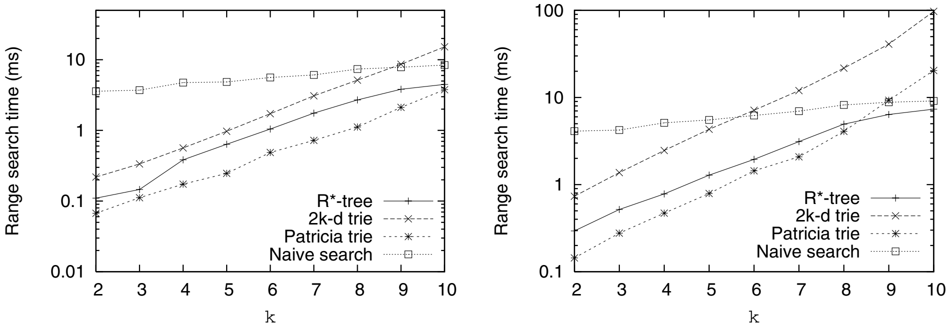


FIG. 5.3. *The range search time in milliseconds for the R*-tree, the Patricia trie, the* $2k$*-d trie, and the naive search for k-d query hypercube sizes of (left)* 0.01% *and (right)* 0.1% *of total space volume for maxsize* $= 0.01$ *(*$n = 100{,}000$ *and* $2 \leq k \leq 10$*).*

5.3. Combined text and spatial data. We tested the Patricia tries with real-life text data (names randomly chosen from the Canadian geographical names database [19]). We used kr to denote the number of dimensions of the text data; i.e., given a k-d key $v = (v_1, v_2, \ldots, v_k)$, $kr = 1$ means that one attribute in v is the text string, and the remaining $k - 1$ attributes are the point data. Experimental results are shown in Figure 5.4 for the range search time in the k-d tree, the Patricia trie, the ternary search tree (TST), and the naive search for $E(F) = \log_2 n$. When $kr = 1$ and $k < 12$, the Patricia trie has a speed-up factor up to 3.2 over the k-d tree, and 3.3 over the TST. When $kr = 3$, the Patricia is up to 2.9 times faster than the k-d tree and 3.1 times faster than the TST. The naive search method is best when $k \geq 12$.

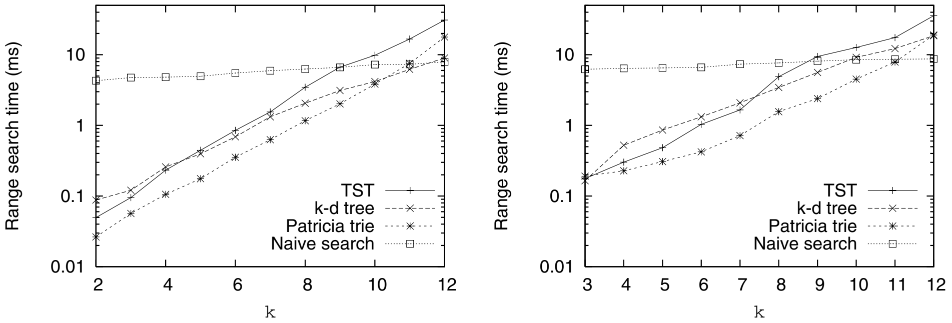


FIG. 5.4. *Range search time for the k-d tree, the Patricia trie, the TST, and the naive method for* $n = 100{,}000$ *k-d combined textual and spatial data, where (left)* $kr = 1$ *and (right)* $kr = 3$*. (*$E(F) = \log_2 n$*. The textual data are chosen from the Canadian geographical names database* [19]*.)*

6. Conclusions and open questions. We have shown that Patricia trie is a good linear space data structure supporting dynamic operations for a variety of orthogonal range search problems. Using Patricia tries for combined text and spatial data search is straightforward as we treat text data as additional $(k-r)$ dimensions. Our expected time analysis of range search for the k-d Patricia trie shows that we expect to visit $O(F+\log n+k(1+2n^{1/k}\Delta)^{k-1})$ nodes for cubical range queries of side length Δ. Experimental results show that the Patricia trie outperforms the k-d tree when $F \le \log_2 n$ and $k < \log n$ for uniform random k-d points and random k-d hyperrectangle queries. We also show that Patricia tries can be used to answer ϵ-approximate orthogonal range counting queries visiting $O(k \log n/\epsilon^{k-1})$ nodes for cubical range queries. Can the ϵ-approximate range search result be improved closer to the lower bound of $\Omega(\log n + 1/\epsilon^{k-1})$ in fixed dimension? Experimental results show that if we allow small relative errors, the number of nodes visited for range counting can be reduced on average by 1/2 for query hypercubes with volumes of 0.001, $\epsilon = 0.05$, $2 \le k \le 10$, and $n =$ 1,000,000. Range reporting queries have a less dramatic improvement, because of additional $O(F)$ nodes visited, which is the dominating term in the number of the nodes visited during range reporting. It is an open question how well this approach works when the text search dominates (i.e., has more dimensions than the spatial data). How will the Patricia trie behave under different distributions of the input data and the query hyperrectangles?

REFERENCES

[1] P. Agarwal, *Range searching*, in Handbook of Discrete and Computational Geometry, CRC Press, Boca Raton, FL, 1997, pp. 575–598.

[2] S. Arya and D. Mount, *Approximate range searching*, Comput. Geom., 17 (2000), pp. 135–163.

[3] J. Bentley, *Multidimensional binary search trees for associative searching*, Comm. ACM, 18 (1975), pp. 509–517.

[4] J. Bentley and J. Friedman, *Data structures for range searching*, Comput. Surveys, 11 (1979), pp. 397–409.

[5] L. Bu and B. Nickerson, *Multidimensional orthogonal range search using tries*, in Proceedings of the 15th Canadian Conference on Computational Geometry, Dalhousie University, Halifax, NS, Canada, 2003, pp. 161–165.

[6] P. Chanzy, L. Devroye, and C. Zamora-Cura, *Analysis of range search for random k-d trees*, Acta Inform., 37 (2001), pp. 355–383.

[7] B. Chazelle, *Lower bounds for orthogonal range search:* II. *The arithmatic model*, J. Assoc. Comput. Mach., 37 (1990), pp. 439–463.

[8] V. Gaede and O. Gunther, *Multidimensional access methods*, ACM Comput. Surveys, 30 (1998), pp. 170–231.

[9] H. Jagadish, N. Koudas, and D. Srivastava, *On effective multi-dimensional indexing for strings*, in Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, TX, 2000, pp. 403–414.

[10] P. Kirschenhofer and H. Prodinger, *Multidimensional digital searching—alternative data structures*, Random Structures Algorithms, 5 (1994), pp. 123–134.

[11] D. Knuth, *The Art of Computer Programming: Sorting and Searching*, Vol. 3, 2nd ed., Addison–Wesley, Reading, MA, 1998.

[12] D. Lee and C. Wong, *Worst-case analysis for region and partial region searches in multidimensional binary search trees and balanced quad trees*, Acta Informat., 9 (1977), pp. 23–29.

[13] D. Morrison, *Patricia - practical algorithm to retrieve information coded in alphanumeric*, J. Assoc. Comput. Mach., 14 (1968), pp. 514–534.

[14] B. G. Nickerson and Q. Shi, *Tries for combined text and spatial data range search*, in Proceedings of the 16th Canadian Conference on Computational Geometry, Concordia University, Montreal, QC, Canada, 2004, pp. 149–153.

[15] H. Samet, *Foundations of Multidimensional and Metric Data Structures*, 1st ed., Morgan Kaufmann, San Francisco, 2006.

[16] Q. Shi and B. G. Nickerson, *Approximate Orthogonal Range Search Using Patricia Tries*, Technical report TR05-172, Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada, 2005.

[17] Q. Shi and B. G. Nickerson, *k-d Range Search with Binary Patricia Tries*, Technical report TR04-168, Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada, 2004.

[18] W. Szpankowski, *Patricia tries again revisited*, J. Assoc. Comput. Mach., 37 (1990), pp. 691–711.

[19] Geographical Names Board of Canada, *Principles and Procedures for Geographical Naming*, http://geonames.rncan.gc.ca/ (2001).

SIAM J. COMPUT.
Vol. 37, No. 5, pp. 1387–1400

QUANTUM PROPERTY TESTING*

HARRY BUHRMAN[†], LANCE FORTNOW[‡], ILAN NEWMAN[§], AND HEIN RÖHRIG[¶]

Abstract. A language L has a property tester if there exists a probabilistic algorithm that given an input x queries only a small number of bits of x and distinguishes the cases as to whether x is in L and x has large Hamming distance from all y in L. We define a similar notion of quantum property testing and show that there exist languages with good quantum property testers but no good classical testers. We also show there exist languages which require a large number of queries even for quantumly testing.

Key words. quantum computing, property testing

AMS subject classification. 68Q10

DOI. 10.1137/S0097539704442416

1. Introduction.

Suppose we have a large data set, for example, a large chunk of the World Wide Web or a genomic sequence. We would like to test whether the data has a certain property, but we may not have the time to look at the entire data set or even a large portion of it.

To handle these types of problems, Rubinfeld and Sudan [35] and Goldreich, Goldwasser, and Ron [25] have developed the notion of property testing. Testable properties come in many varieties including graph properties, e.g., [25, 4, 20, 22, 1, 27, 33, 26, 6, 8, 7], algebraic properties of functions [13, 35, 18], Boolean functions and languages [5, 21], and geometric objects [3, 16]. Nice surveys in this area can be found in [34, 19].

In this model, the property tester has random access to the n input bits similar to the black-box oracle model. The tester can query only a small number of input bits; the set of indices is usually of constant size and chosen probabilistically. Clearly, we cannot determine from this small number of bits whether the input sits in some language L. However, for many languages we can distinguish the case that the input is in L from the case that the input differs from all inputs in L of the same length by some constant fraction of input bits. Note that we do not consider the time complexity of the testing algorithms in this paper.

Since there are many examples where quantum computation gives us an advantage over classical computation [12, 37, 36, 28], one may naturally ask whether using

*Received by the editors March 23, 2004; accepted for publication (in revised form) June 19, 2007; published electronically January 16, 2008. This research was done while all authors were visiting the NEC Research Institute. A preliminary version of this paper appeared in *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003, pp. 480–488.

http://www.siam.org/journals/sicomp/37-5/44241.html

[†]Centrum voor Wiskunde en Informatica (CWI), NL-1098 SJ Amsterdam, The Netherlands, and Faculty of Science, University of Amsterdam, NL-1098 SM Amsterdam, The Netherlands (buhrman@cwi.nl). This author's research was partially supported by the EU fifth framework project QAIP, IST-1999-11234.

[‡]NEC Research Institute, Princeton, NJ 08540. Current address: Department of Computer Science, University of Chicago, Chicago, IL 60637 (fortnow@cs.uchicago.edu).

[§]Department of Computer Science, Haifa University, 31905 Haifa, Israel (ilan@cs.haifa.ac.il). This author's research was partially supported by the Israeli Science Foundation (ISF) 55/03.

[¶]Centrum voor Wiskunde en Informatica (CWI), NL-1098 SJ Amsterdam, The Netherlands. Current address: Google, Inc., Mountain View, CA 94043 (mail@hein.roehrig.name). This author's research was partially supported by the EU fifth framework project QAIP, IST-1999-11234.

quantum computation may lead to better property testers. By using the quantum oracle-query model developed by Beals et al. [10], we can easily extend the definitions of property testing to the quantum setting.

Beals et al. [10] have shown that for all total functions we have a polynomial relationship between the number of queries required by quantum machine and that needed by a classical machine. For greater separations one needs to impose a promise on the input. The known examples, such as those due to Simon [37] and Bernstein and Vazirani [12], require considerable structure in the promise. Property testing amounts to the natural promise of either being in the language or far from each input in the language. This promise would seem to have too little structure to give a separation, but in fact we can prove that quantum property testing can greatly improve on classical testing.

We show that every subset of Hadamard codes has a quantum property tester with O(1) queries and that most subsets would require $\Theta(\log n)$ queries to test with a probabilistic tester. This shows that indeed quantum property testers are more powerful than classical testers. Moreover, we also give an example of a language where the quantum tester is exponentially more efficient.

Beals et al. [10] observed that every k-query quantum algorithm gives rise to a degree-$2k$ polynomial in the input bits, which gives the acceptance probability of the algorithm; thus, a quantum property tester for P gives rise to a polynomial that is on all binary inputs between 0 and 1, that is, at least 2/3 on inputs with the property P and at most 1/3 on inputs far from having the property P. Szegedy [39] asked whether it is possible to algebraically characterize the complexity of classical testing by the minimum degree of such polynomials; however, our separation results imply that there are properties, for which such polynomials have constant degree, but for which the best classical tester needs $\Omega(\log n)$ queries. Hence, the minimum degree is only a lower bound, which sometimes is not tight.

A priori it is conceivable that every language has a quantum property tester with a small number of queries. We show that this is not the case. We prove that for most properties of a certain size, every quantum algorithm requires $\Omega(n)$ queries. We then show that a natural property, namely, the range of a d-wise independent pseudorandom generator, cannot be quantumly tested with less than $(d+1)/2$ queries for every odd $d \leq n/\log n - 1$.

While our paper is the first to explicitly consider property testing in the quantum setting, several previous papers have considered related testing problems [31, 17]. The algorithms of Hales and Hallgren [29] give a property tester for periodicity when the bad function is also periodic.

2. Preliminaries.

We will use the following formal definition of property testing from Goldreich [24].

Definition 1. *Let S be a finite set and P a set of functions mapping S to $\{0,1\}$. A* property tester *for P is a probabilistic oracle machine M, which given a distance parameter $\epsilon > 0$ and oracle access to a function $f : S \to \{0,1\}$ satisfies the following conditions:*

1. *the tester accepts f if it is in P: if $f \in P$, then $\Pr(M^f(\epsilon) = 1) \geq 2/3$;*
2. *the tester rejects f if it is far from P: if $|\{x \in S : f(x) \neq g(x)\}| > \epsilon \cdot |S|$, for every $g \in P$, then $\Pr(M^f(\epsilon) = 1) \leq 1/3$.*

Here M^f denotes that the machine M is provided with the oracle for f.

Definition 2. *The complexity of the tester is the number of oracle queries it makes: a property P has an* (ϵ, q)-tester *if there is a tester for P that makes at most*

q oracle queries for distance parameter ϵ.

We often consider a language $L \subseteq \{0,1\}^*$ as the family of properties $\{P_n\}$ with P_n the characteristic functions of the length-n strings from L and analyze the query complexity $q = q(\epsilon, n)$ asymptotically for large n.

To define quantum property testing we simply modify Definition 1 by allowing M to be a quantum oracle machine. We need to be careful to make sure our oracle queries are unitary operations. If $|f(x)| = |g(y)|$ for all $x, y \in S$ and $f, g \in P$, we use the oracle-query model by Beals et al. [10]: we define the unitary transformation U_f that maps the basis state $|x, y, z\rangle$ to $|x, y \oplus f(x), z\rangle$, where $|x| = \lceil \log |S| \rceil$, $|y| = |f(x)|$, and $\oplus$ denotes bitwise exclusive or. In case there are x, y, f, g so that $|f(x)| \neq |g(y)|$, we define U_f as mapping $|x, l, y, z\rangle$ to $\big|x,\, l + |f(x)| \bmod k,\, y \oplus 0^{k-|f(x)|} f(x),\, z\big\rangle$, where $k = \max\{|f(x)| : f \in P \text{ and } x \in S\}$, $|x| = \lceil \log |S| \rceil$, $|l| = \lceil \log k \rceil$, and $|y| = k$.

We recommend the book of Nielsen and Chuang [32] for background information on quantum computing.

3. Separating quantum and classical property testing. We show that there exist languages with $(\epsilon, \mathrm{O}(1))$ quantum property testers that do not have $(\epsilon, \mathrm{O}(1))$ classical testers.

THEOREM 1. *There is a language L that is ϵ-testable by a quantum test with $\mathrm{O}(1/\epsilon)$ queries but for which every probabilistic ϵ-test with $\epsilon \leq 1/2$ requires $\Omega(\log n)$ queries.*

We use Hadamard codes to provide examples for Theorem 1.

DEFINITION 3. *The* Hadamard code *of $y \in \{0,1\}^{\log n}$ is $x = h(y) \in \{0,1\}^n$, where $x_i = y \cdot i$ for every index i. Here $i \in \{0, \ldots, n-1\}$ and we identify $\{0, \ldots, n-1\}$ with $\{0,1\}^{\log n}$; $y \cdot i = \sum_{j=0}^{\log n - 1} y_j i_j \bmod 2$ denotes the inner product of two vectors $y, i \in \mathbb{F}_2^{\log n}$.*

Note that the Hadamard mapping $h : \{0,1\}^{\log n} \to \{0,1\}^n$ is one-to-one. Bernstein and Vazirani [12] showed that given a codeword x, a quantum computer can extract the y for which $x = h(y)$ with one query to an oracle for the bits of x, whereas a classical probabilistic procedure needs $\Omega(\log n)$ queries. Based on this separation for a decision problem we construct for $A \subseteq \{0,1\}^{\log n}$ the property $P_A \subseteq \{0,1\}^n$:

$$P_A := \{x : \exists y \in A \text{ s.t. } x = h(y)\}.$$

Theorem 1 follows from the following two lemmas.

LEMMA 2. *For every A, P_A has an $(\epsilon, \mathrm{O}(1/\epsilon))$ quantum tester. Furthermore, the test has one-sided error.*

LEMMA 3. *For most $A \subseteq \{0,1\}^{\log n}$, P_A requires $\Omega(\log n)$ queries for a $1/2$-test, even for testers with two-sided error.*

Before we prove Lemma 2 we note that for every A, P_A can be tested by a classical one-sided-error algorithm with $\mathrm{O}(1/\epsilon + \log n)$ queries even nonadaptively; hence, the result of Lemma 3 is tight. An $\mathrm{O}(1/\epsilon \log n)$-test follows from Theorem 4 below. The slightly more efficient test, of query complexity $\log n + \mathrm{O}(1/\epsilon)$, is the following: First we query x_{2^i}, $i = 0, \ldots, \log n - 1$. Note that if $x = h(y)$, then $y_i = x_{2^i}$ for $i = 0, \ldots, \log n - 1$. Thus a candidate y for $x = h(y)$ is found. If $y \notin A$, then x is rejected. Then $k = \mathrm{O}(1/\epsilon)$ times the following check is performed: an index $i \in \{0, \ldots, n-1\}$ is chosen independently and uniformly at random and if $x_i \neq y \cdot i$, then x is rejected. Otherwise, x is accepted. Clearly, if x is rejected, then $x \notin P_A$. It is easily verified that if x has Hamming distance more than ϵn from every z in P_A, then with constant probability x is rejected.

Proof of Lemma 2. P_A can be checked with $\mathrm{O}(1/\epsilon)$ queries on a quantum computer: The test is similar to the test above, except that y can be found in $\mathrm{O}(1)$ queries: k times query for random i, j values x_i, x_j, and $x_{i\oplus j}$. If $x_i \oplus x_j \neq x_{i\oplus j}$, reject. $k = \mathrm{O}(1/\epsilon)$ is sufficient to detect an input x that is ϵn-far from being a Hadamard codeword with high probability. Now run the Bernstein–Vazirani algorithm to obtain y. Accept if and only if $y \in A$. Obviously, if $x \in P_A$, the given procedure accepts, and if x is far from each $x' \in P_A$, then it is either far from being a Hadamard codeword or it is close to a Hadamard codeword $h(y')$ for a $y' \notin A$; note that in this case x is far from every $h(y)$, $y \in A$, as two distinct Hadamard codewords are of Hamming distance $n/2$. Thus, in this case the second part of the tester succeeds with high probability in finding y' and rejects because $y' \notin A$. We also note that this algorithm has one-sided error. □

Proof of Lemma 3. The lower bound makes use of the Yao principle [40]: let D be an arbitrary probability distribution on positive and negative inputs, i.e., on inputs that either belong to P_A or are $n/2$-far from P_A. Then if every deterministic algorithm that makes at most q queries errs with probability at least $1/16$ with respect to input chosen according to D, then q is a lower bound on the number of queries of any randomized algorithm for testing P_A with error probability bounded by $1/16$.

For our lower bound, D will be the uniform distribution over Hadamard codewords of length n, namely, generated by choosing $y \in \{0,1\}^{\log n}$ uniformly at random and setting $x = h(y)$. Note that for any $A \subset \{0,1\}^{\log n}$, D is concentrated on positive and negative inputs as required, as two Hadamard codewords are of Hamming distance $n/2$ apart.

The lower bound will be established by a counting argument. We show that for a fixed tester that makes $q \leq (\log n)/2$ queries the probability over random choices of A that the algorithm errs on at most $1/16$ of the inputs is much less than $1/T$, where T is the number of such algorithms. By the union bound it follows that for most properties there is no such algorithm.

Indeed, choose $A \subseteq \{0,1\}^{\log n}$ by picking independently each $i \in \{0,1\}^{\log n}$ to be in A with probability $1/2$. Let $\mathcal{T}$ be any fixed deterministic decision tree performing at most q queries in every branch. Let $A_{\mathcal{T}} := \{y \mid \mathcal{T}(h(y)) = \text{accept}\}$ and let $\mathrm{err}(\mathcal{T}, A) := \big|(A \setminus A_{\mathcal{T}}) \cup (A_{\mathcal{T}} \setminus A)\big|/n$ denote the error probability of $\mathcal{T}$ for property P_A with input distribution D. Assume first that $|A_{\mathcal{T}}| \leq n/2$. Since for a $h(y) \in \{0,1\}^n$ chosen according to D we have $\Pr_y[\mathcal{T}(h(y)) = \text{accept}] = |A_{\mathcal{T}}|/n \leq 1/2$, it follows by a Chernoff-type bound [9] that $\Pr_A[|A \cap A_{\mathcal{T}}| \geq 3/4|A|] \leq 2^{-n/8}$. If $|A \cap A_{\mathcal{T}}| < 3/4|A|$, then $\mathcal{T}$ will be wrong on at least $|A|/4$ of the positive inputs. With high probability, A is not too small: a Chernoff-type bound implies $\Pr_A[|A| \leq n/4] \leq 2^{-n/16}$. Then $\Pr_A[\mathrm{err}(\mathcal{T}, A) < 1/16] \leq \Pr_A\big[|A| \leq n/4\big] + \Pr_A\big[\mathrm{err}(\mathcal{T}, A) < 1/16 \;\big|\; |A| \geq n/4\big] \leq 2^{-n/8} + 2^{-n/16} \leq 2 \cdot 2^{-n/16}$. If $|A_{\mathcal{T}}| > n/2$, the same reasoning shows that with probability of at most $2 \cdot 2^{-n/16}$, $\mathcal{T}$ will err with D-probability less than $1/16$ on the negative inputs. Overall, we have for every fixed $\mathcal{T}$

$$\Pr_A[\mathrm{err}(\mathcal{T}, A) \leq 1/16] \leq 2 \cdot 2^{-n/16}.$$

Now let us bound from above the number T of algorithms that make at most q queries. As an algorithm may be adaptive, it can be defined by $2^q - 1$ query positions for all queries on all branches and a Boolean function $f : \{0,1\}^q \to \{\text{accept}, \text{reject}\}$ of the decision made by the algorithm for the possible answers. Hence, there are at most $T \leq (2n)^{2^q}$ such algorithms. However, for $q \leq (\log n)/2$, we have $T \cdot 2 \cdot 2^{-n/16} = \mathrm{o}(1)$, which shows that for most A as above, every $1/2$-test that queries at most $(\log n)/2$

many queries has error probability of at least 1/16. Standard amplification techniques then imply that for some constant c, every 1/2-test that performs $c \log n$ many queries has error greater than 1/3. $\square$

Theorem 4. *Let $P \subseteq \{0,1\}^n$ be a property with $|P| > 0$. For any $\epsilon > 0$, P can be ϵ-tested by a one-sided error classical algorithm using* $\mathrm{O}((\log |P|)/\epsilon)$ *queries.*

Proof. Denote the input by $y \in \{0,1\}^n$ and $s := |P|$. Consider the following algorithm: query the input y in $k := \ln(3s^2)/\epsilon$ random places; accept if there is at least one $x \in P$ consistent with the bits from the input and reject otherwise. Clearly, if $y \in P$, this algorithm works correctly.

If y is ϵ-far from each $x \in P$, then for every specific $x \in P$, $\Pr[x_i = y_i] \leq 1-\epsilon$ when choosing an $i \in [n]$ uniformly at random. With k indices chosen independently and uniformly at random, the probability for no disagreement with x becomes $(1-\epsilon)^k \leq 1/(3s^2)$. Therefore, the probability that there is no disagreement for at least one of the s members of P is at most $1/(3s)$, and so with probability 2/3 for a y that is far from P, we will rule out every $x \in P$ as being consistent with y. $\square$

4. An exponential separation. In this section, we show that a quantum computer can be exponentially more efficient than a classical computer in testing certain properties.

Theorem 5. *There exists a language L that for every $\epsilon = \Omega(1)$ is* $(\epsilon, \log n \log \log n)$ *quantumly testable, but every classical* $1/8$*-test for L requires* $\Omega(\sqrt{n})$ *queries.*

The language that we provide is inspired by Simon's problem [37], and our quantum testing algorithm makes use of Brassard and Høyer's algorithm for Simon's problem [14]. Simon's problem is to find $s \in \{0,1\}^n \setminus \{0^n\}$ from a function-query oracle for some $f : \{0,1\}^n \to \{0,1\}^n$ with the promise that $f(x) = f(y) \Leftrightarrow x = y \oplus s$. Simon proved that, classically, $\Omega(2^{n/2})$ queries are required on average to find s and gave a quantum algorithm for determining s with an expected number of queries that is polynomial in n; Brassard and Høyer improved the quantum algorithm to worst-case polynomial time. Their algorithm produces in each run a z with $z \cdot s = 0$ that is linearly independent to all previously computed such z's. Essentially, our quantum tester uses this subroutine to try to extract information about s until it fails repeatedly. Høyer [30] analyzed this approach in group-theoretic terms, obtaining an alternative proof to Theorem 7. Friedl et al. [23] generalize Theorem 7 to hold for languages based on any finite Abelian group.

In the following, let $N = 2^n$ denote the length of the binary string encoding a function $f : \{0,1\}^n \to \{0,1\}$. For $x \in \{0,1\}^n$ let $x[j]$ denote the jth bit of x, i.e., $x = x[1] \ldots x[n]$. We define

$$L := \{f \in \{0,1\}^N : \exists s \in \{0,1\}^n \setminus \{0^n\} \ \forall x \in \{0,1\}^n \ f(x) = f(x \oplus s)\}.$$

Theorem 5 follows from the following two theorems.

Theorem 6. *Every classical* $1/8$*-tester for L must make* $\Omega(\sqrt{N})$ *queries, even when allowing two-sided error.*

Theorem 7. *There is a quantum property tester for L making* $\mathrm{O}(\log N \log \log N)$ *queries. Moreover, this quantum property tester makes all its queries nonadaptively.*

Proof of Theorem 6. We again apply the Yao principle [40] as in the proof of Lemma 3: we construct two distributions, P and U, on positive and at least $(N/8)$-far negative inputs, respectively, and let D be a distribution that is defined by $D = (P+U)/2$. We will show that every adaptive decision tree $\mathcal{T}$ has error $1/2 - \mathrm{o}(1)$ on a random input chosen according to D.

The distribution P is defined as follows: We first choose $s \in \{0,1\}^n$ at random. This defines a matching M_s of $\{0,1\}^n$ by matching x with $x \oplus s$. Now a function f_s is defined by choosing for each matched pair independently $f_s(x) = f_s(x \oplus s) = 1$ with probability 1/2 and $f_s(x) = f_s(x \oplus s) = 0$ with probability 1/2. Clearly, this defines a distribution that is concentrated on positive inputs. Note that it might be that by choosing different s's we end up choosing the same function. However, these will be considered different events in the probability space; i.e., the atomic events in P are the pairs (s, f_s) as described above.

Now let $\tilde{U}$ be the uniform distribution over all functions: we select the function by choosing for each x independently $f(x) = 1$ with probability 1/2 and 0 with probability 1/2. Since every function has a nonzero probability, $\tilde{U}$ is not supported exclusively on the negative instances. We define U to be $\tilde{U}$ conditioned on the event that the input is $N/8$ far from the property. As we show in Lemma 8, a function chosen according to $\tilde{U}$ is $N/8$ far from having the property with very high probability, and hence it will be a good approximation for U.

Let $\mathcal{T}$ be any deterministic decision tree. Let v be a vertex of $\mathcal{T}$. We will show that for every vertex v of small depth in $\mathcal{T}$, $\Pr_P[\text{input } f \text{ is consistent with } v] = \Pr_U[\text{input } f \text{ is consistent with } v](1 + \mathrm{o}(1))$, from which we will conclude that $\mathcal{T}$ has error $1/2 - \mathrm{o}(1)$.

Definition 4. *For $f : \{0,1\}^n \to \{0,1\}$ and $s \in \{0,1\}^n$, we define $n_s := |\{x : f(x) = f(x \oplus s)\}|$.*

Lemma 8. *Let f be chosen according to $\tilde{U}$. Then $\Pr_{\tilde{U}}[\exists s \in \{0,1\}^n : n_s \geq 3N/4] = e^{-\Omega(N)}$.*

Proof. Let f be chosen according to $\tilde{U}$ and $s \in \{0,1\}^n$. By a Chernoff bound [9], we obtain $\Pr_{\tilde{U}}[n_s \geq 3N/4] = e^{-\Omega(N)}$ for every fixed s. Together with the union bound over all $2^n = N$ choices of s this yields $\Pr_{\tilde{U}}[\exists s \in \{0,1\}^n : n_s \geq 3N/4] = N \cdot e^{-\Omega(N)} = e^{-\Omega(N)}$. □

For every s, we need to change $(N - n_s)/2$ values of f to get an input f' that has the property $f'(x) = f'(x \oplus s)$ for all x. Hence, Lemma 8 implies that with probability $1 - e^{-\Omega(N)}$ an input chosen according to $\tilde{U}$ will be $N/8$ far from having the property.

From the definition of $\tilde{U}$, we immediately obtain the following.

Lemma 9. *Let $\mathcal{T}$ be any fixed deterministic decision tree and let v be a vertex of depth d in $\mathcal{T}$. Then $\Pr_{\tilde{U}}[f \text{ is consistent with the path to } v] = 2^{-d}$.*

We now want to derive a similar bound as in Lemma 9 for functions chosen according to P. For this we need the following definition for the event that after d queries, nothing has been learned about the hidden s.

Definition 5. *Let $\mathcal{T}$ be a deterministic decision tree and u a vertex in $\mathcal{T}$ at depth d. We denote the path from the root of $\mathcal{T}$ to v by $\mathrm{path}(v)$. Every vertex v in $\mathcal{T}$ defines a query position $x_v \in \{0,1\}^n$. For $f = f_s$ chosen according to P, we denote by B_v the event $B_v := \{(s, f_s) : s \neq x_u \oplus x_w \; \forall u, w \in \mathrm{path}(v)\}$.*

Lemma 10. *Let v be a vertex of depth d in a decision tree $\mathcal{T}$. Then $\Pr_P[B_v] \geq 1 - \binom{d-1}{2}/N$.*

Proof. B_v does not occur if for some v, w on the path to v we have $s = x_v \oplus x_w$. As there are $d - 1$ such vertices, there are at most $\binom{d-1}{2}$ pairs. Each of these pairs excludes exactly one s, and there are N possible values of s. □

Lemma 11. *Let v be a vertex of depth d in a decision tree $\mathcal{T}$ and let f be chosen according to P. Then $\Pr_P[f \text{ is consistent with } v \mid B_v] = 2^{-d}$.*

Proof. By the definition of P, f gets independently random values on vertices that are not matched. But if B_v occurs, then no two vertices along the path to v are matched, and hence the claim follows. □

Procedure SimonTester

```
for $k = 0$ to $n - 1$ do
    $l \leftarrow 0$
    repeat
        $z \leftarrow$ SimonSampler($z_1, \ldots, z_k$)
        $l \leftarrow l + 1$
    until $z \neq 0^n$ or $l > 2(\log n)/\epsilon^2$
    if $z = 0^n$ then
        accept
    else
        $z_{k+1} \leftarrow z$
reject
```

Procedure SimonSampler($z_1, \ldots, z_k$)

```
input: $z_1, \ldots, z_k \in \{0,1\}^n$
output: $z \in \{0,1\}^n$
quantum workspace: $\mathcal{X} \otimes \mathcal{Y} \otimes \mathcal{Z}$, where
$\mathcal{X}$ is $n$ qubits $\mathcal{X} = \mathcal{X}_1 \otimes \cdots \otimes \mathcal{X}_n$, $\mathcal{X}_i = \mathbb{C}^2$,
$\mathcal{Y} = \mathbb{C}^2$ is one qubit, and
$\mathcal{Z}$ is $k$ qubits $\mathcal{Z} = \mathcal{Z}_1 \otimes \cdots \otimes \mathcal{Z}_k$, $\mathcal{Z}_j = \mathbb{C}^2$
initialize the workspace to $|0^n\rangle|0\rangle|0^k\rangle$
apply $H_{2^n}$ to $\mathcal{X}$
apply $U_f$ to $\mathcal{X} \otimes \mathcal{Y}$
apply $H_{2^n}$ to $\mathcal{X}$
for $j = 1$ to $k$ do
    $i \leftarrow \min\{i : z_j[i] = 1\}$
    apply CNOT with control $\mathcal{X}_i$ and target $\mathcal{Z}_j$
    apply $|x\rangle \mapsto |x \oplus z_j\rangle$ to $\mathcal{X}$ conditional on $\mathcal{Z}_j$
    apply $H_2$ to $\mathcal{Z}_j$
return measurement of $\mathcal{X}$
```

Now we can complete the proof of the theorem: assume that $\mathcal{T}$ is a deterministic decision tree of depth $d = \mathrm{o}(\sqrt{N})$ and let v be any leaf of $\mathcal{T}$. Then by Lemmas 10 and 11, we get that $\Pr_P[f$ is consistent with $v] = (1 \pm \mathrm{o}(1))2^{-d}$. On the other hand, by Lemmas 8 and 9 we get that $\Pr_U[f$ is consistent with $v] = (1 \pm \mathrm{o}(1))2^{-d}$, and hence $\mathcal{T}$ has only o(1) bias factor of being right on every leaf. This implies that its error probability is $1/2 - \mathrm{o}(1)$. □

Proof of Theorem 7. We give a quantum algorithm making $\mathrm{O}(\log N \,\log\log N)$ queries to the quantum oracle for input $f \in \{0,1\}^N$. We will show that it accepts with probability 1 if $f \in L$ and rejects with high probability if the Hamming distance between f and every $g \in L$ is at least ϵN. Pseudocode for our algorithm is given in the above procedures; it consists of a classical main program SimonTester and a quantum subroutine SimonSampler adapted from Brassard and Høyer's algorithm for Simon's problem [14, section 4]. The quantum gates used are the 2^n-dimensional Hadamard transform H_{2^n}, which applies

$$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

individually to each of n qubits, the quantum oracle query U_f, and classical reversible operations run in quantum superposition.

The following technical lemma captures the operation of the quantum subroutine SimonSampler. For $i_1, \ldots, i_J$ fixed, let $Y_J := \{y \in \{0,1\}^n : \forall j \leq J \; y[i_j] = 0\}$ denote the length-n binary strings that are 0 at positions $i_1, \ldots, i_J$.

LEMMA 12. *When* SimonSampler *is passed k linearly independent vectors $z_1, \ldots, z_k$ so that all $i_j := \min\{i : z_j[i] = 1\}$ are distinct for $1 \leq j \leq k$, then the state $|\psi\rangle$ before the measurement is*

$$\frac{\sqrt{2^k}}{N} \sum_{x \in \{0,1\}^n} \sum_{y \in Y_k} (-1)^{x \cdot y} |y\rangle |f(x)\rangle |x \cdot z_1\rangle \cdots |x \cdot z_k\rangle.$$

Proof. We follow the steps of subroutine SimonSampler when it is passed k linearly independent vectors $z_1, \ldots, z_k$ so that all $i_j := \min\{i : z_j[i] = 1\}$ are distinct for $1 \leq j \leq k$:

$$|0^n\rangle|0\rangle|0^k\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle|0\rangle|0^k\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle|f(x)\rangle|0^k\rangle$$
$$\mapsto \frac{1}{N} \sum_{x,y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle|f(x)\rangle|0^k\rangle.$$

This is the state before the **for** loop is entered. We claim and proceed to show by induction that after the Jth execution of the loop body, the state is

$$\frac{\sqrt{2^J}}{N} \sum_{x \in \{0,1\}^n} \sum_{y \in Y_J} (-1)^{x \cdot y} |y\rangle |f(x)\rangle |x \cdot z_1\rangle \cdots |x \cdot z_J\rangle |0^{k-J}\rangle.$$

Executing the body of the loop for $j = J + 1$,

$$\frac{\sqrt{2^J}}{N} \sum_{x \in \{0,1\}^n} \sum_{y \in Y_J} (-1)^{x \cdot y} |y\rangle |f(x)\rangle |x \cdot z_1\rangle \cdots |x \cdot z_J\rangle |0\rangle |0^{k-J-1}\rangle$$
$$\mapsto \frac{\sqrt{2^J}}{N} \sum_{x \in \{0,1\}^n} \sum_{y \in Y_J} (-1)^{x \cdot y} |y\rangle |f(x)\rangle |x \cdot z_1\rangle \cdots |x \cdot z_J\rangle |y[i_{j+1}]\rangle |0^{k-J-1}\rangle$$
$$= \frac{\sqrt{2^J}}{N} \sum_{\substack{x \in \{0,1\}^n \\ y \in Y_{J+1} \\ b \in \{0,1\}}} (-1)^{x \cdot (y \oplus b z_{J+1})} |y \oplus b z_{J+1}\rangle |f(x)\rangle |x \cdot z_1\rangle \cdots |x \cdot z_J\rangle |b\rangle |0^{k-J-1}\rangle$$

(Here we used the fact that $Y_J = Y_{J+1} \,\dot\cup\, (z_{J+1} \oplus Y_{J+1})$.)

$$\mapsto \frac{\sqrt{2^J}}{N} \sum_{\substack{x \in \{0,1\}^n \\ y \in Y_{J+1} \\ b \in \{0,1\}}} (-1)^{x \cdot (y \oplus b z_{J+1})} |y\rangle |f(x)\rangle |x \cdot z_1\rangle \cdots |x \cdot z_J\rangle |b\rangle |0^{k-J-1}\rangle$$
$$= \frac{\sqrt{2^{J+1}}}{N} \sum_{x \in \{0,1\}^n} \sum_{y \in Y_{J+1}} (-1)^{x \cdot y} |y\rangle |f(x)\rangle |x \cdot z_1\rangle \cdots |x \cdot z_J\rangle$$
$$\times \frac{1}{\sqrt{2}} \sum_{b \in \{0,1\}} (-1)^{x \cdot (b z_{J+1})} |b\rangle |0^{k-J-1}\rangle$$

$$\mapsto \frac{\sqrt{2^{J+1}}}{N} \sum_{x\in\{0,1\}^n} \sum_{y\in Y_{J+1}} (-1)^{x\cdot y}|y\rangle|f(x)\rangle|x\cdot z_1\rangle\cdots|x\cdot z_{J+1}\rangle|0^{k-J-1}\rangle. \qquad \square$$

As an immediate consequence, we can establish the invariant that in SimonTester $\{z_1,\ldots,z_k\}$ is always linearly independent with $i_j = \min\{i : z_j[i] = 1\}$ distinct for $1 \le j \le k$; moreover, if $f \in L$, then just as in Simon's algorithm, a nonzero z is orthogonal to the hidden s.

LEMMA 13. *If measuring the first register, $\mathcal{X}$, yields a nonzero value z, then*

1. *$\{z_1,\ldots,z_k,z\}$ is linearly independent;*
2. *$\min\{i : z[i] = 1\}$ is distinct from i_j for $1 \le j \le k$; and*
3. *if $f \in L$, then $z\cdot s = 0$ for every $s \ne 0^n$ such that $f(x) = f(x\oplus s)$ for all x.*

Proof. If we measure the state from Lemma 12, then for the value z of the first register it holds that $z \in Y_k$. This implies 2, from which follows 1. For 3, as in Simon's original algorithm, if there is a $s \ne 0^n$ so that for all x, $f(x) = f(x\oplus s)$, then we can rewrite the state from Lemma 12 as

$$\frac{\sqrt{2^k}}{N} \sum_{\substack{x:x<x\oplus s\\ y\in Y_k}} |y\rangle \left((-1)^{x\cdot y}|f(x)\rangle + (-1)^{(x\oplus s)\cdot y}|f(x\oplus s)\rangle\right)|x\cdot z_1\rangle\cdots|x\cdot z_k\rangle$$

$$= \frac{\sqrt{2^k}}{N} \sum_{x:x<x\oplus s} \sum_{y\in Y_k} |y\rangle(-1)^{x\cdot y}\left(1+(-1)^{s\cdot y}\right)|f(x)\rangle|x\cdot z_1\rangle\cdots|x\cdot z_k\rangle.$$

Hence, only y with $s\cdot y = 0$ will have nonzero amplitude. □

Next, we want to assess the probability of obtaining $z = 0^n$ in SimonTester line 4. We let P_0 denote the projection operator mapping $|0^n\rangle|y\rangle|z\rangle \mapsto |0^n\rangle|y\rangle|z\rangle$ and $|x\rangle|y\rangle|z\rangle \mapsto 0$ for $x \ne 0^n$; hence, $\|P_0|\psi\rangle\|^2$ is the probability of obtaining 0 when measuring subspace $\mathcal{X}$ of the quantum register in state $|\psi\rangle$. We can characterize the probability for outcome $z = 0^n$ in terms of the following definition and lemma.

DEFINITION 6. *For $c \in \{0,1\}^k$ and $z_1,\ldots,z_k \in \{0,1\}^n$ we define $D_c := \{x \in \{0,1\}^n : x\cdot z_1 = c[1],\ \ldots,\ x\cdot z_k = c[k]\}$.*

LEMMA 14. *Let $|\psi\rangle$ be the state before the measurement in* SimonSampler *when* SimonSampler *is passed k linearly independent vectors $z_1,\ldots,z_k$ so that all $i_j := \min\{i : z_j[i] = 1\}$ are distinct for $1 \le j \le k$.*

1. *$\|P_0|\psi\rangle\|^2 = 1$ if and only if for every $c \in \{0,1\}^k$, f is constant when restricted to D_c.*
2. *If $\|P_0|\psi\rangle\|^2 \ge 1-\epsilon^2/2$, then f differs in at most ϵN points from some function g that is constant when restricted to D_c for every $c \in \{0,1\}^k$.*

Proof. For $b \in \{0,1\}$ let $D_{b,c} := D_c \cap f^{-1}\{b\} = \{x : f(x) = b$ and $x\cdot z_1 = c[1],\ \ldots,\ x\cdot z_k = c[k]\}$. Note that the $D_{b,c}$ and D_c also depend on $z_1,\ldots,z_k$ and the $D_{b,c}$ depend on f. Let

$$|\psi_0\rangle := \frac{\sqrt{2^k}}{N} \sum_{x\in\{0,1\}^n} |0^n\rangle|f(x)\rangle|x\cdot z_1\rangle\cdots|x\cdot z_k\rangle$$

$$= \frac{\sqrt{2^k}}{N} \sum_{b\in\{0,1\}} \sum_{c\in\{0,1\}^k} |D_{b,c}|\,|0^n\rangle|b\rangle|c[1]\rangle\cdots|c[k]\rangle.$$

By Lemma 12, at the end of SimonSampler the system is in state $|\psi\rangle = |\psi_0\rangle + |\psi_0^\perp\rangle$ for some $|\psi_0^\perp\rangle$ orthogonal to $|\psi_0\rangle$. We consider the case $\|P_0|\psi\rangle\|^2 = 1$. Then the register

$\mathcal{X}$ must be in state $|0^n\rangle$, and thus $|\psi\rangle = |\psi_0\rangle$. Since the state has norm 1, we know that

$$\sum_{b\in\{0,1\}} \sum_{c\in\{0,1\}^k} |D_{b,c}|^2 = \frac{N^2}{2^k}. \tag{1}$$

The $D_{b,c}$ partition $\{0,1\}^n$ and the $D_c = D_{0,c} \cup D_{1,c}$ have the same size for all $c \in \{0,1\}^k$ because they are cosets of D_0. Therefore,

$$\sum_{b\in\{0,1\}} \sum_{c\in\{0,1\}^k} |D_{b,c}| = N \text{ and } |D_{0,c}| + |D_{1,c}| = \frac{N}{2^k} \quad \forall c \in \{0,1\}^k. \tag{2}$$

$|D_{0,c}|^2 + |D_{1,c}|^2 \le N^2/2^{2k}$, but in order for (1) to hold, $|D_{0,c}|^2 + |D_{1,c}|^2$ must be exactly $N^2/2^{2k}$. This can be achieved only if either $D_{0,c}$ or $D_{1,c}$ is empty. Thus f must be constant when restricted to D_c for any $c \in \{0,1\}^k$. Conversely, if f is constant when restricted to D_c for any $c \in \{0,1\}^k$, then (1) holds; therefore $\||\psi_0\rangle\| = 1$ and $|\psi\rangle = |\psi_0\rangle$. This concludes the proof of case 1 of the lemma.

If $\|P_0|\psi\rangle\|^2 = \||\psi_0\rangle\|^2 \ge 1 - \delta$, then

$$\sum_{b\in\{0,1\}} \sum_{c\in\{0,1\}^k} |D_{b,c}|^2 \ge (1-\delta)\,\frac{N^2}{2^k}. \tag{3}$$

Nevertheless, the constraints (2) hold; let $r2^k$ be the number of $c \in \{0,1\}^k$ so that $\min\{|D_{0,c}|, |D_{1,c}|\} \ge \gamma N/2^k$. Then

$$\sum_{b\in\{0,1\}} \sum_{c\in\{0,1\}^k} |D_{b,c}|^2 \le r2^k(\gamma^2 + (1-\gamma)^2)\frac{N^2}{2^{2k}} + (1-r)2^k\frac{N^2}{2^{2k}},$$

and using (3), we obtain $r \le \delta/(1-\gamma^2-(1-\gamma)^2)$. With $\delta = \epsilon^2/2$ and $\gamma = \epsilon/2$, this implies $r \le \epsilon$. But then

$$\sum_{c\in\{0,1\}^k} \min\{|D_{0,c}|, |D_{1,c}|\} \le r2^k\frac{N}{2^{k+1}} + (1-r)2^k\gamma\frac{N}{2^k} \le \epsilon N. \qquad \square$$

We need to relate these two cases to membership in L and bound the number of repetitions needed to distinguish between the two cases. This is achieved by the following two lemmas.

LEMMA 15. *Let k be the minimum number of linearly independent vectors $z_1, \ldots, z_k$ so that for each $c \in \{0,1\}^k$, f is constant when restricted to D_c. Then $f \in L$ if and only if $k < n$.*

Proof. If $k < n$, then there exists an $s \ne 0^n$ with $s \cdot z_1 = 0, \ldots, s \cdot z_k = 0$. For each such s and all x, we have $x \cdot z_1 = (x \oplus s) \cdot z_1, \ldots, x \cdot z_k = (x \oplus s) \cdot z_k$ and $x \in D_{f(x), x\cdot z_1, \ldots, x\cdot z_k}$ and $x \oplus s \in D_{f(x\oplus s), x\cdot z_1, \ldots, x \cdot z_k}$; therefore $f(x) = f(x \oplus s)$. Conversely, for $f \in L$, $S := \{s : \forall x,\ f(x) = f(x \oplus s)\}$ is a nontrivial subspace of $\{0,1\}^n$; therefore $S^\perp = \{z : z \cdot s = 0\ \forall s \in S\}$ is a proper subspace of $\{0,1\}^n$. Let $z_1, \ldots, z_k$ be an arbitrary basis of $S^\perp$. $\square$

LEMMA 16. *Let $0 < q < 1$ and let $|\phi_1\rangle, \ldots, |\phi_m\rangle$ be quantum states satisfying $\|P_0|\phi_j\rangle\|^2 < 1 - \delta$ for $1 \le j \le m$. If $m = \log q / \log(1-\delta) = \Theta(-(\log q)/\delta)$, then with probability at most q measuring the $\mathcal{X}$ register of $|\phi_1\rangle, \ldots, |\phi_m\rangle$ will yield m times outcome 0.*

Proof.

$$\Pr\left[m \text{ times } 0 \;\middle|\; \forall j : \|P_0|\phi_j\rangle\|^2 < 1-\delta\right] < (1-\delta)^m = (1-\delta)^{\log q \,/\, \log(1-\delta)} = q. \qquad \square$$

Now all the ingredients for wrapping up the argument are at hand; first consider $f \in L$. Let $S := \{s : f(x) = f(x \oplus s)\ \forall x\}$ be the set of all "Simon promises" of f and $S^\perp := \{z : z \cdot s = 0\ \forall s \in S\}$ the vectors that are orthogonal to all such promises. By Lemma 13 the nonzero z computed by the algorithm lie in $S^\perp$ and are linearly independent; therefore after $\dim S^\perp$ rounds of the **for** loop in SimonTester, we measure $z = 0^n$ with certainty. Since $f \in L$, $\dim S > 0$, and thus $\dim S^\perp < n$.

If f is ϵn-far from being in L, then by Lemma 15 f is ϵn-far from being close to a function for which a $k < n$ and $z_1, \dots, z_k$ exist so that f is constant when restricted to D_c for any of the $c \in \{0,1\}^k$. Therefore, by case 2 of Lemma 14, for all $k < n$, $\|P_0|\psi\rangle\|^2 < 1 - \epsilon^2/2$. Thus, Lemma 16 guarantees that we accept with probability at most $1/3$ if we let $q = 1/(3n)$, and thus $m = \mathrm{O}((\log n)/\epsilon^2)$.

This concludes the proof of Theorem 7. $\square$

5. Quantum lower bounds.

5. Quantum lower bounds. In this section we prove that not every language has a fast quantum property tester.

THEOREM 17. *Most properties containing $2^{n/20}$ elements of $\{0,1\}^n$ require quantum property testers using $\Omega(n)$ queries.*

Proof. Fix n, a small ϵ, and a quantum algorithm A making $q := n/400$ queries. Pick a property P as a random subset of $\{0,1\}^n$ of size $2^{n/20}$. Let

$$P_\epsilon := \{y : d(x,y) < \epsilon n \text{ for some } x \in P\},$$

where $d(x,y)$ denotes the Hamming distance between x and y. Using $\sum_{k=0}^{\epsilon n} \binom{n}{k} \leq 2^{H(\epsilon)n}$ for

$$H(\epsilon) := -\epsilon \log \epsilon - (1-\epsilon)\log(1-\epsilon),$$

we obtain $|P_\epsilon| \leq 2^{(1/20 + H(\epsilon))n}$. In order for A to test properties of size $2^{n/20}$, it needs to reject with high probability on at least $2^n - 2^{(1/20+H(\epsilon))n}$ inputs; but then, the probability that A accepts with high probability on a random $x \in \{0,1\}^n$ is bounded by $2^{(1/20+H(\epsilon))n}/2^n$, and therefore the probability that A accepts with high probability on $|P|$ random inputs is bounded by

$$2^{-(1-1/20-H(\epsilon))n|P|} = 2^{-2^{n/20+\Theta(\log n)}}.$$

We would like to sum this success probability over all algorithms using the union bound to argue that for most properties no algorithm can succeed. However, there is an uncountable number of possible quantum algorithms with arbitrary quantum transitions. But by Beals et al. [10], the acceptance probability of A can be written as a multilinear polynomial of degree at most $2q$ where the n variables are the bits of the input; using results of Bennett et al. [11] and Solovay and Yao [38], every quantum algorithm can be approximated by another algorithm such that the coefficients of the polynomials describing the accepting probability are integers of absolute value less than $2^{n^{\mathrm{O}(1)}}$ over some fixed denominator. There are less than $2^{nH(2q/n)}$ degree-$2q$ monomials in n variables, and thus we can limit ourselves to $2^{n^{\mathrm{O}(1)} 2^{nH(2q/n)}} \leq 2^{2^{(n/20)\cdot(91/100)+\Theta(\log n)}}$ algorithms.

Thus, by the union bound, for most properties of size $2^{n/20}$, no quantum algorithm with q queries will be a tester for it. $\square$

We also give an explicit natural property that requires a large number of quantum queries to test. We will make use of the following lemma.

Lemma 18 (see [2]). *Suppose $n = 2^k - 1$ and $d = 2t + 1 \leq n$. Then there exists a multiset of n-bit strings $P = P(n, d) \subseteq \{0, 1\}^n$ of size $2(n + 1)^t$ such that under the uniform distribution over P, the Boolean random variables $\xi_1, \ldots, \xi_n$ that are the projection of $\xi \in P$ on its coordinates are d-wise independent, each taking the values* 0 *and* 1 *with probability* 1/2.

The proof of Lemma 18 is constructive, and the construction is uniform in n. For given n and d, consider the language $P(n, d)$ of n-bit strings, where $P(n, d)$ is the range of n Boolean d-wise independent variables, as asserted by the lemma. Classically, deciding membership in $P(n, d)$ takes more than d queries: for all d positions $i_1, \ldots, i_d$ and every string $v \in \{0, 1\}^d$, there is a $z \in P(n, d)$ whose restriction to $i_1, \ldots, i_d$ is v. On the other hand, $\lfloor \log |P| \rfloor + 1 = \mathrm{O}(d \log n)$ queries are always sufficient.

Theorem 19. *Let $d \leq n/ \log n - 1$ be odd and let $P = P(n, d)$ be the range of a d-wise independent Boolean variable as asserted by Lemma* 18. *Then for constant $\epsilon < 1/2$, any ϵ-quantum tester for P requires at least $(d + 1)/2$ quantum queries.*

Proof. For a property $P \subseteq \{0, 1\}^n$, again let $P_\epsilon := \{y : d(x, y) < \epsilon n$ for some $x \in P\}$. By [10], a quantum computer that ϵ-tests a property P with T queries gives rise to a degree-$2T$ multilinear n-variable polynomial $p(x) = p(x_1, \ldots, x_n)$ that approximates P in the sense that $|p(x) - f(x)| \leq 1/3$ for every $x \in P \cup (\{0, 1\}^n \setminus P_\epsilon)$. Let $p(x_1, \ldots, x_n)$ be the corresponding polynomial to a quantum ϵ-test for P. We show that there must be high-degree monomials in p by comparing the expectation of $p(x)$ for randomly chosen $x \in \{0, 1\}^n$ with the expectation of $p(x)$ for randomly chosen $x \in P$.

By the definition of P_ϵ and Lemma 18 we have

$$|P_\epsilon| \leq 2^{H(\epsilon)n} |P| = \mathrm{O}(2^{H(\epsilon)n + d \log n}).$$

Hence for $d = n/ \log n - \omega(1/ \log n)$ and $\epsilon < 1/2$ we get that $|P_\epsilon| = \mathrm{o}(2^n)$. Thus for x uniformly distributed over $\{0, 1\}^n$ we have

$$\mathrm{E}[p(x)] = \frac{|P_\epsilon|}{2^n} \mathrm{E}[p(x) \mid x \in P_\epsilon] + \left(1 - \frac{|P_\epsilon|}{2^n}\right) \mathrm{E}[p(x) \mid x \notin P_\epsilon] \leq 1/3 + \mathrm{o}(1).$$

On the other hand, by the properties of p above, for x distributed uniformly over P it holds that $\mathrm{E}[p(x) \mid x \in P] \geq 2/3$. Considering $p(x) = \sum_i \alpha_i m_i(x)$ as a linear combination of n-variable multilinear monomials m_i, we have, by the linearity of expectation, $\mathrm{E}[p(x_1, \ldots, x_n)] = \sum_i \alpha_i \mathrm{E}[m_i(x_1, \ldots, x_n)]$. But for every m_i of degree at most d, by the d-wise independence of the bits of each $x \in P$ it follows that $\mathrm{E}[m_i(x) \mid x \in P] = \mathrm{E}[m_i(x) \mid x \in U]$, where U is the uniform distribution on $\{0, 1\}^n$. Thus p must contain monomials of degree greater than d in order for those two expectations to differ by $1/3 - \mathrm{o}(1)$. We conclude that the number of queries T is greater than $d/2$. □

6. Further research. Our paper opens the door to the world of quantum property testing. Several interesting problems remain, including the following:

- Can one get the greatest possible separation of quantum and classical property testing; i.e., is there a language that requires $\Omega(n)$ classical queries but only $\mathrm{O}(1)$ quantum queries to test?
- Are there other natural problems that do not have quantum property testers? We conjecture, for instance, that the language $\{uuvv : u, v \in \Sigma^*\}$ does not have a quantum property tester.

- Beals et al. [10] observed that any k-query quantum algorithm gives rise to a degree-$2k$ polynomial in the input bits that gives the acceptance probability of the algorithm; thus, a quantum property tester for P gives rise to a polynomial that is on all binary inputs between 0 and 1, that is, at least $2/3$ on inputs with the property P and at most $1/3$ on inputs far from having the property P. Szegedy [39] suggested algebraically characterizing the complexity of classical testing by the minimum degree of such polynomials; as mentioned in the introduction, our results imply that this cannot be the case for classical testers. However, it is an open question whether quantum property testing can be algebraically characterized in this way.
- Related to the second and the third item is the following question about polynomials: Is there a property $P \subseteq \{0,1\}^n$ for which every quantum ϵ-tester requires at least $\Omega(n)$ queries but for which there is a polynomial of constant degree $p(x_1, \ldots, x_n)$ such that $0 \leq p(x) \leq 1$ for every x and $p(x) \leq 1/3$ for x's that are ϵ-far from P while $p(x) \geq 2/3$ for every $x \in P$? Such a P will show that polynomial characterization of quantum property testing, as suggested above, is impossible. It will also require other means of proving quantum nontestability results.

We hope that further research will lead to a greater understanding of what can and cannot be tested with quantum property testers.

Acknowledgment. We thank Ronitt Rubinfeld for discussions and pointers on property testing.

REFERENCES

[1] N. Alon, *Testing subgraphs in large graphs*, in Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science, 2001, pp. 434–441.

[2] N. Alon, L. Babai, and A. Itai, *A fast and simple randomized parallel algorithm for the maximal independent set problem*, J. Algorithms, 7 (1986), pp. 567–583.

[3] N. Alon, S. Dar, M. Parnas, and D. Ron, *Testing of clustering*, SIAM J. Discrete Math., 16 (2003), pp. 393–417.

[4] N. Alon, E. Fischer, M. Krivelevich, and M. Szegedy, *Efficient testing of large graphs*, in Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science, 1999, pp. 656–666.

[5] N. Alon, I. Newman, M. Krivelevich, and M. Szegedy, *Regular languages are testable with a constant number of queries*, in Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science, 1999, pp. 645–655.

[6] N. Alon and A. Shapira, *Testing subgraphs in directed graphs*, J. Comput. System Sci., 69 (2004), pp. 354–382.

[7] N. Alon and A. Shapira, *A characterization of the (natural) graph properties testable with one-sided error*, in Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, 2005, pp. 429–438.

[8] N. Alon and A. Shapira, *Every monotone graph property is testable*, in Proceedings of the 37th Annual ACM Symposium on Theory of Computing, 2005, pp. 128–137.

[9] N. Alon and J. H. Spencer, *The Probabilistic Method*, 2nd ed., Wiley-Interscience, New York, 2000.

[10] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf, *Quantum lower bounds by polynomials*, J. ACM, 48 (2001), pp. 778–797.

[11] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, *Strengths and weaknesses of quantum computing*, SIAM J. Comput., 26 (1997), pp. 1510–1523.

[12] E. Bernstein and U. Vazirani, *Quantum complexity theory*, SIAM J. Comput., 26 (1997), pp. 1411–1473.

[13] M. Blum, M. Luby, and R. Rubinfeld, *Self-testing and self-correcting programs, with applications to numerical programs*, J. Comput. System Sci., 47 (1993), pp. 549–595.

[14] G. Brassard and P. Høyer, *An exact quantum polynomial-time algorithm for Simon's prob-*

lem, in Proceedings of the 5th Israeli Symposium on Theory of Computing and Systems (ISTCS'97), IEEE Computer Society Press, Los Alamitos, CA, 1997, pp. 12–23.

[15] H. Buhrman, L. Fortnow, I. Newman, and H. Röhrig, *Quantum property testing*, in Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2003, pp. 480–488.

[16] A. Czumaj and C. Sohler, *Abstract combinatorial programs and efficient property testers*, SIAM J. Comput., 34 (2005), pp. 580–615.

[17] W. van Dam, F. Magniez, M. Mosca, and M. Santha, *Self-testing of universal and fault-tolerant sets of quantum gates*, in Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, 2000, pp. 688–696.

[18] F. Ergün, S. Kannan, S. Kumar, R. Rubinfeld, and M. Vishwanathan, *Spot-checkers*, J. Comput. System Sci., 60 (2000), pp. 717–751.

[19] E. Fischer, *The art of uninformed decisions: A primer to property testing*, Bull. Eur. Assoc. Theor. Comput. Sci. EATCS, 75 (2001), pp. 97–126.

[20] E. Fischer, *Testing graphs for colorability properties*, in Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms, 2001, pp. 873–882.

[21] E. Fischer, G. Kindler, D. Ron, S. Safra, and A. Samorodnitsky, *Testing juntas*, J. Comput. System Sci., 68 (2004), pp. 753–787.

[22] E. Fischer and I. Newman, *Testing of matrix properties*, in Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, 2001, pp. 286–295.

[23] K. Friedl, F. Magniez, M. Santha, and P. Sen, *Quantum testers for hidden group properties*, in Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science, 2003.

[24] O. Goldreich, *Combinatorial property testing (a survey)*, in Randomization Methods in Algorithm Design, AMS, Providence, RI, 1999, pp. 45–59.

[25] O. Goldreich, S. Goldwasser, and D. Ron, *Property testing and its connection to learning and approximation*, J. ACM, 45 (1998), pp. 653–750.

[26] O. Goldreich and D. Ron, *Property testing in bounded-degree graphs*, Algorithmica, 32 (2002), pp. 302–343.

[27] O. Goldreich and L. Trevisan, *Three theorems regarding testing graph properties*, in Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science, 2001, pp. 460–469.

[28] L. K. Grover, *A fast quantum mechanical algorithm for database search*, in Proceedings of the 28th ACM Symposium on Theory of Computing, 1996, pp. 212–219.

[29] L. Hales and S. Hallgren, *An improved quantum Fourier transform algorithm and applications*, in Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science, 2000, pp. 515–525.

[30] P. Høyer, *Fourier Sampling*, private communication, 2001.

[31] D. Mayers and A. Yao, *Quantum cryptography with imperfect apparatus*, in Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science, 1998, pp. 503–509.

[32] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, Cambridge, UK, 2000.

[33] M. Parnas and D. Ron, *Testing the diameter of graphs*, Random Structures Algorithms, 20 (2002), pp. 165–183.

[34] D. Ron, *Property testing*, in Handbook of Randomized Computing, Comb. Optim. 9, S. Rajasekaran, P. M. Pardalos, J. H. Reif, and J. D. P. Rolim, eds., Kluwer Academic Publishers, Dordrecht, The Netherlands, 2001, pp. 597–649.

[35] R. Rubinfeld and M. Sudan, *Robust characterizations of polynomials with applications to program testing*, SIAM J. Comput., 25 (1996), pp. 252–271.

[36] P. W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM J. Comput., 26 (1997), pp. 1484–1509.

[37] D. R. Simon, *On the power of quantum computation*, SIAM J. Comput., 26 (1997), pp. 1474–1483.

[38] R. Solovay and A. Yao, *Quantum Circuit Complexity and Universal Quantum Turing Machines*, manuscript, 1996.

[39] M. Szegedy, *private communication*, 1999.

[40] A. C.-C. Yao, *Probabilistic computations: Toward a unified measure of complexity*, in Proceedings of the 18th Annual IEEE Symposium on Foundations of Computer Science, 1977, pp. 222–227.

SIAM J. COMPUT.
Vol. 37, No. 5, pp. 1401–1428

COMPRESSION IN FINITE FIELDS AND TORUS-BASED CRYPTOGRAPHY[*]

K. RUBIN[†] AND A. SILVERBERG[†]

This paper is dedicated to the memory of the cat Ceilidh

Abstract. We present efficient compression algorithms for subgroups of multiplicative groups of finite fields, we use our compression algorithms to construct efficient public key cryptosystems called $\mathbb{T}_2$ and CEILIDH, we disprove some conjectures, and we use the theory of algebraic tori to give a better understanding of our cryptosystems, the Lucas-based, XTR, and Gong–Harn cryptosystems, and conjectured generalizations.

Key words. multiplicative groups, compression, torus-based cryptography, CEILIDH

AMS subject classifications. 94A60, 68P25, 14G50, 11T71

DOI. 10.1137/060676155

1. Introduction.

In this paper we present efficient compression algorithms for the elements of the subgroup of order $q^2 - q + 1$ in $\mathbb{F}_{q^6}^\times$, the multiplicative group of the finite field with q^6 elements, and for the elements of the subgroup of order $q + 1$ in $\mathbb{F}_{q^2}^\times$. We use our compression algorithms to create efficient public key cryptosystems, called CEILIDH and $\mathbb{T}_2$. We also disprove some conjectures from [4] about efficient compression in $\mathbb{F}_{q^n}^\times$. In addition, we show that our compression algorithms, Lucas-based, XTR, and Gong–Harn compression, and conjectural generalizations rely on the mathematical properties of algebraic tori, which are concepts from algebraic geometry that are generalizations of the multiplicative group of a field. We believe that understanding the mathematics that underlies the associated cryptosystems is a useful aid to better understand their properties and their security.

Let $\Phi_n(x)$ denote the nth cyclotomic polynomial, i.e., the monic polynomial in $\mathbb{Z}[x]$ of degree $\varphi(n)$ whose complex roots are exactly the primitive nth roots of unity. The multiplicative group $\mathbb{F}_q^\times = \mathbb{F}_q - \{0\}$ is a cyclic group of order $q - 1 = \Phi_1(q)$. Note that

$$x^n - 1 = \prod_{d|n} \Phi_d(x), \quad \text{so} \quad |\mathbb{F}_{q^n}^\times| = q^n - 1 = \prod_{d|n} \Phi_d(q).$$

For example,

$$|\mathbb{F}_{q^2}^\times| = q^2 - 1 = (q+1)(q-1) = \Phi_2(q)\Phi_1(q),$$

$$|\mathbb{F}_{q^6}^\times| = q^6 - 1 = (q^2 - q + 1)(q^2 + q + 1)(q + 1)(q - 1) = \Phi_6(q)\Phi_3(q)\Phi_2(q)\Phi_1(q).$$

[*] Received by the editors November 28, 2006; accepted for publication (in revised form) July 9, 2007; published electronically January 16, 2008. Preliminary versions of parts of this paper appeared in the proceedings of Crypto 2003 [28], the conference in honor of the 60th birthday of Hugh Cowie Williams [29], and ANTS VI [30]. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.

http://www.siam.org/journals/sicomp/37-5/67615.html

[†] Mathematics Department, University of California, Irvine, 103 MSTB, Irvine, CA 92697-3875 (krubin@uci.edu, asilverb@uci.edu). The first author's research was partially supported by NSF grant DMS-0457481. The second author's research was partially supported by NSA grants H98230-05-1-0044 and H98230-07-1-0039.

Let $G_{q,n}$ denote the subgroup of $\mathbb{F}_{q^n}^\times$ of order $\Phi_n(q)$.

In Diffie–Hellman key agreement, a finite field $\mathbb{F}_q$ and an element $g \in G_{q,1} = \mathbb{F}_q^\times$ are public. Alice (resp., Bob) transmits g^a (resp., g^b), where a (resp., b) is Alice's (resp., Bob's) secret. Then Alice and Bob share the secret $g^{ab} = (g^a)^b = (g^b)^a$.

When doing cryptography in the multiplicative group of a finite field $\mathbb{F}_{q^n}$, mathematically one is taking the $\mathbb{F}_{q^n}$-points of the multiplicative group $\mathbb{G}_m$, which is the same as the $\mathbb{F}_q$-points of the restriction of scalars $\mathrm{Res}_{\mathbb{F}_{q^n}/\mathbb{F}_q}\mathbb{G}_m$. This restriction of scalars decomposes (up to isogeny) as a product of algebraic tori that we will denote $\mathbb{T}_d$, one for each divisor d of n. Thus when doing cryptography in $\mathbb{F}_{q^n}^\times$, one is reduced to studying the tori $\mathbb{T}_d$. The torus $\mathbb{T}_d$ is an algebraic group over $\mathbb{F}_q$ of dimension $\varphi(d)$ whose $\mathbb{F}_q$-points form the group $G_{q,d}$ defined above. Being an algebraic torus just means that over an extension field (in this case, $\mathbb{F}_{q^d}$) the algebraic variety is isomorphic to a product of copies of the multiplicative group $\mathbb{G}_m$. Since $\mathbb{T}_d(\mathbb{F}_q) \cong G_{q,d} \subseteq \mathbb{F}_{q^d}^\times$, the subgroup $\mathbb{T}_d(\mathbb{F}_q)$ is subject to index calculus attacks on $\mathbb{F}_{q^d}^\times$; so if $d < n$, then $\mathbb{T}_d$ does not inherit the full security of $\mathbb{F}_{q^n}^\times$. Since almost no element of $\mathbb{T}_n(\mathbb{F}_q)$ lies in a proper subfield of $\mathbb{F}_{q^n}$, the torus $\mathbb{T}_n$ can be viewed as the cryptographically most significant part of $\mathbb{F}_{q^n}^\times$.

Since $\dim(\mathbb{T}_n) = \varphi(n)$, when the transmitted information comes from the group $G_{q,n} = \mathbb{T}_n(\mathbb{F}_q)$ one would hope to be able to compress transmissions down to $\varphi(n)\log q$ bits, rather than the $n \log q$ bits one must use for arbitrary elements of $\mathbb{F}_{q^n}^\times$. In other words, one would like to find an efficiently computable "compression" function f, defined on almost all of $G_{q,n}$, with values in $\mathbb{F}_q^{\varphi(n)}$, such that

(i) $f(h)$ and a determine $f(h^a)$,

(ii) $f(g)$ and $f(h)$ determine $f(gh)$,

(iii) f has an efficiently computable inverse j (a "decompression" map), defined on almost all of $\mathbb{F}_q^{\varphi(n)}$.

This would improve the efficiency of transmissions of group elements for discrete log-based cryptography on $\mathbb{F}_{q^n}^\times$ by a factor of $n/\varphi(n)$.

We represent this with a diagram:

(1.1)
$$\begin{tikzcd}
\mathbb{F}_q^{\varphi(n)} \arrow[r, dashed, bend left, "j"] & G_{q,n} \arrow[l, dashed, bend left, "f"]
\end{tikzcd}$$

where the dotted arrows signify that f and j need not be defined everywhere; they might be undefined on a "small" number of elements.

Whenever one has a compression map f with a corresponding decompression map j as above, the following protocols give generalized Diffie–Hellman key agreement and ElGamal encryption and signature schemes for the group $G_{q,n}$. Note that such maps f and j allow one to compress and decompress transmissions not only for Diffie–Hellman and ElGamal, but also for any cryptosystem whose security relies on the difficulty of the discrete logarithm problem in the multiplicative group $\mathbb{F}_{q^n}^\times$.

Choose an element $g \in G_{q,n}$ whose order ℓ is divisible by a large prime number (having chosen a prime power q such that $\Phi_n(q)$ has a large prime divisor).

Torus-based Diffie–Hellman key agreement:

Alice chooses an integer a randomly in the interval $[1, \ell - 1]$. Similarly, Bob chooses a random integer b from the same range.

- Alice sends $P_A := f(g^a) \in \mathbb{F}_q^{\varphi(n)}$ to Bob.

- Bob sends $P_B := f(g^b) \in \mathbb{F}_q^{\varphi(n)}$ to Alice.
- They share $(j(P_B))^a = g^{ab} = (j(P_A))^b$, and also $f(g^{ab})$.

Torus-based ElGamal encryption:
Alice's private key: An integer a, random in the interval $[1, \ell - 1]$.
Alice's public key: $P_A := f(g^a) \in \mathbb{F}_q^{\varphi(n)}$.

- Bob represents the message M in $\langle g \rangle$ and picks a random r between 1 and $\ell - 1$. The ciphertext is (c, d), where $c = f(g^r)$ and $d = f(M \cdot j(P_A)^r)$.
- To decrypt a ciphertext (c, d), Alice computes $M = j(d) \cdot j(c)^{-a}$.

As pointed out by a referee, in practice one would use hybrid encryption rather than textbook ElGamal, in which case a symmetric encryption key would be derived from $f(j(P_A)^r)$.

Torus-based ElGamal signatures:
Fix a cryptographic hash function $H : \{0,1\}^* \to \mathbb{Z}/\ell\mathbb{Z}$ (i.e., the function is easy to compute but hard to invert) and a key derivation function $h : \mathbb{F}_q^{\varphi(n)} \to \mathbb{Z}/\ell\mathbb{Z}$.
Alice's private key: An integer a, random in the interval $[1, \ell - 1]$.
Alice's public key: $P_A := f(g^a) \in \mathbb{F}_q^{\varphi(n)}$.

- To sign a message $M \in \{0,1\}^*$, Alice chooses a random integer r between 1 and $\ell - 1$ with $\gcd(r, \ell) = 1$. Alice's signature on M is (c, d), where $c = f(g^r) \in \mathbb{F}_q^{\varphi(n)}$ and $d = r^{-1}(H(M) - ah(c)) \pmod{\ell}$.
- Bob accepts Alice's signature if and only if
$$g^{H(M)} = j(P_A)^{h(c)} \cdot j(c)^d.$$

The signature length is $\varphi(n)\log_2(q) + \log_2(\ell)$ bits, as opposed to $n\log_2(q) + \log_2(\ell)$ bits in the classical ElGamal signature scheme over $\mathbb{F}_{q^n}$.

Examples of compression functions f that satisfy (i) above (but not (ii) or (iii)) are the trace functions used in the XTR and Lucas-based cryptosystems, which we now recall. (See also [19, 2].)

Lucas-based cryptosystems [25, 39, 40, 34, 35, 3], including LUC, are based on Lucas functions [23]. One way to interpret them is that they compress elements of $G_{q,2} \subset \mathbb{F}_{q^2}^\times$ using the trace map $\mathrm{Tr} : \mathbb{F}_{q^2} \to \mathbb{F}_q$ defined by $\mathrm{Tr}(x) = x + x^q$. In Lucas-based key agreement, Alice and Bob transmit $\mathrm{Tr}(g^a)$ and $\mathrm{Tr}(g^b)$, respectively, where $g \in G_{q,2}$. It turns out that Alice and Bob each have enough information to reconstruct $\mathrm{Tr}(g^{ab})$. Each party transmits only one element of $\mathbb{F}_q$, rather than one element of $\mathbb{F}_{q^2}$, thereby doubling the efficiency over Diffie–Hellman per unit of security against attacks on the discrete log problem in $\langle g \rangle \subset \mathbb{F}_{q^2}^\times$.

The Gong–Harn cryptosystem [10], which is based on linear feedback shift registers, can be viewed as using two symmetric functions to compress elements of $G_{q,3} \subset \mathbb{F}_{q^3}^\times$, namely, the trace map $\mathrm{Tr} : \mathbb{F}_{q^3} \to \mathbb{F}_q$ defined by $\mathrm{Tr}(x) = x + x^q + x^{q^2}$ and the map $\sigma_2 : \mathbb{F}_{q^3} \to \mathbb{F}_q$ defined by $\sigma_2(x) = x \cdot x^q + x \cdot x^{q^2} + x^q \cdot x^{q^2}$. These are two of the three symmetric functions on $\{x, x^q, x^{q^2}\}$; the third is the norm map $x \mapsto x \cdot x^q \cdot x^{q^2}$, which sends $G_{q,3}$ to 1. In Gong–Harn key agreement, Alice (resp., Bob) transmits $(\mathrm{Tr}(g^a), \sigma_2(g^a))$ (resp., $(\mathrm{Tr}(g^b), \sigma_2(g^b))$), where $g \in G_{q,3}$. It turns out that Alice and Bob each have enough information to reconstruct $\mathrm{Tr}(g^{ab})$ and $\sigma_2(g^{ab})$. Each party transmits only two elements of $\mathbb{F}_q$, rather than one element of $\mathbb{F}_{q^3}$, thereby improving efficiency over Diffie–Hellman by a factor of $3/2 = 3/\varphi(3)$ per unit of security against attacks on the discrete log problem in $\langle g \rangle \subset \mathbb{F}_{q^3}^\times$.

Brouwer, Pellikaan, and Verheul [5] and XTR [21] use the trace map $\mathrm{Tr} : \mathbb{F}_{q^6} \to \mathbb{F}_{q^2}$ defined by $\mathrm{Tr}(x) = x + x^{q^2} + x^{q^4}$ to compress elements of $G_{q,6} \subset \mathbb{F}_{q^6}^\times$. In XTR key agreement, Alice and Bob transmit $\mathrm{Tr}(g^a)$ and $\mathrm{Tr}(g^b)$, respectively, where $g \in G_{q,6}$. It turns out that they each have enough information to reconstruct a shared secret $\mathrm{Tr}(g^{ab})$. Each party transmits only one element of $\mathbb{F}_{q^2}$, rather than one element of $\mathbb{F}_{q^6}$, thereby tripling the efficiency over Diffie–Hellman per unit of security against attacks on the discrete log problem in $\langle g \rangle \subset \mathbb{F}_{q^6}^{\times}$. Brouwer, Pellikaan, and Verheul [5] asked whether this can be extended to larger n to represent elements of $G_{q,n}$ by $\varphi(n)$ elements of $\mathbb{F}_q$. In [4], Bosma, Hutton, and Verheul state precise conjectures on extending the above systems to larger n.

In XTR, the Gong–Harn cryptosystem, and the Lucas-based cryptosystems, Alice can compute $f(g^{ab})$ from $f(g^b)$ and a, for a suitable f coming from symmetric functions. In other words, these cryptosystems can exponentiate, as is needed for doing (analogues of) Diffie–Hellman. However, they cannot multiply in a straightforward way, as is needed for a direct use of ElGamal, since, for example, $\mathrm{Tr}(g)$ and $\mathrm{Tr}(h)$ do not determine $\mathrm{Tr}(gh)$. For example, for XTR, $\mathrm{Tr}(h) = \mathrm{Tr}(h^{q^2})$ for every h, but it is not the case in general that $\mathrm{Tr}(hg) = \mathrm{Tr}(h^{q^2}g)$ for all $g, h \in G_{q,6}$. However, if one orders the Galois conjugates and transmits a couple of extra bits to specify which conjugate has been chosen, then one can reconstruct an element of $G_{q,6}$ from its trace.

In sections 2–3 below we present our compression algorithms. We construct explicit maps f and j as in (1.1) when $n = 2$ and 6 and obtain the $\mathbb{T}_2$ and CEILIDH (or $\mathbb{T}_6$) cryptosystems. We show that they can be explained and implemented in an elementary way without any knowledge of algebraic geometry or algebraic tori (only basic definitions of finite fields are required).

We give background on algebraic tori in section 4 and study the algebraic tori $\mathbb{T}_n$ in section 5. In section 6 we consider rationality results and conjectures for the tori $\mathbb{T}_n$, since whenever the torus $\mathbb{T}_n$ is rational over $\mathbb{F}_q$, compression and decompression maps f and j exist for $G_{q,n}$. In particular, we explain the mathematics that we used to obtain the CEILIDH compression algorithm and prove that it works. We briefly mention stable rationality in section 7. In section 8 we discuss security considerations.

In section 9.1 we study group actions on tori, in order to give in sections 9.2 and 10 a deeper mathematical understanding of the Lucas-based systems, XTR, Gong–Harn, and the Bosma–Hutton–Verheul conjectural cryptosystems of [4]. We define an action of certain symmetric groups on the tori $\mathbb{T}_n$ and show (with S_e denoting the symmetric group on e letters) that

- the Lucas-based cryptosystems are "based on" the quotient variety $\mathbb{T}_2/S_2$,
- the Gong–Harn cryptosystem is based on the quotient variety $\mathbb{T}_3/S_3$,
- XTR is based on the quotient variety $\mathbb{T}_6/S_3$, and
- conjectural cryptosystems of Bosma–Hutton–Verheul would rely on the quotient varieties $\mathbb{T}_{30}/(S_3 \times S_5)$ or $\mathbb{T}_{30}/(S_2 \times S_3 \times S_5)$.

These quotient varieties are *not* groups. This is why the Lucas-based systems, Gong–Harn, and XTR do not have straightforward multiplication. However,

- Diffie–Hellman is based on the algebraic group (and algebraic torus) $\mathbb{T}_1 = \mathbb{G}_m$,
- the $\mathbb{T}_2$-cryptosystem is based on the algebraic group (and algebraic torus) $\mathbb{T}_2$,
- CEILIDH is based on the algebraic group (and algebraic torus) $\mathbb{T}_6$, and
- the (sometimes conjectural) $\mathbb{T}_n$-cryptosystems are based on the algebraic group (and algebraic torus) $\mathbb{T}_n$.

We therefore called the $\mathbb{T}_n$-cryptosystems "torus-based cryptosystems." (Later authors used our terminology more generally to refer to any cryptosystem using the group $G_{q,n}$ for some q and n, even those based on quotients of tori.)

In section 10 we disprove conjectures from [4] and thereby show that symmetric polynomials are not the correct functions to use for compression in $G_{q,n}$ when n has at least 3 distinct prime divisors.

Security and parameter selection for CEILIDH are exactly the same as for XTR. The advantage of the CEILIDH (resp., $\mathbb{T}_2$) cryptosystem over XTR (resp., LUC) is that CEILIDH and $\mathbb{T}_2$ make full use of the multiplication in the group $G_{q,n}$ (for $n = 6$ and 2). This is especially useful for signature schemes. However, XTR and LUC have computational efficiency advantages over CEILIDH and $\mathbb{T}_2$ (key agreement can be performed with fewer operations). See [11] for a comparison of CEILIDH and XTR.

Since the pairings in pairing-based cryptography take values in the algebraic tori considered here, our torus-based cryptography techniques can be used to improve the efficiency of pairing-based cryptography by compressing pairing values [33, 12].

In [31] we study analogues in the setting of elliptic curves and abelian varieties.

2. $\mathbb{T}_2$ compression and the $\mathbb{T}_2$-cryptosystem. Let $n = 2$ and let q be a prime power. One can write $\mathbb{F}_{q^2} = \mathbb{F}_q(\delta)$ for some $\delta \in \mathbb{F}_{q^2}^\times$ with $D := \delta^2 \in \mathbb{F}_q^\times$ if q is odd and $D := \delta^2 + \delta \in \mathbb{F}_q^\times$ if q is even. Since $\delta^q = -\delta$ if q is odd and $\delta^q = \delta + 1$ if q is even, we have

$$\begin{aligned} G_{q,2} &= \{a + b\delta : a, b \in \mathbb{F}_q \text{ and } (a + b\delta)^{q+1} = 1\} \\ &= \begin{cases} \{a + b\delta : a, b \in \mathbb{F}_q \text{ and } a^2 - Db^2 = 1\} & \text{if } q \text{ is odd,} \\ \{a + b\delta : a, b \in \mathbb{F}_q \text{ and } a^2 + Db^2 + ab = 1\} & \text{if } q \text{ is even.} \end{cases} \end{aligned}$$

Hilbert's Theorem 90 leads naturally to the following maps f and j. Define a compression map

$$f : G_{q,2} - \{1, -1\} \to \mathbb{F}_q \qquad \text{by} \qquad f(c + d\delta) = \frac{1 + c}{d},$$

and define a decompression map

$$j : \mathbb{F}_q \to G_{q,2} \quad \text{by} \quad j(a) = \frac{a + \delta}{a + \delta^q} = \begin{cases} \frac{a+\delta}{a-\delta} & \text{if } q \text{ is odd,} \\ \frac{a+\delta}{a+\delta+1} & \text{if } q \text{ is even.} \end{cases}$$

It is easy to check that f and j are inverse maps where they are defined, and if $a, b \in \mathbb{F}_q$ and $a \neq -b$ (resp., $a \neq b + 1$), then

$$\begin{aligned} j(a)j(b) &= j\left(\tfrac{ab+D}{a+b}\right) \quad \text{if } q \text{ is odd,} \\ j(a)j(b) &= j\left(\tfrac{ab+D}{a+b+1}\right) \quad \text{if } q \text{ is even.} \end{aligned}$$

To do $\mathbb{T}_2$-cryptography, use f to represent the elements of $G_{q,2} - \{1, -1\}$ in $\mathbb{F}_q$, and do all multiplications and exponentiations directly in $\mathbb{F}_q$ (without needing to use j), using the operation on (most of) $\mathbb{F}_q$:

$$a * b = \frac{ab + D}{a + b}, \qquad \text{respectively,} \qquad a * b = \frac{ab + D}{a + b + 1}$$

if q is odd, respectively, even.

3. CEILIDH compression and the CEILIDH public key system. The acronym *CEILIDH* (pronounced "cayley," like the Scottish Gaelic word ceilidh) stands for *Compact, Efficient, Improves on LUC, Improves on Diffie–Hellman.* The CEILIDH key agreement (resp., encryption, resp., signature) scheme is torus-based Diffie–Hellman (resp., ElGamal encryption, resp., ElGamal signatures) in the case $n = 6$.

3.1. CEILIDH compression algorithm. When $n = 6$, we can generate explicit examples of maps f and j at will. Next we give our algorithm for doing so. In section 6 below we will give a proof that it works and explain the mathematics behind it.

For a polynomial h in two variables with coefficients in $\mathbb{F}_q$, let

$$V(h) = \{(a, b) \in \mathbb{F}_q^2 : h(a, b) = 0\}.$$

Fix a prime power q. Fix $x \in \mathbb{F}_{q^2} - \mathbb{F}_q$, so $\mathbb{F}_{q^2} = \mathbb{F}_q(x)$, and choose a basis $\{\alpha_1, \alpha_2, \alpha_3\}$ of $\mathbb{F}_{q^3}$ over $\mathbb{F}_q$. Then $\{\alpha_1, \alpha_2, \alpha_3, x\alpha_1, x\alpha_2, x\alpha_3\}$ is a basis of $\mathbb{F}_{q^6}$ over $\mathbb{F}_q$. Let $\sigma \in \mathrm{Gal}(\mathbb{F}_{q^6}/\mathbb{F}_q)$ be the element of order 2, i.e., $\sigma(z) = z^{q^3}$. Define a map $j_0 : \mathbb{F}_q^3 \hookrightarrow \mathbb{F}_{q^6}^\times$ by

$$j_0(u, v, w) = \frac{\gamma + x}{\gamma + \sigma(x)},$$

where $\gamma = u\alpha_1 + v\alpha_2 + w\alpha_3$. Let

$$U = \{(u, v, w) \in \mathbb{F}_q^3 : N_{\mathbb{F}_{q^6}/\mathbb{F}_{q^2}}(j_0(u, v, w)) = 1\}.$$

A calculation in Mathematica shows that U is a hypersurface in $\mathbb{F}_q^3$ defined by a quadratic equation in u, v, w. Fix a point $\beta = (\beta_1, \beta_2, \beta_3) \in U(\mathbb{F}_q)$. Adjust the basis $\{\alpha_1, \alpha_2, \alpha_3\}$ if necessary, to ensure that the tangent plane at β to the surface U is $u = \beta_1$. If $(a, b) \in \mathbb{F}_q \times \mathbb{F}_q$, then the intersection of U with the line $\beta + t(1, a, b)$ consists of two points, namely, β and a point $g(a, b) \in U$ of the form $\beta + \frac{1}{h(a,b)}(1, a, b)$, where $h(a, b) \in \mathbb{F}_q[a, b]$ is an explicit polynomial that can be computed using Mathematica. The map g is an isomorphism

$$g : \mathbb{F}_q^2 - V(h) \xrightarrow{\sim} U - \{\beta\},$$

and $j_0 \circ g$ defines an isomorphism

$$j : \mathbb{F}_q^2 - V(h) \xrightarrow{\sim} G_{q,6} - \{1, j_0(\beta)\}.$$

For the inverse isomorphism, suppose that $t = c + dx \in G_{q,6} - \{1, j_0(\beta)\}$ with $c, d \in \mathbb{F}_{q^3}$. Write $(1 + c)/d = u\alpha_1 + v\alpha_2 + w\alpha_3$ with $u, v, w \in \mathbb{F}_q$, and define

$$f(t) = \left(\frac{v - \beta_2}{u - \beta_1}, \frac{w - \beta_3}{u - \beta_1}\right).$$

Then $f : G_{q,6} - \{1, j_0(\beta)\} \xrightarrow{\sim} \mathbb{F}_q^2 - V(h)$ satisfies $f \circ j = \mathrm{id}$ and $j \circ f = \mathrm{id}$.

3.2. Explicit examples of maps *f* and *j*. Using the above algorithm, we produce explicit examples, where ζ_m denotes an mth root of unity in $\bar{\mathbb{F}}_q$.

Example 3.1. To ensure that $\mathbb{F}_{q^6} = \mathbb{F}_q(\zeta_9)$, restrict to prime powers $q \equiv 2$ or 5 (mod 9). Let $x = \zeta_3$ and let $(\alpha_1, \alpha_2, \alpha_3) = (1, \zeta_9 + \zeta_9^{-1}, \zeta_9^2 + \zeta_9^{-2})$. The hypersurface

U is given by the quadratic equation $u^2 - u - v^2 + vw - w^2 = 0$. Let $\beta = (0, 0, 0)$. The above algorithm gives a map $j : \mathbb{F}_q^2 \to G_{q,6}$ defined by

$$j(a, b) = (r + s\zeta_3)/(r + s\zeta_3^2),$$

where

$$r = 1 + a(\zeta_9 + \zeta_9^{-1}) + b(\zeta_9^2 + \zeta_9^{-2}), \quad s = h(a, b) = 1 - a^2 - b^2 + ab,$$

and a map $f : G_{q,6} - \{1, \zeta_3^2\} \xrightarrow{\sim} \mathbb{F}_q^2 - V(h)$ defined by $f(t) = (v/u, w/u)$, where $t = c + d\zeta_3$ with $c, d \in \mathbb{F}_{q^3}$ and $(1 + c)/d = u + v(\zeta_9 + \zeta_9^{-1}) + w(\zeta_9^2 + \zeta_9^{-2})$ with $u, v, w \in \mathbb{F}_q$.

Example 3.2. In order to ensure that $\mathbb{F}_{q^6} = \mathbb{F}_q(\zeta_7)$, restrict to prime powers $q \equiv 3$ or $5 \pmod 7$. We can then let $x = \sqrt{-7}$, $\beta = (1, 0, 2)$, and $(\alpha_1, \alpha_2, \alpha_3) = (1, \zeta_7 + \zeta_7^{-1}, \zeta_7^2 + \zeta_7^{-2} + 1)$. The above algorithm outputs a map $j : \mathbb{F}_q^2 \to G_{q,6}$ defined by $j(a, b) = (r + s\sqrt{-7})/(r - s\sqrt{-7})$, where

$$s = h(a, b) = (2a^2 + b^2 - ab + 2a - 4b - 3)/14,$$

$$r = h(a, b) + 1 + a(\zeta_7 + \zeta_7^{-1}) + (2h(a, b) + b)(\zeta_7^2 + \zeta_7^{-2} + 1),$$

and a map $f : G_{q,6} - \{1, \zeta_7^2\} \xrightarrow{\sim} \mathbb{F}_q^2 - V(h)$ defined by

$$f(t) = \left(\frac{v}{u - 1}, \frac{w - 2}{u - 1}\right),$$

where $t = c + d\sqrt{-7}$ with $c, d \in \mathbb{F}_{q^3}$ and $(1 + c)/d = u + v(\zeta_7 + \zeta_7^{-1}) + w(\zeta_7^2 + \zeta_7^{-2} + 1)$ with $u, v, w \in \mathbb{F}_q$. Here U is defined by $3u^2 - 2uv - 2v^2 + 4uw + vw - w^2 = 7$.

Example 3.3. Let q be an odd prime power congruent to 2, 6, 7, or 11 (mod 13), and let $z = \zeta_{13} + \zeta_{13}^{-1}$. Then $\mathbb{F}_{q^{12}} = \mathbb{F}_q(\zeta_{13})$ and $\mathbb{F}_{q^6} = \mathbb{F}_q(z)$. Let $x = \sqrt{13}$, let $\beta = (-1, 0, 3)$, let $y = \zeta_{13} + \zeta_{13}^{-1} + \zeta_{13}^5 + \zeta_{13}^{-5} \in \mathbb{F}_{q^3}$, and let $(\alpha_1, \alpha_2, \alpha_3) = (y^2, y + \frac{y^2}{2}, 1)$. The above algorithm outputs a map $j : \mathbb{F}_q^2 \to G_{q,6}$ defined by

$$j(a, b) = (r - s\sqrt{13})/(r + s\sqrt{13}),$$

where

$$r = (3(a^2 + b^2) + 7ab + 34a + 18b + 40)y^2 + 26ay$$

$$- (21a(3 + b) + 9(a^2 + b^2) + 28b + 42),$$

$$s = 3(a^2 + b^2) + 7ab + 21a + 18b + 14,$$

and a map $f : G_{q,6} - \{1, -2z^5 + 6z^3 - 4z - 1\} \to \mathbb{F}_q^2$ defined by

$$f(t) = \left(\frac{v}{u + 1}, \frac{w - 3}{u + 1}\right),$$

where $t = c + d\sqrt{13}$ with $c, d \in \mathbb{F}_{q^3}$ and $(1 + c)/d = uy^2 + v(y + \frac{y^2}{2}) + w$ with $u, v, w \in \mathbb{F}_q$. Here U is defined by $14u^2 + 21uv + 3v^2 + 18uw + 7vw + 3w^2 = -13$.

4. Algebraic tori. In this section we briefly introduce algebraic tori in order to explain the mathematics underlying compression algorithms for $G_{q,n} \subseteq \mathbb{F}_{q^n}^\times$.

If M/k is a finite Galois extension and V is a variety defined over M, write $\mathrm{Res}_{M/k}V$ for the Weil restriction of scalars of V from M to k. Then $\mathrm{Res}_{M/k}V$ is a variety defined over k together with a morphism

$$\eta : \mathrm{Res}_{M/k}V \to V \tag{4.1}$$

defined over M that induces an isomorphism

$$\eta : (\mathrm{Res}_{M/k}V)(k) \xrightarrow{\sim} V(M). \tag{4.2}$$

A precise technical definition is that the restriction of scalars $\mathrm{Res}_{M/k}V$ is uniquely defined by the universal property that for every scheme X over k (and therefore every variety X over k) and every morphism $f : X \to V$, there exists a unique morphism $f_0 : X \to \mathrm{Res}_{M/k}V$ such that $\eta \circ f_0 = f$. See section 1.3 of [38] or section 3.12 of [36] for more on the restriction of scalars.

If V is an algebraic variety and D is a finite set, write

$$V^D := \bigoplus_{\delta \in D} V \cong V^{|D|}.$$

If D is a group, then D acts on V^D by permuting the summands. Let $\mathbb{A}^d$ denote d-dimensional affine space (so $\mathbb{A}^d(k) = k^d$), and let $\mathbb{A}^D := (\mathbb{A}^1)^D$.

If V is defined over k and $\Gamma = \mathrm{Gal}(M/k)$, then the morphism η of (4.1) induces an isomorphism

$$\bigoplus_{\gamma \in \Gamma} \eta^\gamma : \mathrm{Res}_{M/k}V \xrightarrow{\sim} V^\Gamma \tag{4.3}$$

defined over M (see section 1.3 of [38]), where $\eta^\gamma : \mathrm{Res}_{M/k}V \to V$ is the morphism defined by applying γ to the coefficients of the rational functions that define η.

Let $\mathbb{G}_m$ denote the multiplicative group over a field k. Then $\mathbb{G}_m$ $(\subset \mathbb{A}^1)$ is an algebraic group over k such that $\mathbb{G}_m(F) = F^\times$ for all extension fields F of k.

Definition 4.1. *An* algebraic torus *over a field k is an algebraic group over k that over some larger field is isomorphic to a product of copies of $\mathbb{G}_m$. A field over which the torus becomes isomorphic to a product of multiplicative groups is called a* splitting field *for the torus; one says that the torus* splits *over that field.*

Good references for algebraic tori are [26, 36].

Example 4.2. (i) For every positive integer n, $\mathbb{G}_m^n$ is an n-dimensional algebraic torus.

(ii) If L/k is an extension of degree n, then $\mathrm{Res}_{L/k}\mathbb{G}_m$ is an n-dimensional algebraic torus over k that splits over L (by (4.3) with $V = \mathbb{G}_m$).

5. The algebraic tori $\mathbb{T}_{L/k}$ and $\mathbb{T}_n$. Next we define the algebraic tori that underlie the XTR, Gong–Harn, Lucas-based, $\mathbb{T}_2$, and CEILIDH cryptosystems and give some of their basic properties.

Suppose L/k is a finite Galois extension and $n := [L : k]$ is squarefree. Suppose $k \subseteq F \subseteq L$, and let $G = \mathrm{Gal}(L/k)$, $H = \mathrm{Gal}(L/F)$, and $e = |H|$. For $1 \le i \le e$ let $\sigma_{i,F}$ denote the composition

$$\sigma_{i,F} : \mathrm{Res}_{L/F}\mathbb{A}^1 \xrightarrow{\sim} \mathbb{A}^H \longrightarrow \mathbb{A}^1, \tag{5.1}$$

where the first map is the isomorphism (defined over L) coming from (4.3) and the second map is the ith symmetric polynomial of the e projection maps $\mathbb{A}^H \to \mathbb{A}^1$. (Recall that the first symmetric polynomial of $x_1, \dots, x_e$ is $\sum_{i=1}^{e} x_i$, the second is $\sum_{i<j} x_i x_j$, and the eth is $\prod_{i=1}^{e} x_i$.)

The next lemma will be used to define the algebraic tori $\mathbb{T}_{L/k}$ and prove properties about them.

Lemma 5.1.

(i) *The maps* $\sigma_{i,k} : \mathrm{Res}_{L/k}\mathbb{A}^1 \longrightarrow \mathbb{A}^1$ *are defined over* k.

(ii) *For every* $1 \le i \le n$ *the following diagram is commutative:*

$$\begin{tikzcd}
(\mathrm{Res}_{L/k}\mathbb{A}^1)(k) \arrow[r, "\sigma_{i,k}"] \arrow[d, "\cong"'] & \mathbb{A}^1(k) \arrow[d, "\cong"'] \\
L \arrow[r, "\sigma_{i,k}"] & k
\end{tikzcd}$$

where the bottom map $\sigma_{i,k}$ *sends* $\alpha \in L$ *to the ith symmetric polynomial evaluated on the set of* G*-conjugates of* α*, the right map is the natural identification, and the left map is the composition of* (4.2) *with the natural identification* $\mathbb{A}^1(L) \cong L$.

Proof. Part (i) follows since symmetric functions are symmetric, while (ii) follows from the definitions and the fact that $(\eta(v))^\sigma = \eta^\sigma(v)$ for all $v \in (\mathrm{Res}_{L/k}\mathbb{A}^1)(k)$ and $\sigma \in \mathrm{Gal}(L/k)$. □

Lemma 5.1(ii) shows that $\sigma_{n,k}$ and $\sigma_{1,k}$ correspond to the usual norm and trace maps from $(\mathrm{Res}_{L/k}\mathbb{A}^1)(k) \cong L$ to k. Applying $\mathrm{Res}_{F/k}$ to (5.1) and using that $\mathrm{Res}_{L/k}\mathbb{A}^1 = \mathrm{Res}_{F/k}(\mathrm{Res}_{L/F}\mathbb{A}^1)$, we obtain maps

$$\tilde{\sigma}_{i,F} : \mathrm{Res}_{L/k}\mathbb{A}^1 \longrightarrow \mathrm{Res}_{F/k}\mathbb{A}^1 \tag{5.2}$$

for $1 \le i \le e$. Let $\mathrm{N}_{L/F,k} := \tilde{\sigma}_{e,F}$ and $\mathrm{Tr}_{L/F,k} := \tilde{\sigma}_{1,F}$.

Definition 5.2. *Define* $\mathbb{T}_{L/k}$ *by*

$$\mathbb{T}_{L/k} := \ker\left[\mathrm{Res}_{L/k}\mathbb{G}_m \xrightarrow{\oplus \mathrm{N}_{L/M,k}} \bigoplus_{k \subseteq M \subsetneq L} \mathrm{Res}_{M/k}\mathbb{G}_m\right].$$

Let $\mathbb{T}_n$ *(or* $\mathbb{T}_{n,q}$*) denote* $\mathbb{T}_{\mathbb{F}_{q^n}/\mathbb{F}_q}$.

By definition, $\mathbb{T}_{L/k}$ is a subvariety and algebraic subgroup of $\mathrm{Res}_{L/k}\mathbb{G}_m$, defined over k. When L/k is abelian but not cyclic, the algebraic group $\mathbb{T}_{L/k}$ has dimension zero (see Proposition 5.3 of [24]). Lemmas 5.4 and 5.6 below show that if L/k is cyclic, then $\mathbb{T}_{L/k}$ is isomorphic over L to $\mathbb{G}_m^{\varphi(n)}$, and thus $\mathbb{T}_{L/k}$ is an algebraic torus of dimension $\varphi(n)$ that splits over L. When L/k is cyclic, $\mathbb{T}_{L/k}$ is the variety V_L defined in section 5 of [24] with $V = \mathbb{G}_m$ (see Remark 5.11 of [24]). We first need some notation, which will also be used in sections 9–10.

Definition 5.3. *If* Γ *is a finite group and* Δ *is a subgroup, let* Γ/Δ *denote the coset space. For* $i = 1, \dots, |\Delta|$*, let* σ_i *denote the ith symmetric function for* $i = 1, \dots, |\Delta|$ *and define*

$$s_i : \mathbb{A}^\Gamma \to \mathbb{A}^{\Gamma/\Delta} \quad \text{by} \quad (\alpha_g)_{g\in\Gamma} \mapsto (\sigma_i(\{\alpha_\gamma : \gamma \in g\Delta\}))_{g\Delta \in \Gamma/\Delta}.$$

Let $\mathbb{N}_\Delta$ *be the restriction of* $s_{|\Delta|}$ *to* $\mathbb{G}_m^\Gamma$*, i.e.,*

$$\mathbb{N}_\Delta : \mathbb{G}_m^\Gamma \to \mathbb{G}_m^{\Gamma/\Delta}, \qquad (\alpha_g)_{g\in\Gamma} \mapsto \left(\prod_{\gamma \in g\Delta} \alpha_\gamma\right)_{g\Delta \in \Gamma/\Delta},$$

and let

$$\mathbb{T}_\Gamma := \ker\left[\mathbb{G}_m^\Gamma \xrightarrow{\oplus\mathbb{N}_\Delta} \bigoplus_{1\neq\Delta\subseteq\Gamma}\mathbb{G}_m^{\Gamma/\Delta}\right]$$

$$= \left\{(x_g)_{g\in\Gamma} : \prod_{h\in\Delta} x_{gh} = 1 \textit{ for all } g\in\Gamma \textit{ and all subgroups } \Delta\neq 1 \textit{ of } \Gamma\right\}.$$

Viewing $\mathbb{G}_m$ as an algebraic group over a field k, then $\mathbb{T}_\Gamma$ is an algebraic group over k. The next lemma, which we will use repeatedly, follows directly from the definitions of $\mathbb{T}_{L/k}$ and $\mathbb{T}_G$.

Lemma 5.4. *The isomorphism* $\mathrm{Res}_{L/k}\mathbb{G}_m \xrightarrow{\sim} \mathbb{G}_m^G$ *given by* (4.3) *(with* $V = \mathbb{G}_m$*) restricts to an isomorphism* $\mathbb{T}_{L/k} \xrightarrow{\sim} \mathbb{T}_G$ *(defined over L).*

The next result is used to prove Lemma 5.6 and Proposition 5.8 below. For a proof see, for example, Theorem 1 of [6] or Theorem 2 of [32]. We thank D. Bernstein and H. Lenstra for pointing out these references.

Lemma 5.5. *For every positive integer n,* $\Phi_n(x)$ *and the set*

$$\left\{\frac{x^n-1}{x^t-1} : t \mid n \textit{ and } 1\leq t\neq n\right\}$$

generate the same ideal of $\mathbb{Z}[x]$*.*

Lemma 5.6 is used to prove Theorems 5.7 and 10.9 below. Its proof can be ignored by the casual reader.

Lemma 5.6. *Suppose* Γ *is a cyclic group of squarefree order. Let* Ω *be the subset of* Γ *consisting of all generators of* Γ*. The projection map* $\mathbb{G}_m^\Gamma \twoheadrightarrow \mathbb{G}_m^\Omega$ *restricts to an isomorphism* $\mathbb{T}_\Gamma \xrightarrow{\sim} \mathbb{G}_m^\Omega$ *of algebraic groups over k.*

Proof. Let $m = |\Gamma|$. If Δ is a subgroup of Γ, let $N_\Delta := \sum_{h\in\Delta} h$. Let I denote the ideal of $\mathbb{Z}[\Gamma]$ generated by $\{N_\Delta : \Delta\neq 1 \text{ is a subgroup of } \Gamma\}$. The map $\mathrm{Hom}_{\mathbb{Z}}(\mathbb{Z}[\Gamma],\mathbb{G}_m)\to\mathbb{G}_m^\Gamma$ defined by $\phi\mapsto(\phi(g))_{g\in\Gamma}$ induces a commutative diagram

$$\begin{tikzcd}
\mathrm{Hom}(\mathbb{Z}[\Gamma]/I,\mathbb{G}_m) \arrow[r, hook] \arrow[d, "\cong"] & \mathrm{Hom}(\mathbb{Z}[\Gamma],\mathbb{G}_m) \arrow[r, two heads] \arrow[d, "\cong"] & \mathrm{Hom}(\oplus_{\gamma\in\Omega}\mathbb{Z}\gamma,\mathbb{G}_m) \arrow[d, "\cong"] \\
\mathbb{T}_\Gamma \arrow[r, hook] & \mathbb{G}_m^\Gamma \arrow[r, two heads] & \mathbb{G}_m^\Omega
\end{tikzcd}$$

where the vertical maps are group isomorphisms and the top and bottom rows are the natural maps. For each $g\in\Gamma$, let $\bar{g}$ denote its image in $\mathbb{Z}[\Gamma]/I$. Let τ denote a generator of Γ. Since Γ is cyclic, $\tau\mapsto x$ induces an isomorphism $\mathbb{Z}[\Gamma]\xrightarrow{\sim}\mathbb{Z}[x]/(x^m-1)\mathbb{Z}[x]$. By Lemma 5.5, this map induces an isomorphism $\mathbb{Z}[\Gamma]/I\xrightarrow{\sim}\mathbb{Z}[x]/\Phi_m(x)\mathbb{Z}[x]\cong\mathbb{Z}[\zeta_m]$ that sends τ to ζ_m. Since m is squarefree, the primitive mth roots of unity form a $\mathbb{Z}$-basis for $\mathbb{Z}[\zeta_m]$ (see, for example, [22]), i.e., $\mathbb{Z}[\zeta_m] = \oplus_{a\in R}\mathbb{Z}\zeta_m^a$, where $R := (\mathbb{Z}/m\mathbb{Z})^\times$. It follows that $\mathbb{Z}[\Gamma]/I = \oplus_{a\in R}\mathbb{Z}\bar{\tau}^a = \oplus_{\gamma\in\Omega}\mathbb{Z}\bar{\gamma}$. This says exactly that the natural group homomorphism $\oplus_{\gamma\in\Omega}\mathbb{Z}\gamma\to\mathbb{Z}[\Gamma]/I$ is an isomorphism. Therefore, the composition in the top line of the commutative diagram is an isomorphism. Thus the composition in the bottom line of the diagram is an isomorphism, as desired. □

If V and W are algebraic groups over k, a homomorphism $f: V\to W$ is an *isogeny* over k if f is surjective and defined over k and $\dim(V) = \dim(W)$. If an isogeny between V and W exists, we say V and W are *isogenous* over k.

Theorem 5.7. *If L/k is a cyclic extension of degree n, then*

(i) $\mathbb{T}_{L/k}$ *is an algebraic torus of dimension* $\varphi(n)$ *that splits over* L*;*

(ii) *letting* $\mathrm{N}_{L/M}$ *denote the usual norm map from* L *to* M*, then*

$$\mathbb{T}_{L/k}(k) \cong \{\alpha \in L^\times : \mathrm{N}_{L/M}(\alpha) = 1 \textit{ for all } k \subseteq M \subsetneq L\}; \textit{ and}$$

(iii) $\mathrm{Res}_{L/k}\mathbb{G}_m$ *is isogenous over* k *to* $\oplus_M \mathbb{T}_{M/k}$*, where* M *runs over all intermediate extensions* $k \subseteq M \subseteq L$*.*

Proof. By Lemma 5.4, $\mathbb{T}_{L/k}$ is isomorphic over L to $\mathbb{T}_G$, which by Lemma 5.6 is isomorphic over k to $\mathbb{G}_m^{\varphi(n)}$. This gives (i). Part (ii) follows from Lemma 5.1(ii) with $i = n$. For (iii), see pp. 60–61 of [36] or Theorem 5.2 of [24]. □

Recall that $G_{q,n}$ is the subgroup of $\mathbb{F}_{q^n}^\times$ of order $\Phi_n(q)$.

PROPOSITION 5.8.

(i) $\mathbb{T}_n(\mathbb{F}_q) \cong G_{q,n}$.

(ii) $G_{q,n} = \{\alpha \in \mathbb{F}_{q^n}^\times : \mathrm{N}_{\mathbb{F}_{q^n}/\mathbb{F}_{q^t}}(\alpha) = 1$ *for all* $t|n$ *with* $t \neq n\}$.

(iii) $\#\mathbb{T}_n(\mathbb{F}_q) = \Phi_n(q)$.

Proof. The cyclic group $\mathrm{Gal}(\mathbb{F}_{q^n}/\mathbb{F}_q)$ is generated by the Frobenius automorphism $\alpha \mapsto \alpha^q$. Hence if t divides n, then $\mathrm{N}_{\mathbb{F}_{q^n}/\mathbb{F}_{q^t}}(\alpha) = \alpha^{(q^n-1)/(q^t-1)}$ for all $\alpha \in \mathbb{F}_{q^n}$. Thus by Theorem 5.7(ii),

$$\begin{aligned}\mathbb{T}_n(\mathbb{F}_q) &\cong \{\alpha \in \mathbb{F}_{q^n}^\times : \mathrm{N}_{\mathbb{F}_{q^n}/\mathbb{F}_{q^t}}(\alpha) = 1 \text{ for all } t|n \text{ with } t \neq n\} \\ &= \{\alpha \in \mathbb{F}_{q^n}^\times : \alpha^c = 1\},\end{aligned}$$

where $c = \gcd\{(q^n-1)/(q^t-1) : t \mid n \text{ and } t \neq n\}$. By Lemma 5.5, $c = \Phi_n(q)$. Now (i) and (ii) follow from the definition of $G_{q,n}$, and (iii) follows from (i). □

6. Rationality and the $\mathbb{T}_n$-cryptosystem.

We will recall what it means for a variety to be rational. This concept is useful since whenever an algebraic torus is rational, there exist compression and decompression maps. We give a mathematical explanation for why the torus $\mathbb{T}_6$ that underlies CEILIDH (and XTR) is rational that proves the correctness of the algorithm in section 3.1 and the formulas in section 3.2. We also discuss generalizing CEILIDH and XTR.

DEFINITION 6.1. *A* rational *map between algebraic varieties is a function defined by quotients of polynomials that is defined almost everywhere (i.e., on a Zariski open set). A* birational isomorphism *between algebraic varieties is a rational map that has a rational inverse (the maps are inverses wherever both are defined). A* d*-dimensional variety over* k *is* rational *over* k *if it is birationally isomorphic over* k *to* $\mathbb{A}^d$*.*

Note that birational isomorphisms of algebraic groups are not necessarily group isomorphisms. Further, rational maps are not necessarily functions—they might fail to be defined on a lower dimensional set.

If $\mathbb{T}_n$ is rational over k (i.e., birationally isomorphic over k to $\mathbb{A}^{\varphi(n)}$), then by Proposition 5.8(i), almost all elements of $G_{q,n}$ can be represented by $\varphi(n)$ elements of $\mathbb{F}_q$, and we obtain efficient "$\mathbb{T}_n$-cryptosystems" using the "torus-based" protocols given in the introduction.

The sets $G_{q,n}$ and $\mathbb{F}_q^{\varphi(n)}$ are of size approximately $q^{\varphi(n)}$. The "bad" sets where the maps f or j are not defined lie in algebraic subvarieties of dimension at most $\varphi(n) - 1$ and therefore have at most $cq^{\varphi(n)-1}$ elements for some constant c. Thus the probability that an element lands in the bad set is at worst c/q, which will be small for large q. In any given case the bad sets might be even smaller. In the examples in section 3, the maps j are defined on all of $\mathbb{F}_q^2$, and the maps f are defined at all but 2 elements of $G_{q,6}$.

Next we give the mathematics that proves that the algorithm of section 3.1 is correct. Suppose L/k is a cyclic degree 6 extension, and F_2 (resp., F_3) are the quadratic (resp., cubic) extensions of k in L:

$$\begin{tikzcd}
& L \arrow[dl, dash] \arrow[dr, dash] & \\
F_2 \arrow[dr, dash, "2"'] & & F_3 \arrow[dl, dash, "3"] \\
& k &
\end{tikzcd}$$

The one-dimensional algebraic torus $\mathbb{T}_{L/F_3}$ is, by definition, the kernel of the norm map $N_{L/F_3} : L \to F_3$. Let $\mathbb{T} := \mathrm{Res}_{F_3/k}(\mathbb{T}_{L/F_3})$. Then $\mathbb{T}$ is an algebraic torus over k of dimension 3. As in section 2, the torus $\mathbb{T}_{L/F_3}$, corresponding to the quadratic extension L/F_3, is rational over k (i.e., is birationally isomorphic over k to $\mathbb{A}^1$), and thus the torus $\mathbb{T}$ is rational over k (i.e., birationally isomorphic over k to $\mathbb{A}^3$). The two-dimensional torus $\mathbb{T}_{L/k}$ is the hypersurface cut out by the equation $\mathrm{N}_{L/F_2} = 1$ inside the torus $\mathbb{T}$, where N_{L/F_2} denotes the norm map from L to F_2. This hypersurface is defined by a quadratic equation that can be used to parametrize the hypersurface. When $k = \mathbb{F}_q$, then the above says that $\mathbb{T}_{6,q}$ is the two-dimensional subvariety of the three-dimensional torus $\mathrm{Res}_{\mathbb{F}_{q^3}/\mathbb{F}_q}(\mathbb{T}_{2,q^3})$ that is cut out by the equation $\mathrm{N}_{\mathbb{F}_{q^6}/\mathbb{F}_{q^2}} = 1$.

Fix $x \in F_2 - k$, so $F_2 = k(x)$, and choose a basis $\{\alpha_1, \alpha_2, \alpha_3\}$ of F_3 over k. Then $\{\alpha_1, \alpha_2, \alpha_3, x\alpha_1, x\alpha_2, x\alpha_3\}$ is a basis of L over k. Let $\sigma \in \mathrm{Gal}(L/k)$ be the element of order 2. Define a (one-to-one) map $j_0 : \mathbb{A}^3(k) \hookrightarrow L^\times$ by

$$j_0(u, v, w) = \frac{\gamma + x}{\gamma + \sigma(x)},$$

where $\gamma = u\alpha_1 + v\alpha_2 + w\alpha_3$. Then $N_{L/F_3}(j_0(\mathbf{u})) = 1$ for every $\mathbf{u} = (u, v, w)$. Let

$$U = \{\mathbf{u} \in \mathbb{A}^3 : N_{L/F_2}(j_0(\mathbf{u})) = 1\}.$$

By Definition 5.2, $j_0(\mathbf{u}) \in \mathbb{T}_{L/k}$ if and only if $u \in U$, so restricting j_0 to U gives a morphism

$$j_0 : U \to \mathbb{T}_{L/k} - \{1\}. \tag{6.1}$$

We will next define a birational map from $\mathbb{A}^2$ to U. A calculation in Mathematica shows that U is a hypersurface in $\mathbb{A}^3$ defined by a quadratic equation in u, v, w. Fix a point $\beta = (\beta_1, \beta_2, \beta_3) \in U(k)$. By adjusting the basis $\{\alpha_1, \alpha_2, \alpha_3\}$ if necessary, we can assume without loss of generality that the tangent plane at β to the surface U is the plane $u = \beta_1$. If $(a, b) \in k \times k$, then the intersection of U with the line $\beta + t(1, a, b)$ consists of two points, namely, β and $g(a, b) = \beta + \frac{1}{h(a,b)}(1, a, b)$ for some $h(a, b) \in k[a, b]$. The map g defines a morphism

$$g : \mathbb{A}^2 - V(h) \to U - \{\beta\}, \tag{6.2}$$

so $j_0 \circ g$ defines a morphism

$$j : \mathbb{A}^2 - V(h) \to \mathbb{T}_{L/k} - \{1, j_0(\beta)\}. \tag{6.3}$$

For the inverse, write $t = c + dx \in \mathbb{T}_{L/k}(k) - \{1, j_0(\beta)\}$ with $c, d \in F_3$. One checks easily that $d \neq 0$, and if $\gamma = (1+c)/d$, then $\gamma/\sigma(\gamma) = t$. Write $(1+c)/d = u\alpha_1 + v\alpha_2 + w\alpha_3$ with $u_i \in k$, and define

$$f(t) = \left(\frac{v - \beta_2}{u - \beta_1}, \frac{w - \beta_3}{u - \beta_1} \right).$$

It follows from the discussion above that $f : \mathbb{T}_{L/k} - \{1, j_0(\beta)\} \xrightarrow{\sim} \mathbb{A}^2 - V(h)$ satisfies $f \circ j = \mathrm{id}$ and $j \circ f = \mathrm{id}$, so (6.1), (6.2), and (6.3) are isomorphisms, and we obtain the following.

Theorem 6.2. *The above maps f and j induce inverse birational isomorphisms over k between $\mathbb{T}_{L/k}$ and $\mathbb{A}^2$.*

Note that in the examples in section 3.2, the coefficients of the rational maps f and j are independent of q.

Remark 6.3. While the choice of j_0 on first glance might look obvious, in fact replacing j_0 by the seemingly just as obvious $j_1(u, v, w) = (\gamma x + 1)/(\gamma\sigma(x) + 1)$ leads to a hypersurface U defined by a cubic, rather than a quadratic, that does not seem to easily lead to a parametrization, and thus does not easily lead to efficient functions f and j. This is especially relevant when trying to generalize to the case of $n = 30$, where it is not at all clear how to correctly choose a generalization of j_0.

Lenstra [20] asked whether XTR can be generalized to obtain more security (see also [5]). The next interesting case after $n = 6$ (i.e., the first case where $n/\varphi(n) > 6/\varphi(6) = 3$) is when $n = 30$, where finding efficient generalizations of the XTR or CEILIDH compression/decompression maps is an open question. (However, see the next section for other techniques.) The following problem is discussed in sections 5–6 of [36] and can be viewed as giving a general mathematical framework for the question of extending XTR and CEILIDH.

Voskresenskiĭ's Conjecture. *If L/k is a finite cyclic extension of fields, then $\mathbb{T}_{L/k}$ is rational over k; i.e., if $n = [L : k]$, there is a birational isomorphism over k*

$$\mathbb{T}_{L/k} \dashrightarrow \mathbb{A}^{\varphi(n)}.$$

By work of Klyachko and Voskresenskiĭ, this conjecture is known to hold when n is a product of at most two prime powers [17] (see also section 6.3 of [36]). In sections 3.2 and 2 above we gave explicit birational isomorphisms in some cases where $n = 6$ and 2. A $\mathbb{T}_n$-cryptosystem arises for every n for which Voskresenskiĭ's Conjecture is true over a finite field with efficiently computable birational maps.

When n is divisible by more than two distinct primes, Voskresenskiĭ's Conjecture is still an open question (despite a claim to the contrary in [37]). In particular, the conjecture is not known when $n = 30 = 2 \cdot 3 \cdot 5$.

7. Stable rationality.

In Definition 7.1 below we give the definition of stable rationality. One reason that Voskresenskiĭ's Conjecture would be difficult to disprove is that the tori $\mathbb{T}_{L/k}$ (for L/k cyclic) are known to always be stably rational over k (see the corollary on p. 61 of [36]), and it seems to be very difficult to prove the nonrationality of a stably rational torus. Although the stable rationality of $\mathbb{T}_{L/k}$ does not enable one to represent elements of $G_{q,n}$ in $\mathbb{F}_q^{\varphi(n)}$, it does allow one to represent elements of $G_{q,n} \times \mathbb{F}_q^r$ in $\mathbb{F}_q^{\varphi(n)+r}$ for a suitable r. In the language of the mathematical framework of this paper, the paper [8] of van Dijk and Woodruff can be viewed as a way to make clever use of the stable rationality of the algebraic tori $\mathbb{T}_n$ by encoding the message to be encrypted or signed in the extra affine piece $\mathbb{A}^r$.

DEFINITION 7.1. *A variety V over k is called* stably rational *over k if $V \times \mathbb{A}^r$ is rational over k for some $r \geq 0$ (i.e., $V \times \mathbb{A}^r$ is birationally isomorphic over k to $\mathbb{A}^s$ for some r and s).*

In [8], van Dijk and Woodruff used the polynomial identity

$$\Phi_n(x) = \prod_{d|n} (x^d - 1)^{\mu(n/d)}$$

to obtain an "almost bijection" between $G_{q,n} \times \mathbb{F}_q^r$ and $\mathbb{F}_q^s$, where

$$r = \sum_{d|n, \mu(n/d)=-1} d, \qquad s = \sum_{d|n, \mu(n/d)=1} d.$$

In particular, this gave an "almost bijection" between $G_{q,30} \times \mathbb{F}_q^{32}$ and $\mathbb{F}_q^{40}$, from which they obtained public key cryptosystems. In [7], the rationality of $\mathbb{T}_6$, the ideas of [8], and the polynomial identity

$$\Phi_n(x) \prod_{i=2}^{r-1} \Phi_{p_1 \cdots p_i}(x^{p_{i+2} \cdots p_r}) = \Phi_{p_1 p_2}(x^{p_3 \cdots p_r}),$$

where $n = p_1 \cdots p_r$ is a product of $r \geq 2$ distinct primes, are used to obtain an "almost bijection" between $G_{q,n} \times \mathbb{F}_q^{n/3 - \varphi(n)}$ and $\mathbb{F}_q^{n/3}$ if n is divisible by 6, giving a useful "almost bijection" between $G_{q,30} \times \mathbb{F}_q^2$ and $\mathbb{F}_q^{10}$. This improves the efficiency of the cryptosystems in [8].

It is an open question to find a birational isomorphism over $\mathbb{F}_q$ between $\mathbb{T}_{30} \times \mathbb{A}^1$ and $\mathbb{A}^9$ (or to prove its nonexistence).

8. Security considerations. The map $\alpha \mapsto (\alpha^{(q^n-1)/\Phi_t(q)})_{t|n}$ gives a homomorphism

$$\mathbb{F}_{q^n}^\times \cong (\mathrm{Res}_{\mathbb{F}_{q^n}/\mathbb{F}_q} \mathbb{G}_m)(\mathbb{F}_q) \to \bigoplus_{t|n} \mathbb{T}_t(\mathbb{F}_q) \cong \bigoplus_{t|n} G_{q,t} = G_{q,n} \oplus \bigoplus_{\substack{t|n \\ t \neq n}} G_{q,t}$$

whose kernel and cokernel have orders whose prime divisors all divide n. We have $G_{q,t} \subseteq \mathbb{F}_{q^t}^\times$ for all t, so for $t|n$ and $t < n$ the elements of the subgroups $G_{q,t}$ lie in a strictly smaller field than $\mathbb{F}_{q^n}$ and are therefore vulnerable to attacks on the discrete logarithm problem in $\mathbb{F}_{q^t}^\times$ for $t|n$ with $t < n$. By Lemma 1 of [4], if $h \in G_{q,n}$ is an element of prime order not dividing n, then $\mathbb{F}_q(h) = \mathbb{F}_{q^n}$; i.e., almost none of the elements of $G_{q,n}$ lie in a proper subfield of $\mathbb{F}_{q^n}$.

Part (ii) of the following result shows that the finite cyclic group $G_{q,n} = \mathbb{T}_n(\mathbb{F}_q)$ is as cryptographically secure as $\mathbb{F}_{q^n}^\times$ against the known subexponential attacks on the discrete logarithm problem.

PROPOSITION 8.1. *Suppose p is a prime, m and n are positive integers, $q = p^m$, and $(n, q) \neq (6, 2)$. Then*

(i) $\min\{k \in \mathbb{Z}^+ : \Phi_n(q)$ *divides* $p^k - 1\} = mn$*; and*

(ii) *the smallest extension F of $\mathbb{F}_p$ such that $G_{q,n} \subseteq F^\times$ is $\mathbb{F}_{q^n}$.*

Proof. Let k be the smallest positive integer such that $\Phi_n(q)$ divides $p^k - 1$. Since $\Phi_n(q)$ divides $q^n - 1$, we have $k \leq mn$. First suppose $mn > 2$. Since $(n, q) \neq (6, 2)$, it follows from a result of Zsigmondy (see Theorem 8.3, section IX of [14]) that $\Phi_{mn}(p)$ has a prime divisor ℓ that does not divide mn. By Lemma 4 of [27], mn is the order of p modulo ℓ. Since ℓ divides $\Phi_{mn}(p)$, which divides $\Phi_n(p^m)$, which divides $p^k - 1$, we have $mn \le k$. Thus $k = mn$, as desired. If $n = 1$, then clearly $k = m$. If $n = 2$ and $m = 1$, then clearly $k = 2$. This gives (i). Part (ii) follows from (i) since $|G_{q,n}| = \Phi_n(q)$ and $q^n = p^{mn}$. □

In a 2004 preprint, Kohel [18] suggests attacking cryptography on $G_{q,n}$ by using the fact that when n is odd and relatively prime to q, the tori $\mathbb{T}_n$ and $\mathbb{T}_{2n}$ are subschemes of the generalized Jacobian of a singular hyperelliptic curve $y^2 = cxf(x)^2$, where $f(x) \in \mathbb{F}_q[x]$ is irreducible of degree n. This seems like an interesting point of view that needs to be fleshed out and studied more fully.

Gaudry introduced a new probabilistic index calculus attack on the discrete logarithm problem for abelian varieties in his 2004 preprint [9]. Granger and Vercauteren [13] did an analogue of Gaudry's attack for the multiplicative group $\mathbb{G}_m$, which gives an attack on a subgroup of $\mathbb{F}_{q^6}^\times$ whose order is a 160-bit prime that is faster than Pollard ρ (which has complexity $O(\sqrt{q})$) when q is a sufficiently large fifth power (and therefore this attack applies also to subgroups of $\mathbb{F}_{q^{30}}^\times$), but has not been compared to index calculus attacks.

Joux et al. [15, 16] recently obtained efficient variants of the function field and number field sieve that bring the complexity of these attacks on the discrete log problem in $\mathbb{F}_{p^n}^\times$ to $L_{p^n}(1/3)$ for all finite fields $\mathbb{F}_{p^n}$, including the intermediate range where only $L_{p^n}(1/2)$ was previously known. They point out that the tori $\mathbb{T}_2$ and $\mathbb{T}_6$, which underlie LUC, XTR, and CEILIDH, appear to be safe from such attacks, as are cryptosystems based on the difficulty of the discrete log problem in $\mathbb{T}_{30}$ over $\mathbb{F}_p$ for 64-bit primes p, but not for 32-bit p.

To summarize, CEILIDH and XTR seem to be safe from known attacks, if one takes the parameter q to be a prime of at least 170 ($\approx \frac{1024}{6}$) bits. For $\mathbb{T}_{30}$-cryptosystems, Joux recommends taking 64-bit primes q to avoid all known attacks.

9. Interpreting discrete log cryptosystems in terms of quotients of tori. We will show that the XTR, Gong–Harn, and Lucas-based cryptosystems are based on the rationality of certain quotients of algebraic tori by the action of certain (finite) symmetric groups. In particular, Theorems 9.7 and 9.8, and the definition of the maps $\tilde{\sigma}_{i,F}$ in (5.2), show that the Lucas-based, Gong–Harn, and XTR cryptosystems are "based on" the quotient varieties $\mathbb{T}_2/S_2$, $\mathbb{T}_3/S_3$, and $\mathbb{T}_6/S_3$, respectively, and the conjectural "Looking beyond XTR" systems in [4] would be based on the quotient varieties $\mathbb{T}_{30}/(S_3 \times S_5)$ or $\mathbb{T}_{30}/(S_2 \times S_3 \times S_5)$, where S_r denotes the symmetric group on r letters, and the actions of these symmetric groups on $\mathbb{T}_n$ are defined in section 9.1. Theorem 9.11 shows that $\mathbb{T}_2/S_2$, $\mathbb{T}_3/S_3$, and $\mathbb{T}_6/S_3$ are rational varieties (and that is why the cryptosystems have efficient compression).

More precisely, for XTR, information exchanged corresponds to a $\mathrm{Gal}(\mathbb{F}_{q^6}/\mathbb{F}_{q^2})$-conjugacy class of $G_{q,6}$, which by Theorems 9.7 and 9.8 corresponds to an element of $\mathbb{T}_6/S_3$. The cryptosystem XTR takes advantage of the fact that $\mathbb{T}_6/S_3$ is rational, and the trace map from $\mathbb{F}_{q^6}$ to $\mathbb{F}_{q^2}$ induces a morphism and birational isomorphism $\mathbb{T}_6/S_3 \to \mathbb{A}^2 (= \mathrm{Res}_{\mathbb{F}_{q^2}/\mathbb{F}_q}\mathbb{A}^1)$ over $\mathbb{F}_q$ as in Theorem 9.11 and therefore gives a compact representation of $\mathbb{T}_6/S_3$ (i.e., an element of $(\mathbb{T}_6/S_3)(\mathbb{F}_q)$ is represented by two elements of $\mathbb{F}_q$). The set of equivalence classes $\mathbb{T}_6/S_3$ is not a group, because multiplication in $\mathbb{T}_6$ does not send S_3-orbits to S_3-orbits. This explains why XTR does not have a straightforward way to multiply. However, exponentiation in $\mathbb{T}_6$ does send S_3-orbits to S_3-orbits, and it induces a well-defined exponentiation in $\mathbb{T}_6/S_3$, and therefore in the set $\Lambda(\mathbb{F}_q, \mathbb{F}_{q^2}, \mathbb{F}_{q^6})$ of XTR traces (defined below).

Similarly for Lucas-based cryptosystems, the elements being exchanged correspond to elements of $\mathbb{T}_2/S_2$, and the trace map from $\mathbb{F}_{p^2}$ to $\mathbb{F}_p$ induces a morphism and birational isomorphism $\mathbb{T}_2/S_2 \to \mathbb{A}^1$ over $\mathbb{F}_p$.

From now on, L/k is a finite cyclic extension, $n := [L : k]$ is squarefree, and

$$k \subseteq F \subseteq L, \quad G := \mathrm{Gal}(L/k), \quad H := \mathrm{Gal}(L/F), \quad e := |H|, \quad d := n/e.$$

We define an algebraic variety $\mathcal{X}_F$ that underlies XTR, Gong–Harn, and the Lucas-based cryptosystems (with $k = \mathbb{F}_q$ and $(F, L) = (\mathbb{F}_{q^2}, \mathbb{F}_{q^6})$, $(\mathbb{F}_q, \mathbb{F}_{q^3})$, and $(F_q, \mathbb{F}_{q^2})$, respectively). Theorem 9.11 below shows that in those cases, $\mathcal{X}_F$ is rational. Theorem 9.11 can be viewed as a rephrasing of a result in [5]. Phrasing Theorem 9.11 in terms of quotients of algebraic tori and birational isomorphisms makes precise the underlying mathematics. This helped us find counterexamples in more general cases (see section 10), and may be useful in the future to indicate what ideas might be necessary to obtain correct and useful generalizations.

When $(k, F, L) = (\mathbb{F}_q, \mathbb{F}_{q^n}, \mathbb{F}_{q^n})$, then $(n, d, e) = (n, n, 1)$ and the varieties $\mathcal{X}_F$ and $\mathbb{T}_n/S_e$ are $\mathbb{T}_n$ itself, corresponding to the $\mathbb{T}_n$-cryptosystems ($\mathbb{T}_2$ is the case $(n, d, e) = (2, 2, 1)$ and CEILIDH is the case $(6, 6, 1)$). An effective proof of Voskresenskiĭ's Conjecture would provide a birational isomorphism between $\mathbb{T}_n$ and $\mathbb{A}^{\varphi(n)}$.

Because the details become more technical from this point on, we recommend that the casual reader ignore the proofs, lemmas, and propositions and concentrate on the definitions, theorem statements, and examples.

9.1. Group actions on tori. We next define actions of symmetric groups on the tori $\mathbb{T}_{L/k}$. If Γ is a finite set, let Σ_Γ denote the group of permutations of Γ. As an abstract group, Σ_G (resp., Σ_H) is the symmetric group S_n (resp., S_e). Since n is squarefree, there is a unique subgroup $J \subseteq G$ such that $G = H \times J$. This decomposition induces inclusions $\Sigma_H \subseteq \Sigma_G \subseteq \mathrm{Aut}_k(\mathbb{A}^G)$ and $\Sigma_H \subseteq \Sigma_G \subseteq \mathrm{Aut}_k(\mathbb{G}_m^G)$. More concretely, the action of $\pi \in \Sigma_H = S_e$ on $\mathbb{A}^G = \mathbb{A}^n$ is $(x_i)_{i \in \mathbb{Z}/n\mathbb{Z}} \mapsto (x_{\pi^{-1}(i)})_{i \in \mathbb{Z}/n\mathbb{Z}}$, where S_e acts on $G = \mathbb{Z}/n\mathbb{Z}$ via the decomposition $\mathbb{Z}/n\mathbb{Z} \cong \mathbb{Z}/e\mathbb{Z} \times \mathbb{Z}/d\mathbb{Z}$, with trivial action on the second factor. See also Examples 9.3 and 9.4 below. We have

$$\mathbb{A}^n = \mathbb{A}^G \underset{L}{\cong} \mathrm{Res}_{L/k}\mathbb{A}^1 \supset \mathrm{Res}_{L/k}\mathbb{G}_m \supset \mathbb{T}_{L/k}.$$

The action of Σ_H on $\mathrm{Res}_{L/k}\mathbb{A}^1 \cong \mathbb{A}^G$ sends $\mathrm{Res}_{L/k}\mathbb{G}_m$ to $\mathrm{Res}_{L/k}\mathbb{G}_m$. The images of Σ_H in $\mathrm{Aut}_L(\mathrm{Res}_{L/k}\mathbb{A}^1) \cong \mathrm{Aut}_L(\mathbb{A}^G)$ and in $\mathrm{Aut}_L(\mathrm{Res}_{L/k}\mathbb{G}_m) \cong \mathrm{Aut}_L(\mathbb{G}_m^G)$ are stable under the action of $\mathrm{Gal}(L/k)$ (by Corollary 1.7(i) of [24] with $\mathcal{I} = \mathcal{J} = \mathbb{Z}[G]$ and $V = \mathbb{G}_a = \mathbb{A}^1$ and $V = \mathbb{G}_m$ and Proposition 4.1 of [24] with $\mathcal{O} = \mathbb{Z}$ and $V = \mathbb{G}_a$ and $\mathbb{G}_m$), and it follows that the quotient varieties $\mathbb{A}^G/\Sigma_H$, $(\mathrm{Res}_{L/k}\mathbb{A}^1)/\Sigma_H$, and $(\mathrm{Res}_{L/k}\mathbb{G}_m)/\Sigma_H$ are all defined over k.

Recall the maps $\tilde{\sigma}_{i,F}$ from (5.2). We will make repeated use of the following lemma.

Lemma 9.1 (Proposition 3.2 of [29]). *The maps $\tilde{\sigma}_{i,F}$ for $1 \le i \le e$ factor through $(\mathrm{Res}_{L/k}\mathbb{A}^1)/\Sigma_H$ and induce a commutative diagram*

$$\begin{tikzcd}
\mathrm{Res}_{L/k}\mathbb{G}_m \arrow[r, two heads] \arrow[drr, "\oplus_{i=1}^{e}\tilde{\sigma}_{i,F}"'] & (\mathrm{Res}_{L/k}\mathbb{G}_m)/\Sigma_H \arrow[r, hook] & (\mathrm{Res}_{L/k}\mathbb{A}^1)/\Sigma_H \arrow[d, "\oplus_{i=1}^{e}\tilde{\sigma}_{i,F}"] \\
 & & (\mathrm{Res}_{F/k}\mathbb{A}^1)^e
\end{tikzcd}$$

where the right-hand vertical map is an isomorphism over k.

If e is divisible by two or more primes, then the action of Σ_H on $\mathrm{Res}_{L/k}\mathbb{G}_m$ does not send $\mathbb{T}_{L/k}$ to itself. We illustrate this concretely in Examples 9.3 and 9.4 below. The following result, which is used in Theorem 9.7 below, tells us which elements of Σ_G do send $\mathbb{T}_{L/k}$ to itself. In particular, Lemma 9.2 shows that if p is a prime divisor of n, then the action of S_p on $\mathbb{A}^n$ $(= \mathbb{A}^G)$ does take $\mathbb{T}_n$ to itself.

Write $G = \prod G_i$, with the G_i cyclic groups of (distinct) prime order.

Lemma 9.2. *If $\sigma \in \Sigma_G$, then $\sigma(\mathbb{T}_{L/k}) \subseteq \mathbb{T}_{L/k}$ if and only if $\sigma \in \prod_i \Sigma_{G_i}$.*

Proof. This follows from Theorem 7.3 of [24]; see also Lemma 3.5 of [29]. □

The following examples give concrete realizations of the tori $\mathbb{T}_n$ that allow explicit computation and show how the symmetric groups act.

Example 9.3. Let $n = e = 6$ and $d = 1$, and let

$$\Gamma = \mathbb{Z}/6\mathbb{Z} \cong \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/3\mathbb{Z} \supset \Omega = (\mathbb{Z}/2\mathbb{Z})^\times \times (\mathbb{Z}/3\mathbb{Z})^\times \cong (\mathbb{Z}/6\mathbb{Z})^\times.$$

By Definition 5.3, $\mathbb{T}_\Gamma \subset \mathbb{G}_m^\Gamma \xrightarrow{\sim} \mathbb{G}_m^6$ can be identified with the 2×3 matrices over $\mathbb{G}_m$ for which each row and column product is 1. By Lemma 5.4 we have $\mathbb{T}_6 \cong \mathbb{T}_\Gamma$ over $\mathbb{F}_{q^6}$, and by Lemma 5.6 we have $\mathbb{G}_m^2 \cong \mathbb{G}_m^\Omega \xrightarrow{\sim} \mathbb{T}_\Gamma \subset \mathbb{G}_m^\Gamma \xrightarrow{\sim} \mathbb{G}_m^6$ via

$$(x_1, x_2) \mapsto \begin{pmatrix} x_1 & x_2 & (x_1x_2)^{-1} \\ x_1^{-1} & x_2^{-1} & x_1x_2 \end{pmatrix}.$$

The action of S_2 interchanges the rows, and the action of S_3 permutes the columns of the 2×3 matrix. However, the action of S_6 on $\mathbb{G}_m^\Gamma = \mathbb{G}_m^6$ does not take $\mathbb{T}_\Gamma$ into itself (i.e., there are permutations of the 6 matrix entries that do not give a matrix of the same form). Thus, the action of S_6 does not take $\mathbb{T}_6$ into itself.

Example 9.4. More generally, if $n = pq$ and

$$\Gamma = \mathbb{Z}/n\mathbb{Z} \cong \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z} \supset \Omega = (\mathbb{Z}/p\mathbb{Z})^\times \times (\mathbb{Z}/q\mathbb{Z})^\times \cong (\mathbb{Z}/n\mathbb{Z})^\times,$$

then by Definition 5.3, $\mathbb{T}_\Gamma \subset \mathbb{G}_m^\Gamma \xrightarrow{\sim} \mathbb{G}_m^n$ can be identified with the $p \times q$ matrices over $\mathbb{G}_m$ for which each row and column product is 1. By Lemma 5.4 we have $\mathbb{T}_n \cong \mathbb{T}_\Gamma$ over $\mathbb{F}_{q^n}$, and by Lemma 5.6 we have $\mathbb{G}_m^{(p-1)(q-1)} \cong \mathbb{G}_m^\Omega \xrightarrow{\sim} \mathbb{T}_\Gamma \subset \mathbb{G}_m^\Gamma \xrightarrow{\sim} \mathbb{G}_m^n$ via $(x_{i,j})_{1 \le i \le p-1, 1 \le j \le q-1} \mapsto$

$$\begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,q-1} & (\prod_{\ell=1}^{q-1} x_{1,\ell})^{-1} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,q-1} & (\prod_{\ell=1}^{q-1} x_{2,\ell})^{-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{p-1,1} & x_{p-1,2} & \cdots & x_{p-1,q-1} & (\prod_{\ell=1}^{q-1} x_{p-1,\ell})^{-1} \\ (\prod_{k=1}^{p-1} x_{k,1})^{-1} & (\prod_{k=1}^{p-1} x_{k,2})^{-1} & \cdots & (\prod_{k=1}^{p-1} x_{k,q-1})^{-1} & \prod_{\ell=1}^{q-1} \prod_{k=1}^{p-1} x_{k,\ell} \end{pmatrix}.$$

Now S_p acts on $\mathbb{T}_\Gamma$ by permuting the rows of the matrix, and S_q acts by permuting the columns. However, the action of S_n on $\mathbb{G}_m^\Gamma = \mathbb{G}_m^n$ does not take $\mathbb{T}_\Gamma$ into itself and so does not take $\mathbb{T}_n$ into itself. More generally, taking $n = p_1p_2 \cdots p_r$, one can represent $\mathbb{T}_\Gamma$ via a $p_1 \times \cdots \times p_r$ multidimensional matrix. The proof of Lemma 5.6 can be viewed as a coordinate-free version of this representation.

Definition 9.5. *Let $\mathcal{X}_F$ denote the image of $\mathbb{T}_{L/k}$ in $(\mathrm{Res}_{L/k}\mathbb{G}_m)/\Sigma_H$. Let $\mathbb{X}_H$ be the image of $\mathbb{T}_G$ under the map $\mathbb{G}_m^G \twoheadrightarrow \mathbb{G}_m^G/\Sigma_H$, with Σ_H acting on $\mathbb{G}_m^G$ by permuting the factors as above.*

It follows from Lemma 5.6 that $\mathbb{T}_G$ and $\mathbb{T}_{L/k}$, and thus $\mathbb{X}_H$ and $\mathcal{X}_F$, are absolutely irreducible.

Write $H = \prod H_i$ with $\{H_i\} \subseteq \{G_i\}$, and define

$$\Sigma'_H := \prod_i \Sigma_{H_i} \subseteq \Sigma_H.$$

More concretely, letting $e = p_1 \cdots p_r$ be the prime factorization of the squarefree positive integer e, and letting $S'_e := S_{p_1} \times \cdots \times S_{p_r}$, then $\Sigma'_H = S'_e$. Note that when e is prime, then $S'_e = \Sigma'_H = \Sigma_H = S_e$. By Lemma 9.2, $\Sigma'_H \subseteq \mathrm{Aut}_{k_s}(\mathbb{T}_{L/k})$. Clearly the map $\mathbb{T}_{L/k} \to \mathcal{X}_F$ factors through $\mathbb{T}_{L/k}/\Sigma'_H$. When $k = \mathbb{F}_q$, we will denote $\mathbb{T}_{L/k}/\Sigma'_H$ by $\mathbb{T}_n/S'_e$.

The next lemma is used to prove Theorem 9.7.

Lemma 9.6. *Suppose Y is an affine variety defined over k, and X is an irreducible affine subvariety of Y defined over k. Suppose $\mathrm{Aut}_{k_s}(Y)$ contains a finite group Σ, and let $\Sigma_0 = \{\gamma \in \Sigma : \gamma(X) \subseteq X\}$. Then the natural map $X/\Sigma_0 \to Y/\Sigma$ induces a birational isomorphism over k from X/Σ_0 to its image in Y/Σ.*

Proof. If $g \in \Sigma$, let $U_g = X - g^{-1}(X)$. Let $U = \cap_{g\in\Sigma-\Sigma_0} U_g$. Then U is a nonempty Zariski-open subset of X. By the definition of U, the natural map $X/\Sigma_0 \to Y/\Sigma$ is injective on the image of U in X/Σ_0, proving the desired result. □

Theorem 9.7. *The natural map $\mathbb{T}_{L/k}/\Sigma'_H \to \mathcal{X}_F$ is a birational isomorphism over k.*

Proof. By Lemmas 9.6 and 9.2, the natural map $\mathbb{T}_{L/k}/\Sigma'_H \to (\mathrm{Res}_{L/k}\mathbb{G}_m)/\Sigma_H$ induces a birational isomorphism to $\mathcal{X}_F$. □

The next result will be used to prove Theorems 10.5 and 10.9.

Theorem 9.8. *Fix an isomorphism $(\phi_1, \ldots, \phi_d) : \mathrm{Res}_{F/k}\mathbb{A}^1 \xrightarrow{\sim} \mathbb{A}^d$ over k (for example, by fixing a k-basis of F). Then the function field $k(\mathcal{X}_F)$ is generated by the symmetric functions $\{\phi_j \circ \tilde{\sigma}_{i,F} : 1 \le i \le e, 1 \le j \le d\}$.*

Proof. By Lemma 9.1, the function field $k((\mathrm{Res}_{L/k}\mathbb{A}^1)/\Sigma_H)$ is generated by the maps $\phi_j \circ \tilde{\sigma}_{i,F}$. Since $\mathcal{X}_F$ is a subvariety of $(\mathrm{Res}_{L/k}\mathbb{A}^1)/\Sigma_H$, the restrictions of those maps to $\mathcal{X}_F$ generate $k(\mathcal{X}_F)$. □

Remark 9.9. Let $G_{L/k} \subseteq L^\times$ be the image of $\mathbb{T}_{L/k}(k)$ under the map of Theorem 5.7(ii), and let $\rho : \mathbb{T}_{L/k} \to \mathcal{X}_F$ be the natural map. Then Theorem 9.8 (combined with Lemma 5.1) shows that ρ induces a one-to-one correspondence between the $\mathrm{Gal}(L/F)$-orbits of $G_{L/k}$ and the subset $\rho(\mathbb{T}_{L/k}(k))$ of $\mathcal{X}_F(k)$. In particular, the $\mathrm{Gal}(\mathbb{F}_{q^n}/\mathbb{F}_{q^d})$-orbits of $G_{q,n}$ are in bijection with the image of $\mathbb{T}_n(\mathbb{F}_q)$ in $\mathcal{X}_{\mathbb{F}_{q^d}}(\mathbb{F}_q)$. When $n = 6$, $k = \mathbb{F}_q$, and $F = \mathbb{F}_{q^2}$, the map $\mathrm{Res}_{\mathbb{F}_{q^6}/\mathbb{F}_q}\mathbb{G}_m \to (\mathrm{Res}_{\mathbb{F}_{q^6}/\mathbb{F}_q}\mathbb{G}_m)/S_3$ induces $\rho : \mathbb{T}_6 \to \mathbb{T}_6/S_3 = \mathcal{X}_F$, a (generically) 6-to-1 map. However, for the induced map on $\mathbb{F}_q$-points $\rho : \mathbb{T}_6(\mathbb{F}_q) \to \mathcal{X}_F(\mathbb{F}_q)$, almost all nonempty fibers have size 3, corresponding to $\mathrm{Gal}(\mathbb{F}_{q^6}/\mathbb{F}_{q^2})$-orbits in $G_{q,6}$.

9.2. Interpreting XTR, Gong–Harn, and Lucas-based systems.

Theorem 9.11 below can be viewed as a rephrasing, in the language of this paper, of a result in section 5 of [5] (see also Proposition 1 of [4]) that says that the minimal polynomial over $\mathbb{F}_{q^d}$ of an element of $G_{p,n}$ can be represented using $\varphi(n)\log_2(p)$ bits if $d = 1$ or 2 and e is prime.

With notation k, L, F, G, H, n, e, and d as before, let $u = \lceil \varphi(n)/d \rceil$. There is a commutative diagram

$$\begin{tikzcd}
\mathbb{T}_{L/k} \arrow[d, two heads] & \subseteq & \mathrm{Res}_{L/k}\mathbb{G}_m \arrow[r, hook] & \mathrm{Res}_{L/k}\mathbb{A}^1 \arrow[r, "\sim"] \arrow[dll] \arrow[d, "\oplus_{i=1}^u \tilde{\sigma}_{i,F}"] & \mathbb{A}^G \arrow[d, "\oplus_{i=1}^u s_i"] \\
\mathbb{T}_{L/k}/\Sigma'_H \arrow[r] & (\mathrm{Res}_{L/k}\mathbb{A}^1)/\Sigma'_H \arrow[rr] & & (\mathrm{Res}_{F/k}\mathbb{A}^1)^u \arrow[r, "\sim"] & (\mathbb{A}^{G/H})^u
\end{tikzcd}$$

where the top and bottom isomorphisms are defined over L and F, respectively, and the functions s_i were defined in Definition 5.3. Let

$$\lambda_F := (\tilde{\sigma}_{1,F}, \ldots, \tilde{\sigma}_{u,F}) : \mathbb{T}_{L/k}/\Sigma'_H \to (\mathrm{Res}_{F/k}\mathbb{A}^1)^u \tag{9.1}$$

denote the composition in the bottom row, and let

$$\Lambda(k, F, L) := \{\lambda_F(\alpha) : \alpha \in \mathbb{T}_{L/k}(k)\} \subseteq (\mathrm{Res}_{F/k}\mathbb{A}^1)^u(k) \cong F^u.$$

Note that $\Lambda(\mathbb{F}_q, \mathbb{F}_{q^d}, \mathbb{F}_{q^n}) = \{(\sigma_1(\alpha), \ldots, \sigma_u(\alpha)) : \alpha \in G_{q,n}\} \subseteq (\mathbb{F}_{q^d})^{\lceil \varphi(n)/d \rceil}$, where $\sigma_i(\alpha)$ is the ith symmetric function on $\{\alpha^\gamma : \gamma \in \mathrm{Gal}(\mathbb{F}_{q^n}/\mathbb{F}_{q^d})\}$. The Lucas-based and XTR cryptosystems correspond to the cases $(n, d, e) = (2, 1, 2)$ and $(6, 2, 3)$, respectively. In these two cases, λ_F is essentially the trace map from $\mathbb{F}_{q^n}$ to $\mathbb{F}_{q^d}$, and $\Lambda(\mathbb{F}_q, \mathbb{F}_{q^d}, \mathbb{F}_{q^n})$ is the set of traces used in the Lucas-based systems and XTR, respectively. Further, when $(n, d, e) = (3, 1, 3)$, then $\Lambda(\mathbb{F}_q, \mathbb{F}_{q^d}, \mathbb{F}_{q^n})$ is the set of values that occur in the Gong–Harn cryptosystem. In Theorem 10.5 below we will show that a conjecture in [4] on how to generalize XTR would imply that λ_F is always a birational isomorphism.

The following result, which will be used to prove Theorem 10.9, gives equivalent conditions for λ_F to be a birational isomorphism.

PROPOSITION 9.10.

(i) *The isomorphism* $\mathbb{T}_{L/k} \xrightarrow{\sim} \mathbb{T}_G$ *of Lemma* 5.4 *induces an isomorphism*

$$\mathcal{X}_F \xrightarrow{\sim} \mathbb{X}_H$$

defined over F.

(ii) *Lemma* 9.1 *remains true when* $\mathrm{Res}_{L/k}\mathbb{G}_m$, $\mathrm{Res}_{L/k}\mathbb{A}^1$, *and* $\tilde{\sigma}_{i,F}$ *are replaced by* $\mathbb{G}_m^G$, $\mathbb{A}^G$, *and* s_i, *respectively, where the* s_i *were defined in Definition* 5.3.

(iii) *There is a commutative diagram, with maps defined over* F,

$$\begin{tikzcd}
\mathcal{X}_F \arrow[r, "\sim"] \arrow[d, "\oplus_{i=1}^e \tilde{\sigma}_{i,F}"'] & \mathbb{X}_H \arrow[d, "\oplus_{i=1}^e s_i"] \\
(\mathrm{Res}_{F/k}\mathbb{A}^1)^e \arrow[r, "\sim"] & (\mathbb{A}^{G/H})^e
\end{tikzcd}$$

where the top map is the isomorphism of (i)*, the bottom isomorphism is given by the eth power of* (4.3) *(with* $V = \mathbb{A}^1$*), and the left map is induced by the map of Lemma* 9.1*.*

(iv) *There is a commutative diagram*

$$\begin{tikzcd}
\mathbb{T}_{L/k}/\Sigma'_H \arrow[r] \arrow[dr, "\lambda_F"'] & \mathcal{X}_F \arrow[r, "\sim"] \arrow[d, "\oplus_{i=1}^u \tilde{\sigma}_{i,F}"] & \mathbb{X}_H \arrow[d, "\oplus_{i=1}^u s_i"] \\
& (\mathrm{Res}_{F/k}\mathbb{A}^1)^u \arrow[r, "\sim"] & (\mathbb{A}^{G/H})^u
\end{tikzcd}$$

where the top left map is the birational isomorphism of Theorem 9.7, *the top right map is from* (i), *and the bottom map is the uth power of* (4.3).

(v) *The following are equivalent:*

(a) λ_F *is a birational isomorphism,*

(b) $\oplus_{i=1}^{u} \tilde{\sigma}_{i,F}$ *is a birational isomorphism,*

(c) $\oplus_{i=1}^{u} s_i$ *is a birational isomorphism.*

Proof. Part (i) follows from Lemma 5.4, (4.3), and the definitions of $\mathcal{X}_F$ and $\mathbb{X}_H$. Part (ii) follows from (4.3). Part (iii) now follows immediately, while (iv) follows from Theorem 9.7 and the definition of λ_F. Part (v) follows from (iv) and the fact that being a birational isomorphism is invariant under change of base field. □

Theorem 9.11. *Suppose e is prime and* $d = 1$ *or* 2. *Then* λ_F *is a birational isomorphism and injective morphism*

$$\mathbb{T}_{L/k}/\Sigma'_H \hookrightarrow (\mathrm{Res}_{F/k}\mathbb{A}^1)^{\varphi(n)/d} \quad (\cong \mathbb{A}^{\varphi(n)})$$

such that $\Lambda(k, F, L)$ *is the image of the composition*

$$\mathbb{T}_{L/k}(k) \longrightarrow (\mathbb{T}_{L/k}/\Sigma'_H)(k) \hookrightarrow (\mathrm{Res}_{F/k}\mathbb{A}^1)^{\varphi(n)/d}(k) \cong F^{\varphi(n)/d}.$$

In this way, $\Lambda(k, F, L)$ *can be identified with the image of* $\mathbb{T}_{L/k}(k)$ *in* $(\mathbb{T}_{L/k}/\Sigma'_H)(k)$.

Proof. By definition, $\Lambda(k, F, L)$ is the image of the composition

$$\mathbb{T}_{L/k}(k) \to (\mathbb{T}_{L/k}/\Sigma'_H)(k) \to (\mathrm{Res}_{F/k}\mathbb{A}^1)^u(k) \cong F^u.$$

When d divides $\varphi(n)$, then $\mathbb{T}_{L/k}$ and $(\mathrm{Res}_{F/k}\mathbb{A}^1)^u$ are both $\varphi(n)$-dimensional varieties over k. Thus to prove the theorem we need only show that when $d = 1$ or 2 and e is prime then λ_F is injective. By Lemma 9.1,

$$(\tilde{\sigma}_{1,F}, \ldots, \tilde{\sigma}_{e,F}) : (\mathrm{Res}_{F/k}\mathbb{A}^1)/\Sigma_H \xrightarrow{\sim} (\mathrm{Res}_{F/k}\mathbb{A}^1)^e. \tag{9.2}$$

Suppose e is prime. Then $\Sigma'_H = \Sigma_H$, and $\mathbb{T}_{L/k}/\Sigma'_H$ is a subvariety of $\mathrm{Res}_{F/k}\mathbb{A}^1/\Sigma_H$.

Suppose first that $d = 1$. By the definitions of $\mathbb{T}_{L/k}$ and $\tilde{\sigma}_{e,F}$, we have $\tilde{\sigma}_{e,F} = \mathrm{N}_{L/F,k} = 1$ on $\mathbb{T}_{L/k}$. Thus $(\tilde{\sigma}_{1,F}, \ldots, \tilde{\sigma}_{e,F}) = (\lambda_F, 1)$ on $\mathbb{T}_{L/k}$. The injectivity of λ_F follows from the injectivity of (9.2).

Now suppose that $d = 2$ (so e is an odd prime). Let M denote the degree e extension of k in L and let ρ denote the element of order 2 in G. We have $\mathrm{N}_{L/M,k}(g) = g \cdot g^{\rho}$ and $\mathrm{N}_{L/M,k} = 1$ on $\mathbb{T}_{L/k}$. Thus ρ is the same as inversion on $\mathbb{T}_{L/k}$. By definition,

$$\tilde{\sigma}_{i,F}(g_1, \ldots, g_e) = \sum_{\substack{S \subseteq \{1,\ldots,e\} \\ |S| = i}} \prod_{j \in S} g_j, \qquad \frac{\tilde{\sigma}_{e-i,F}}{\tilde{\sigma}_{e,F}}(g_1, \ldots, g_e) = \sum_{\substack{S \subseteq \{1,\ldots,e\} \\ |S| = i}} \prod_{j \in S} g_j^{-1}.$$

Since ρ is inversion on $\mathbb{T}_{L/k}$ and $\tilde{\sigma}_{e,F} = 1$ on $\mathbb{T}_{L/k}$, we have $\tilde{\sigma}_{i,F}^{\rho} = \tilde{\sigma}_{e-i,F}/\tilde{\sigma}_{e,F} = \tilde{\sigma}_{e-i,F}$ on $\mathbb{T}_{L/k}$. Thus

$$(\tilde{\sigma}_{1,F}, \ldots, \tilde{\sigma}_{e,F}) = (\tilde{\sigma}_{1,F}, \ldots, \tilde{\sigma}_{(e-1)/2,F}, \tilde{\sigma}^{\rho}_{(e-1)/2,F}, \ldots, \tilde{\sigma}^{\rho}_{1,F}, 1)$$

on $\mathbb{T}_{L/k}$. Since $\lambda_F = (\tilde{\sigma}_{1,F}, \ldots, \tilde{\sigma}_{(e-1)/2,F})$, the injectivity of λ_F again follows from (9.2). □

10. Looking beyond XTR. Lenstra [20] asked if one can use $n = 30$ to do better than XTR. The Bosma–Hutton–Verheul paper "Looking beyond XTR" [4], building on a conjecture in [5], asked whether, for $n > 6$, some set of elementary symmetric polynomials can be used in place of the trace. In particular, [4] asked whether one can recover the values of all the elementary symmetric polynomials (i.e., the entire characteristic polynomial) for $\mathrm{Gal}(\mathbb{F}_{p^n}/\mathbb{F}_{p^d})$ from the first $\lceil\varphi(n)/d\rceil$ of them (this was already answered in the affirmative in [5] when $(d, n/d) = (1, \ell)$ or $(2, \ell)$ with ℓ prime). If this were true, one could use the first $\lceil\varphi(n)/d\rceil$ elementary symmetric polynomials on the set of $\mathrm{Gal}(\mathbb{F}_{p^n}/\mathbb{F}_{p^d})$-conjugates of an element $h \in G_{q,n}$ to compress h, representing it via $\varphi(n)$ elements of $\mathbb{F}_q$.

Of the four conjectures stated in [4], the two "strong" conjectures were disproved there. In Theorem 10.1 and Corollary 10.2 below we disprove the two remaining conjectures (Conjectures 1 and 3 of [4], which were also called (d, e)-**BPV** and n-**BPV** in [4]). In fact, we can do better. We have constructed examples that show not only that the conjectures are false but also that weakening the conjectures does not help. In particular, when $n = 30$ and $p = 7$, we can show that

- for $d = 1$, no 8 ($= \varphi(n)/d$) elementary symmetric polynomials determine *any* of the remaining ones, except for those determined by the symmetry of the characteristic polynomial;
- for $d = 1$, no 10 elementary symmetric polynomials determine *all* of them; and
- for $d = 2$, no 4 ($= \varphi(n)/d$) elementary symmetric polynomials determine all of them.

Rationality of the varieties $\mathbb{T}_n/S'_n$ (or, more generally, the varieties $\mathbb{T}_n/S'_e$) would imply the conjecture in [5] that characteristic polynomials (i.e., Galois-conjugacy classes) of elements of $G_{p,n}$ can be represented using $\varphi(n)\log_2(p)$ bits. We see in Theorem 10.5 below that the conjectures in [4] would imply the stronger statement (when d divides $\varphi(n)$) that the map $\lambda_{\mathbb{F}_{q^d}}$ of (9.1) is a (morphism and) birational isomorphism

$$\mathbb{T}_n/S'_e \to (\mathrm{Res}_{\mathbb{F}_{q^d}/\mathbb{F}_q}\mathbb{A}^1)^{\varphi(n)/d} \cong \mathbb{A}^{\varphi(n)}.$$

Theorem 9.11 above showed this is true when e is a prime and $d = 1$ or 2. In particular, it is true when (d, e) is $(1, 1)$ (Diffie–Hellman), $(1, 2)$ (Lucas-based), $(1, 3)$ (Gong–Harn), and $(2, 3)$ (XTR). Theorem 10.9 below shows that this is false for $(d, e) = (1, 30)$ and $(2, 15)$ in all but at most finitely many characteristics p; i.e., the first eight elementary symmetric polynomials do not induce a birational isomorphism $\mathbb{T}_{30}/S'_{30} = \mathbb{T}_{30}/(S_2 \times S_3 \times S_5) \to \mathbb{A}^8$ over $\mathbb{F}_p$, and the first four elementary symmetric polynomials on the $\mathrm{Gal}(\mathbb{F}_{p^{30}}/\mathbb{F}_{p^2})$-conjugates of an element in $\mathbb{T}_{30}$ do not induce a birational isomorphism $\mathbb{T}_{30}/S'_{15} = \mathbb{T}_{30}/(S_3 \times S_5) \to (\mathrm{Res}_{\mathbb{F}_{p^2}/\mathbb{F}_p}\mathbb{A}^1)^4 \cong \mathbb{A}^8$ over $\mathbb{F}_p$. In summary, elementary symmetric polynomials are not the correct functions to use.

Fix an integer $n > 1$, a prime p, and a factorization $n = de$ with $e > 1$. For $h \in G_{p,n}$, let $P_h^{(d)}$ be the characteristic polynomial of h over $\mathbb{F}_{p^d}$, and define functions $a_j : G_{p,n} \to \mathbb{F}_{p^d}$ by

$$P_h^{(d)}(X) = X^e + a_{e-1}(h)X^{e-1} + \cdots + a_1(h)X + a_0(h).$$

Then $a_0(h) = (-1)^e$. If n is even, then

$$a_j(h) = (-1)^e(a_{e-j}(h))^{p^{n/2}} \tag{10.1}$$

for all $j \in \{1, \ldots, e-1\}$ (see, for example, Theorem 1 of [4] or the proof of Theorem 9.11 above). Let

$$S_{p,n} = \{h \in G_{p,n} : \mathbb{F}_p(h) = \mathbb{F}_{p^n}\}.$$

Next we state Conjectures 1 and 3 (also called (d,e)-**BPV** and n-**BPV**, resp.) of [4].

Conjecture (d,e)-**BPV**. *Let $n = de$ with $e > 1$. Then $\lceil \varphi(n)/d \rceil$ is the smallest positive integer u for which there are polynomials*

$$Q_j \in \mathbb{Z}[X_1^{(0)}, \ldots, X_1^{(d-1)}, X_2^{(0)}, \ldots, X_2^{(d-1)}, \ldots, X_u^{(0)}, \ldots, X_u^{(d-1)}]$$

for all $1 \le j \le e-u-1$, such that for every prime p and every $h \in S_{p,n}$,

$$a_j(h) = \bar{Q}_j(a_{e-1}, a_{e-1}^p, \ldots, a_{e-1}^{p^{d-1}}, a_{e-2}, a_{e-2}^p, \ldots, a_{e-2}^{p^{d-1}}, \ldots, a_{e-u}, a_{e-u}^p, \ldots, a_{e-u}^{p^{d-1}}),$$

where $\bar{Q}_j$ denotes Q_j with coefficients taken modulo p.

Conjecture n-**BPV**. *Suppose $1 < n \in \mathbb{Z}$. Then n has a divisor d such that d divides $\varphi(n)$ and Conjecture $(d, n/d)$-**BPV** holds.*

Theorem 10.1. *Conjecture (d,e)-**BPV** is false when $(d,e) = (1,30)$ and $(2,15)$.*

Proof. Let $u = \lceil \varphi(n)/d \rceil$. Conjecture (d,e)-**BPV** would imply there are polynomials $Q_1, \ldots, Q_{e-u-1} \in \mathbb{Z}[x_1, \ldots, x_u]$ such that $a_j(h) = Q_j(a_{e-u}(h), \ldots, a_{e-1}(h))$ for all primes p, $h \in S_{p,n}$, and $j \in \{1, \ldots, e-u-1\}$; so for each p and h the values $a_{e-u}(h), \ldots, a_{e-1}(h)$ would determine $a_j(h)$ for *every* j. We will disprove Conjecture (d,e)-**BPV** by exhibiting two elements $h, h' \in S_{p,n}$ such that $a_j(h) = a_j(h')$ whenever $e-u \le j \le e-1$ but $a_j(h) \ne a_j(h')$ for at least one $j < e-u$, with $p = 7$ and 11.

Let $n = 30$, and $p = 7$ or 11. Note that $\Phi_{30}(7) = 6568801$ (a prime) and $\Phi_{30}(11) = 31 \times 7537711$. Since $\Phi_{30}(p)$ is relatively prime to 30, by Lemma 1 of [4] we have $S_{p,30} = G_{p,30} - \{1\}$. View the field $\mathbb{F}_{p^{30}}$ as $\mathbb{F}_p[x]/f(x)$ with an irreducible polynomial $f(x) \in \mathbb{F}_p[x]$, and fix a generator g of $G_{p,n}$. Specifically, let $r = (p^{30}-1)/\Phi_{30}(p)$ and let

$$f(x) = x^{30} + x^2 + x + 5, \qquad g = x^r \qquad \text{if } p = 7,$$

$$f(x) = x^{30} + 2x^2 + 1, \qquad g = (x+1)^r \qquad \text{if } p = 11.$$

Case 1. $d = 1$, $e = 30$. Then $u = \lceil \varphi(n)/d \rceil = \varphi(30) = 8$. For $h \in S_{p,30} = G_{p,30} - \{1\}$ and $1 \le j \le 29$ we have $a_j(h) = a_{30-j}(h)$ by (10.1), so we need only consider $a_j(h)$ for $15 \le j \le 29$. By constructing a table of g^i and their characteristic polynomials $P_{g^i}^{(d)}$ for $i = 1, 2, \ldots$, and checking for matching coefficients, we found the examples in Tables 1 and 2 below. The examples in Table 1 (resp., Table 2) disprove Conjecture $(1,30)$-**BPV** with $p = 7$ (resp., 11).

Case 2. $d = 2$, $e = 15$. Then $u = \lceil \varphi(n)/d \rceil = \varphi(30)/2 = 4$. For $h \in S_{p,30} = G_{p,30} - \{1\}$ and $1 \le j \le 14$ we have $a_j(h) = \bar{a}_{15-j}(h)$ by (10.1), where $\bar{a}$ denotes conjugation in $\mathbb{F}_{p^2}$. Thus we need only consider $a_j(h)$ for $8 \le j \le 14$. View $\mathbb{F}_{p^2}$ as $\mathbb{F}_p(i)$, where $i^2 = -1$. A computer search as above leads to the examples in Tables 3 and 4. The examples in Table 3 (resp., Table 4) disprove Conjecture $(2,15)$-**BPV** with $p = 7$ (resp., 11). □

If $n > 1$ is fixed, then Conjecture n-**BPV** of [4] says that there exists a divisor d of both n and $\varphi(n)$ such that $(d, n/d)$-**BPV** holds. Since $\gcd(30, \varphi(30)) = 2$, when

Table 1
Values of $a_j(h) \in \mathbb{F}_7$ *for several* $h \in G_{7,30}$.

$h \setminus j$	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
g^{2754}	3	2	0	6	4	4	2	**5**	**4**	**0**	**2**	**2**	**1**	**4**	**4**
g^{6182}	5	4	4	5	5	3	1	**5**	**4**	**0**	**2**	**2**	**1**	**4**	**4**
g^{5374}	2	0	5	**2**	1	**6**	**4**	**6**	**1**	**1**	**5**	**6**	**4**	**2**	**6**
g^{23251}	4	2	0	**2**	3	**6**	**4**	**6**	**1**	**1**	**5**	**6**	**4**	**2**	**6**

Table 2
Values of $a_j(h) \in \mathbb{F}_{11}$ *for several* $h \in G_{11,30}$.

$h \setminus j$	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
g^{7525}	**10**	**2**	9	7	7	5	6	**9**	**2**	**1**	**8**	**10**	**4**	**1**	**10**
g^{31624}	**10**	**2**	2	4	2	3	10	**9**	**2**	**1**	**8**	**10**	**4**	**1**	**10**
g^{46208}	9	9	6	10	6	10	**10**	**8**	**1**	**3**	**2**	**7**	**4**	**6**	**5**
g^{46907}	7	8	0	0	1	7	**10**	**8**	**1**	**3**	**2**	**7**	**4**	**6**	**5**

Table 3
Values of $a_j(h) \in \mathbb{F}_{49}$ *for certain* $h \in G_{7,30}$.

$h \setminus j$	8	9	10	11	12	13	14
g^{173}	$4+4i$	$5+i$	$1+6i$	$\boldsymbol{4i}$	$\boldsymbol{2+3i}$	$\boldsymbol{6+3i}$	$\boldsymbol{3+i}$
g^{2669}	6	$6+3i$	$5+i$	$\boldsymbol{4i}$	$\boldsymbol{2+3i}$	$\boldsymbol{6+3i}$	$\boldsymbol{3+i}$
g^{764}	$6+6i$	**5**	**5**	**0**	**0**	**6**	**2**
g^{5348}	$6+i$	**5**	**5**	**0**	**0**	**6**	**2**

Table 4
Values of $a_j(h) \in \mathbb{F}_{121}$ *for certain* $h \in G_{11,30}$.

$h \setminus j$	8	9	10	11	12	13	14
g^{9034}	$10+i$	$10i$	$3+3i$	$\boldsymbol{1+4i}$	$\boldsymbol{8+9i}$	$\boldsymbol{5+4i}$	**9**
g^{18196}	$6+8i$	$9+10i$	$8+i$	$\boldsymbol{1+4i}$	$\boldsymbol{8+9i}$	$\boldsymbol{5+4i}$	**9**

$n = 30$ we need only consider $d = 1$ and 2. The following is an immediate consequence of Theorem 10.1.

Corollary 10.2. *Conjectures* (1, 30)-**BPV**, (2, 15)-**BPV**, *and* 30-**BPV** *of* [4] *are false. Thus, Conjectures* 1 *and* 3 *of* [4] *are both false.*

Remark 10.3. For $d = 1$ and $e = 30$, the last two lines of Table 1 (resp., Table 2) show that even the larger collection of values $a_{18}(h)$, $a_{20}(h)$, $\ldots$, $a_{29}(h)$ (resp., $a_{21}(h), \ldots, a_{29}(h)$) does not determine any of the other values when $p = 7$ (resp., $p = 11$). We also found that no eight coefficients determine all the rest; we found 64 pairs of elements so that given any set of eight coefficients, one of these 64 pairs agrees on these coefficients but not everywhere. In fact, we computed additional examples that show that when $p = 7$, no ten coefficients determine all the rest. We also show that when $p = 7$ no set of eight coefficients determines even one additional coefficient.

Suppose now $d = 2$, $e = 15$, and $p = 7$. Then the last two lines of Table 3 show that even the larger collection of values $a_9(h)$, $\ldots$, $a_{14}(h)$ does not determine the remaining value $a_8(h) \in \mathbb{F}_{49}$. We have computed additional examples that show that *no* choice of four of the values $a_8(h), \dots, a_{14}(h)$ determines the other three.

The next lemma is used to prove Theorem 10.5 (and Lemma 10.6) below.

Lemma 10.4. *Suppose L/k is a cyclic extension of degree n, and τ is a generator of $G := \mathrm{Gal}(L/k)$. Then the natural ring homomorphism $\gamma : \mathbb{Z}[G] \to \mathrm{End}(\mathbb{T}_{L/k})$ has kernel $(\Phi_n(\tau))$.*

Proof. This follows from Proposition 4.2(iii) and Lemma 5.4 of [24]. □

Theorem 10.5. *Suppose k is a prime field ($\mathbb{Q}$ or $\mathbb{F}_p$), n is a squarefree integer, L/k is a cyclic extension of degree n, $k \subseteq F \subseteq L$, $d := [F : k]$, and $e := [L : F]$. Suppose d divides $\varphi(n)$. Then Conjecture (d, e)-**BPV** of* [4] *implies that the map λ_F defined in* (9.1) *is a birational isomorphism.*

Proof. Let $u = \varphi(n)/d$. Since $\dim(\mathcal{X}_F) = \dim((\mathrm{Res}_{F/k}\mathbb{A}^1)^u)$, it suffices to show that λ_F induces a surjective map on function fields $k((\mathrm{Res}_{F/k}\mathbb{A}^1)^u) \to k(\mathcal{X}_F)$. Fix an isomorphism $(\phi_1, \dots, \phi_d) : \mathrm{Res}_{F/k}\mathbb{A}^1 \xrightarrow{\sim} \mathbb{A}^d$ over k. Since $\{\phi_j \circ \tilde{\sigma}_{i,F} : 1 \le i \le e, 1 \le j \le d\}$ generates $k(\mathcal{X}_F)$ by Theorem 9.8, it suffices to show that for all $1 \le i \le e$ and $1 \le j \le d$ there is a $g_{i,j} \in k((\mathrm{Res}_{F/k}\mathbb{A}^1)^u)$ such that $g_{i,j} \circ \lambda_F = \phi_j \circ \tilde{\sigma}_{i,F}$.

For $1 \le j \le d$, let $t_j : (\mathrm{Res}_{F/k}\mathbb{A}^1)^u \to \mathrm{Res}_{F/k}\mathbb{A}^1$ be the jth projection. Then $t_i \circ \lambda_F = \tilde{\sigma}_{i,F}$. With Q_i from Conjecture (d, e)-**BPV** and writing $[\tau^i]$ for $\gamma(\tau^i)$ with τ and γ as in Lemma 10.4, for $1 \le i \le e$ define $f_i : \mathbb{T}_{L/k} \to \mathrm{Res}_{F/k}\mathbb{A}^1$ by

$$\begin{aligned} f_i = \tilde{\sigma}_{i,F} - Q_i(\tilde{\sigma}_{e-1,F}, [\tau] \circ \tilde{\sigma}_{e-1,F}, [\tau^2] \circ \tilde{\sigma}_{e-1,F}, \dots, [\tau^{d-1}] \circ \tilde{\sigma}_{e-1,F}, \\ \tilde{\sigma}_{e-2,F}, \dots, [\tau^{d-1}] \circ \tilde{\sigma}_{e-u,F}). \end{aligned}$$

We show below that $f_i = 0$. The desired result then follows by taking

$$g_{i,j} := \phi_j \circ Q_i(t_1, [\tau] \circ t_1, \dots, [\tau^{d-1}] \circ t_1, t_2, [\tau] \circ t_2, \dots, [\tau^{d-1}] \circ t_{e-u}).$$

First suppose $k = \mathbb{Q}$. Viewing $\mathbb{T}_{L/\mathbb{Q}}(\mathbb{Q}) \subseteq L^\times$ via Theorem 5.7(ii), let $A_L := \{\alpha \in \mathbb{T}_{L/\mathbb{Q}}(\mathbb{Q}) : L = \mathbb{Q}(\alpha)\}$. Fix any $\alpha \in A_L$. Let $S(\alpha)$ be the set of all primes ℓ such that $\mathrm{Frob}_\ell(L/\mathbb{Q}) = \tau$, α is integral at ℓ, and ℓ does not divide the discriminant of the minimal polynomial for α over $\mathbb{Q}$. Let $\mathcal{O}_L$ denote the ring of integers of the number field L. Since $\mathrm{Frob}_\ell(L/\mathbb{Q}) = \tau$, we have $\mathcal{O}_L/\ell\mathcal{O}_L \cong \mathbb{F}_{\ell^n}$. Since α is integral at ℓ, and ℓ does not divide the discriminant of α's minimal polynomial, we have $\mathbb{F}_{\ell^n} = \mathbb{F}_\ell(\tilde{\alpha})$, where $\tilde{\alpha}$ is the image of α under $(\mathcal{O}_L)_{(\ell)} \to \mathcal{O}_L/\ell\mathcal{O}_L$, with $(\mathcal{O}_L)_{(\ell)}$ the localization. Conjecture (d, e)-**BPV** implies $\mathrm{ord}_\ell(f_i(\alpha)) > 0$ for all $\ell \in S(\alpha)$. Since $S(\alpha)$ is an infinite set (by the Cebotarev density theorem), $f_i(\alpha) = 0$. Lemma 10.6(ii) below shows that A_L is Zariski-dense in $\mathbb{T}_{L/\mathbb{Q}}$; therefore, $f_i = 0$.

Now suppose $k = \mathbb{F}_p$. Let L' be any cyclic extension of $\mathbb{Q}$ of degree n for which p is inert, and let F' be the subfield of L' of degree d over $\mathbb{Q}$. Since p is inert, the residue field of F' at p is $\mathbb{F}_{p^d} = F$. The map f_i is the reduction modulo p of the f_i defined in characteristic zero and thus is zero. □

The previous proof made use of the following lemma.

Lemma 10.6. *Suppose k is an infinite field, and L is a cyclic extension of k of finite squarefree degree. Let $\iota : \mathbb{T}_{L/k}(k) \hookrightarrow L^\times$ be the inclusion of Theorem* 5.7(ii) *and let $A_L = \{\alpha \in \mathbb{T}_{L/k}(k) : L = k(\iota(\alpha))\}$. Then*

(i) *$\mathbb{T}_{L/k}(k)$ is Zariski-dense in $\mathbb{T}_{L/k}$, and*

(ii) *A_L is Zariski-dense in $\mathbb{T}_{L/k}$.*

Proof. By Theorem 5.7(iii), there is a surjective morphism f defined over k from $\mathrm{Res}_{L/k}\mathbb{G}_m$ onto the connected algebraic group $\mathbb{T}_{L/k}$. Since k is infinite and $\mathrm{Res}_{L/k}\mathbb{G}_m$ is rational, $(\mathrm{Res}_{L/k}\mathbb{G}_m)(k)$ is Zariski-dense in $\mathrm{Res}_{L/k}\mathbb{G}_m$. If U is a nonempty open subset of $\mathbb{T}_{L/k}$, then $f^{-1}(U)$ is a nonempty open subset of $\mathrm{Res}_{L/k}\mathbb{G}_m$ and so contains an $x \in (\mathrm{Res}_{L/k}\mathbb{G}_m)(k)$. Then $f(x) \in \mathbb{T}_{L/k}(k) \cap U$. Now (i) follows.

Let τ be a generator of $G := \mathrm{Gal}(L/k)$ and let $n = |G|$. Let $\omega = \prod_{i=1}^{n-1}(1-\tau^i) \in \mathbb{Z}[G]$ and let $W := \ker\gamma(\omega) \subseteq \mathbb{T}_{L/k}$, with γ as in Lemma 10.4. Then W is closed. Since $\prod_{i=1}^{n-1}(1-x^i)$ is not divisible by $\Phi_n(x)$, Lemma 10.4 implies that $\gamma(\omega) \neq 0$, so $W \neq \mathbb{T}_{L/k}$. Suppose $\beta \in \mathbb{T}_{L/k}(k) - A_L$. By the definition of A_L, $L \neq k(\iota(\beta))$, so there is a $j \in \{1,\dots,n-1\}$ such that $\tau^j(\iota(\beta)) = \iota(\beta)$. Thus $\gamma(\tau^j)(\beta) = \beta$, so $\beta \in W(k)$. Thus $\mathbb{T}_{L/k}(k) - A_L \subseteq W(k)$, so $A_L \cup W(k) = \mathbb{T}_{L/k}(k)$. Let A be the Zariski closure of A_L in $\mathbb{T}_{L/k}$. Then $\mathbb{T}_{L/k}(k) \subseteq A(k) \cup W(k)$. By (i), $\mathbb{T}_{L/k} = A \cup W$. Since $\mathbb{T}_{L/k}$ is irreducible and $W \neq \mathbb{T}_{L/k}$, we have $A = \mathbb{T}_{L/k}$, giving (ii). □

Our next goal (Theorem 10.9) is to show that the conjectures in [4] are false when $n = 30$ in almost all characteristics. Since we do not know whether $\mathbb{T}_{30}$ is rational, we cannot find nice coordinates on $\mathbb{T}_{30}$. However, by Lemma 5.4, $\mathbb{T}_{30}$ is isomorphic over $\mathbb{F}_{q^{30}}$ to $\mathbb{T}_G$, which is isomorphic to $\mathbb{G}_m^8$ by Lemma 5.6. Using explicit coordinates on $\mathbb{G}_m^8$, we can take derivatives with respect to these coordinates, as we do below in the proof of Proposition 10.8. We do not know a direct proof of Theorem 10.9, without going through Proposition 10.8.

Suppose Γ is a cyclic group of order 30, and Δ is a subgroup of Γ of index $d = 1$ or 2. Let $u = \lceil\varphi(n)/d\rceil$, and let

$$\mathbf{s}_\Delta := (s_1,\dots,s_u) : \mathbb{X}_\Gamma \longrightarrow (\mathbb{A}^{\Gamma/\Delta})^u.$$

The idea of the proof of Proposition 10.8 is as follows. Suppose for simplicity that $d = 1$, so $\Delta = \Gamma$. We showed in Theorem 10.1 that $\lambda_{\mathbb{F}_7}$ is not injective. Using the counterexample to injectivity constructed there, and the diagram of Proposition 9.10(iv), we deduce (via the computation of a derivative and Hensel's lemma) that $\mathbf{s}_\Gamma$ over $\mathbb{Q}_7$ is generically not injective, so in particular $\mathbf{s}_\Gamma$ over $\mathbb{Q}_7$ is not a birational isomorphism. It follows that $\mathbf{s}_\Gamma$ over $\mathbb{Q}$ is not a birational isomorphism. Reducing mod ℓ shows that $\mathbf{s}_\Gamma$ over $\mathbb{F}_\ell$ is not a birational isomorphism for all but finitely many primes ℓ.

Lemma 10.7. *With notation as in Definition* 5.3, *the function field* $k(\mathbb{X}_\Delta)$ *is generated by the symmetric functions* $\{s_i : 1 \le i \le |\Delta|\}$.

Proof. Apply Theorem 9.8, Proposition 9.10, and Lemma 5.4. □

Proposition 10.8. *Fix a field* k. *There is a finite set* P *of prime numbers such that if* $\mathrm{char}(k) \notin P$, Γ *is a cyclic group of order* 30, *and* Δ *is a subgroup of* Γ *of index* 1 *or* 2, *then the morphism* $\mathbf{s}_\Delta$ *is not a birational isomorphism.*

Proof. Suppose that $\Delta = \Gamma$. The proof when $[\Gamma : \Delta] = 2$ is exactly analogous. Let $\mathbf{s} := \mathbf{s}_\Gamma$. Note that if Ω is an extension field of k, then the morphism $\mathbf{s}$ is a birational isomorphism over k if and only if it is a birational isomorphism over Ω.

Lemma 5.6 gives an isomorphism $\mathbb{G}_m^8 \xrightarrow{\sim} \mathbb{T}_\Gamma \subseteq \mathbb{G}_m^\Gamma$. Let $t_1,\dots,t_8$ be the coordinates on $\mathbb{T}_\Gamma$ induced by this isomorphism. Viewing the restrictions of $s_1,\dots,s_8$ to $\mathbb{T}_\Gamma$ as rational functions of $t_1,\dots,t_8$, let $J : \mathbb{T}_\Gamma \to \mathbb{A}^1$ be the Jacobian determinant $\det\left(\frac{\partial s_i}{\partial t_j}\right)_{i,j=1,\dots,8}$.

Let $\mathbf{x}$ and $\mathbf{y}$ be the image in $\mathbb{T}_\Gamma$, under the isomorphism of Lemma 5.4, of the first two entries in Table 1 (resp., Table 3 in the case $[\Gamma : \Delta] = 2$). Then $\mathbf{x}$ and $\mathbf{y}$ are two elements of $\mathbb{T}_\Gamma(\mathbb{F}_{7^{30}})$, distinct modulo the action of Σ_Γ (since the first 2 rows of the table differ), such that $\mathbf{s}(\mathbf{x}) = \mathbf{s}(\mathbf{y})$ (since the first 8 entries agree). We computed further that $J(\mathbf{x}) \neq 0$ and $J(\mathbf{y}) \neq 0$.

Set $\beta = \mathbf{s}(\mathbf{x}) = \mathbf{s}(\mathbf{y}) \in (\mathbb{F}_{7^{30}})^8$, and let $\tilde{L}$ be the unramified extension of $\mathbb{Q}_7$ of degree 30. Since $J(\mathbf{x}) \neq 0$ and $J(\mathbf{y}) \neq 0$, by Hensel's lemma for every lift $\tilde{\beta}$ of β to $\tilde{L}^8$ we can find unique lifts $\tilde{\mathbf{x}}$ of $\mathbf{x}$ and $\tilde{\mathbf{y}}$ of $\mathbf{y}$ to $\mathbb{T}_\Gamma(\tilde{L})$ such that $\mathbf{s}(\tilde{\mathbf{x}}) = \mathbf{s}(\tilde{\mathbf{y}}) = \tilde{\beta}$. Thus there is an open (in the 7-adic topology) subset $U \subseteq \tilde{L}^8$ contained in the image of $\mathbf{s}$, over which $\mathbf{s}$ is not one-to-one. It follows that as an algebraic map over $\tilde{L}$, $\mathbf{s}$ is dominant and $\deg(\mathbf{s}) > 1$. Therefore, $\mathbf{s}$ is not a birational isomorphism over $\tilde{L}$. The theorem now follows for all k of characteristic zero. Note that we have shown that $\mathbb{Q}(\mathbb{X}_\Gamma)$ is a finite nontrivial extension of $\mathbb{Q}(\mathbb{A}^8)$.

Let $A := \mathbb{Z}[x_1, \ldots, x_8] \subset \mathbb{Q}(\mathbb{A}^8) \subset \mathbb{Q}(\mathbb{X}_\Gamma)$ and $B := \mathbb{Z}[s_1, \ldots, s_{30}]$. Note that A is a subring of B via the map induced by $x_i \mapsto s_i$. The field of fractions $\mathrm{Frac}(B)$ of B is $\mathbb{Q}(\mathbb{X}_\Gamma)$ by Lemma 10.7. Since this field is a finite nontrivial extension of $\mathrm{Frac}(A) = \mathbb{Q}(\mathbb{A}^8)$, we can choose $0 \neq f \in A$ such that $B' := B[1/f]$ is integral over $A' := A[1/f]$ and $A' \neq B'$.

Let P be the (finite) set of prime numbers that divide f in A. Suppose $p \notin P$. Then pA' is a prime ideal of A'. Since $B/pB = \mathbb{F}_p[s_1, \ldots, s_{30}] \subseteq \mathbb{F}_p(\mathbb{X}_\Gamma)$, B/pB is an integral domain, so pB is a prime ideal of B. Since B' is integral over A', p does not divide f in B, so pB' is a prime ideal of B'. Let $A'_{(p)}$ (resp., $B'_{(p)}$) denote the localization of A' (resp., B') at pA' (resp., pB'). Then

$$\mathrm{Frac}(A'_{(p)}) = \mathrm{Frac}(A') = \mathbb{Q}(\mathbb{A}^8) \neq \mathbb{Q}(\mathbb{X}_\Gamma) = \mathrm{Frac}(B') = \mathrm{Frac}(B'_{(p)}). \tag{10.2}$$

Since $A'_{(p)}$ is a Noetherian local domain of dimension one and its maximal ideal $pA'_{(p)}$ is principal, it follows from Proposition 9.2 of [1] that $A'_{(p)}$ is a principal ideal domain. It follows that $B'_{(p)}$ is a free $A'_{(p)}$-module, of rank > 1 by (10.2). Thus

$$\begin{aligned} \mathbb{F}_p(x_1, \ldots, x_8) &= \mathrm{Frac}(A'/pA') = A'_{(p)}/pA'_{(p)} \\ &\neq B'_{(p)} \otimes_{A'_{(p)}} (A'_{(p)}/pA'_{(p)}) = B'_{(p)}/pB'_{(p)} = \mathrm{Frac}(B'/pB') = \mathbb{F}_p(\mathbb{X}_\Gamma). \end{aligned}$$

Thus $\mathbf{s}$ is not a birational isomorphism over $\mathbb{F}_p$, and the same holds with $\mathbb{F}_p$ replaced by any field of characteristic p. □

THEOREM 10.9. *Fix a field k. There is a finite set P of prime numbers such that if* $\mathrm{char}(k) \notin P$, *$L/k$ is cyclic of degree* 30, *and $k \subseteq F \subseteq L$ with $[F : k] = 1$ or* 2, *then the morphism λ_F is not a birational isomorphism.*

Proof. With $\Gamma = \mathrm{Gal}(L/k)$ and $\Delta = \mathrm{Gal}(L/F)$, apply Propositions 9.10(iv,v) and 10.8. □

Remark 10.10. Theorems 10.9 and 10.5 show that Conjectures $(1, 30)$-**BPV** and $(2, 15)$-**BPV** of [4] are false in all but finitely many characteristics.

Acknowledgment. We thank the referees for helpful comments.

REFERENCES

[1] M. F. ATIYAH AND I. G. MACDONALD, *Introduction to Commutative Algebra*, Addison–Wesley, Reading, MA, 1969.

[2] E. BACH AND J. SHALLIT, *Factoring with cyclotomic polynomials*, Math. Comp., 52 (1989), pp. 201–219.

[3] D. BLEICHENBACHER, W. BOSMA, AND A. K. LENSTRA, *Some remarks on Lucas-based cryptosystems*, in Advances in Cryptology (CRYPTO '95), Lecture Notes in Comput. Sci. 963, Springer-Verlag, Berlin, 1995, pp. 386–396.

[4] W. BOSMA, J. HUTTON, AND E. R. VERHEUL, *Looking beyond XTR*, in Advances in Cryptology (Asiacrypt 2002), Lecture Notes in Comput. Sci. 2501, Springer-Verlag, Berlin, 2002, pp. 46–63.

[5] A. E. BROUWER, R. PELLIKAAN, AND E. R. VERHEUL, *Doing more with fewer bits*, in Advances in Cryptology (Asiacrypt '99), Lecture Notes in Comput. Sci. 1716, Springer-Verlag, Berlin, 1999, pp. 321–332.

[6] N. G. DE BRUIJN, *On the factorization of cyclic groups*, Nederl. Akad. Wetensch. Proc. Ser. A, 56 (1953), pp. 370–377 (= Indagationes Math., 15).

[7] M. van Dijk, R. Granger, D. Page, K. Rubin, A. Silverberg, M. Stam, and D. Woodruff, *Practical cryptography in high dimensional tori*, in Advances in Cryptology (EUROCRYPT 2005), Lecture Notes in Comput. Sci. 3494, Springer-Verlag, Berlin, 2005, pp. 234–250.

[8] M. van Dijk and D. Woodruff, *Asymptotically optimal communication for torus-based cryptography*, in Advances in Cryptology (CRYPTO 2004), Lecture Notes in Comput. Sci. 3152, Springer-Verlag, Berlin, 2004, pp. 157–178.

[9] P. Gaudry, *Index Calculus for Abelian Varieties and the Elliptic Curve Discrete Logarithm Problem*, Cryptology ePrint Archive, Report 2004/073, http://eprint.iacr.org/2004/073.

[10] G. Gong and L. Harn, *Public-key cryptosystems based on cubic finite field extensions*, IEEE Trans. Inform. Theory, 45 (1999), pp. 2601–2605.

[11] R. Granger, D. Page, and M. Stam, *A comparison of CEILIDH and XTR*, in Algorithmic Number Theory (ANTS VI), Lecture Notes in Comput. Sci. 3076, Springer-Verlag, Berlin, 2004, pp. 235–249.

[12] R. Granger, D. Page, and M. Stam, *On small characteristic algebraic tori in pairing-based cryptography*, LMS J. Comput. Math., 9 (2006), pp. 64–85.

[13] R. Granger and F. Vercauteren, *On the discrete logarithm problem on algebraic tori*, in Advances in Cryptology (CRYPTO 2005), Lecture Notes in Comput. Sci. 3621, Springer-Verlag, Berlin, 2005, pp. 66–85.

[14] B. Huppert and N. Blackburn, *Finite Groups* II, Springer-Verlag, Berlin, New York, 1982.

[15] A. Joux and R. Lercier, *The function field sieve in the medium prime case*, in Advances in Cryptology (EUROCRYPT 2006), Lecture Notes in Comput. Sci. 4004, Springer-Verlag, Berlin, 2006, pp. 254–270.

[16] A. Joux, R. Lercier, N. Smart, and F. Vercauteren, *The number field sieve in the medium prime case*, in Advances in Cryptology (CRYPTO 2006), Lecture Notes in Comput. Sci. 4117, Springer-Verlag, Berlin, 2006, pp. 323–341.

[17] A. A. Klyachko, *On the rationality of tori with cyclic splitting field*, in Arithmetic and Geometry of Varieties, Kuybyshev University Press, Kuybyshev, Russia, 1988, pp. 73–78 (in Russian).

[18] D. Kohel, *Constructive and Destructive Facets of Torus-Based Cryptography*, preprint, http://echidna.maths.usyd.edu.au/∼kohel/doc/torus.ps, 2004.

[19] A. K. Lenstra, *Using cyclotomic polynomials to construct efficient discrete logarithm cryptosystems over finite fields*, in Information Security and Privacy, Proceedings of ACISP '97, Lecture Notes in Comput. Sci. 1270, Springer-Verlag, Berlin, 1997, pp. 127–138.

[20] A. K. Lenstra, *The XTR Public Key System*, lecture at MSRI Number-Theoretic Cryptography Workshop, Berkeley, CA, 2000.

[21] A. K. Lenstra and E. R. Verheul, *The XTR public key system*, in Advances in Cryptology (CRYPTO 2000), Lecture Notes in Comput. Sci. 1880, Springer-Verlag, Berlin, 2000, pp. 1–19.

[22] H.-W. Leopoldt, *Über die Hauptordnung der ganzen Elemente eines abelschen Zahlkörpers*, J. Reine Angew. Math., 201 (1959), pp. 119–149.

[23] E. Lucas, *Théorie des fonctions numériques simplement périodiques*, Amer. J. Math., 1 (1878), pp. 184–239, 289–321.

[24] B. Mazur, K. Rubin, and A. Silverberg, *Twisting commutative algebraic groups*, J. Algebra, 314 (2007), pp. 419–438.

[25] W. B. Müller and W. Nöbauer, *Some remarks on public-key cryptosystems*, Studia Sci. Math. Hungar., 16 (1981), pp. 71–76.

[26] T. Ono, *Arithmetic of algebraic tori*, Ann. of Math. (2), 74 (1961), pp. 101–139.

[27] K. Rubin and A. Silverberg, *Supersingular abelian varieties in cryptology*, in Advances in Cryptology (CRYPTO 2002), Lecture Notes in Comput. Sci. 2442, Springer-Verlag, Berlin, 2002, pp. 336–353.

[28] K. Rubin and A. Silverberg, *Torus-based cryptography*, in Advances in Cryptology (CRYPTO 2003), Lecture Notes in Comput. Sci. 2729, Springer-Verlag, Berlin, 2003, pp. 349–365.

[29] K. Rubin and A. Silverberg, *Algebraic tori in cryptography*, in High Primes and Misdemeanours: Lectures in Honour of the 60th Birthday of Hugh Cowie Williams, Fields Inst. Commun. 41, AMS, Providence, RI, 2004, pp. 317–326.

[30] K. Rubin and A. Silverberg, *Using primitive subgroups to do more with fewer bits*, in Algorithmic Number Theory (ANTS VI), Lecture Notes in Comput. Sci. 3076, Springer-Verlag, 2004, pp. 18–41.

[31] K. Rubin and A. Silverberg, *Using abelian varieties to improve pairing-based cryptography*, J. Cryptology, to appear. Also available online from http://math.uci.edu/∼asilverb/bibliography/rubsilav.pdf.

[32] I. J. Schoenberg, *A note on the cyclotomic polynomial*, Mathematika, 11 (1964), pp. 131–136.

[33] M. Scott and P. S. L. M. Barreto, *Compressed pairings*, in Advances in Cryptology (CRYPTO 2004), Lecture Notes in Comput. Sci. 3152, Springer-Verlag, Berlin, 2004, pp. 140–156.

[34] P. J. Smith and M. J. J. Lennon, *LUC: A new public key system*, in Proceedings of the IFIP TC11 Ninth International Conference on Information Security (IFIP/Sec '93), North–Holland, Amsterdam, 1993, pp. 103–117.

[35] P. Smith and C. Skinner, *A public-key cryptosystem and a digital signature system based on the Lucas function analogue to discrete logarithms*, in Advances in Cryptology (Asiacrypt 1994), Lecture Notes in Comput. Sci. 917, Springer-Verlag, Berlin, 1995, pp. 357–364.

[36] V. E. Voskresenskiĭ, *Algebraic Groups and Their Birational Invariants*, Transl. Math. Monogr. 179, AMS, Providence, RI, 1998.

[37] V. E. Voskresenskiĭ, *Stably rational algebraic tori*, J. Théor. Nombres Bordeaux, 11 (1999), pp. 263–268.

[38] A. Weil, *Adeles and Algebraic Groups*, Progr. Math. 23, Birkhäuser Boston, Boston, 1982.

[39] H. C. Williams, *A $p+1$ method of factoring*, Math. Comp., 39 (1982), pp. 225–234.

[40] H. C. Williams, *Some public-key crypto-functions as intractable as factorization*, Cryptologia, 9 (1985), pp. 223–237.

SIAM J. Comput.
Vol. 37, No. 5, pp. 1429–1454

ACCELERATING SIMULATED ANNEALING FOR THE PERMANENT AND COMBINATORIAL COUNTING PROBLEMS*

IVONA BEZÁKOVÁ[†], DANIEL ŠTEFANKOVIČ[‡], VIJAY V. VAZIRANI[§], AND ERIC VIGODA[§]

Abstract. We present an improved "cooling schedule" for simulated annealing algorithms for combinatorial counting problems. Under our new schedule the rate of cooling accelerates as the temperature decreases. Thus, fewer intermediate temperatures are needed as the simulated annealing algorithm moves from the high temperature (easy region) to the low temperature (difficult region). We present applications of our technique to colorings and the permanent (perfect matchings of bipartite graphs). Moreover, for the permanent, we improve the analysis of the Markov chain underlying the simulated annealing algorithm. This improved analysis, combined with the faster cooling schedule, results in an $O(n^7 \log^4 n)$ time algorithm for approximating the permanent of a 0/1 matrix.

Key words. Markov chain Monte Carlo, simulated annealing, cooling schedule, approximate counting problems

AMS subject classifications. 68W20, 68W25, 68W40

DOI. 10.1137/050644033

1. Introduction.

Simulated annealing is an important algorithmic approach for counting and sampling combinatorial structures. Two notable combinatorial applications are estimating the partition function of statistical physics models and approximating the permanent of a nonnegative matrix. For combinatorial counting problems, the general idea of simulated annealing is to write the desired quantity, say, Z, (which is, for example, the number of colorings or matchings of an input graph) as a telescoping product:

$$Z = \frac{Z_{\ell+1}}{Z_\ell}\frac{Z_\ell}{Z_{\ell-1}}\dots\frac{Z_1}{Z_0}Z_0, \tag{1.1}$$

where $Z_{\ell+1} = Z$ and Z_0 is trivial to compute. By further ensuring that each of the ratios Z_i/Z_{i-1} is bounded, a small number of samples (from the probability distribution corresponding to Z_{i-1}) suffices to estimate the ratio. These samples are typically generated from an appropriately designed Markov chain.

Each of the quantities of interest corresponds to the counting problem at a different temperature. The final quantity $Z = Z_{\ell+1}$ corresponds to the zero temperature, whereas the trivial initial quantity Z_0 is the infinite temperature. The temperature

*Received by the editors November 1, 2005; accepted for publication (in revised form) July 18, 2007; published electronically January 16, 2008. An extended abstract of this paper appeared in *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA), ACM, New York, SIAM, Philadelphia, 2006, pp. 900–907.

http://www.siam.org/journals/sicomp/37-5/64403.html

[†] Department of Computer Science, Rochester Institute of Technology, Rochester, NY 14623 (ib@cs.rit.edu). This author's research was supported by NSF grant CCR-0237834.

[‡] Department of Computer Science, University of Rochester, Rochester, NY 14627 (stefanko@cs.rochester.edu).

[§] College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 (vazirani@cc.gatech.edu, vigoda@cc.gatech.edu). The third author's research was supported by NSF grants CCR-0311541 and CCR-0220343. The fourth author's research was supported by NSF grant CCR-0455666.

slowly decreases from a high temperature (easy region) to a low temperature (difficult region). A notable application of simulated annealing to combinatorial counting is the algorithm of Jerrum, Sinclair, and Vigoda [10] for approximating the permanent of a nonnegative matrix. In their algorithm, the cooling schedule is uniform: the rate of cooling was constant.

Our first main result is an improved cooling schedule. In contrast to the previous cooling schedule for the permanent, our schedule is accelerating (the rate of cooling accelerates as the temperature decreases). Consequently, fewer intermediate temperatures are needed, and thus fewer Markov chain samples overall suffice. It is interesting to note that our schedule is similar to the original proposal of Kirkpatrick, Gelatt, and Vecchi [13] and is related to schedules used recently in geometric settings by Lovász and Vempala [14] and Kalai and Vempala [11].

We illustrate our new cooling schedule in the context of colorings, corresponding to the antiferromagnetic Potts model from statistical physics. We present general results defining a cooling schedule for a broad class of counting problems. These general results seem applicable to a wide range of combinatorial counting problems, such as the permanent, and binary contingency tables [1].

The permanent of an $n \times n$ matrix A is defined as

$$\mathrm{per}(A) = \sum_{\sigma} \prod_{i=1}^{n} a_{i,\sigma(i)},$$

where the sum goes over all permutations σ of $[n]$. The permanent of a $0/1$ matrix A is the number of perfect matchings in the bipartite graph with bipartite adjacency matrix A. In addition to traditional applications in statistical physics [12], the permanent has recently been used in a variety of areas, e.g., computer vision [16] and statistics [15]. Jerrum, Sinclair, and Vigoda presented a simulated annealing algorithm [10] for the permanent of nonnegative matrices with running time $O(n^{10} \log^3 n)$ for $0/1$ matrices.

Our cooling schedule reduces the number of intermediate temperatures in the simulated annealing for the permanent from $O(n^2 \log n)$ to $O(n \log^2 n)$. We also improve the analysis of the Markov chain used for sampling. The improved analysis comes from several new inequalities relating sets of perfect matchings in bipartite graphs. The consequence of the new analysis and improved cooling schedule is an $O(n^7 \log^4 n)$ time algorithm for estimating the permanent of a $0/1$ $n \times n$ matrix. Here is the formal statement of our result.

THEOREM 1.1. *For all $\varepsilon > 0$, there exists a randomized algorithm to approximate, within a factor $(1 \pm \varepsilon)$, the permanent of a $0/1$ $n \times n$ matrix A in time $O(n^7 \log^4(n) + n^6 \log^5(n)\varepsilon^{-2})$. The algorithm extends to arbitrary matrices with nonnegative entries.*

The remainder of the paper is organized as follows. We introduce some basic machinery and definitions in the following section. In section 3 we present our new cooling schedule, motivated by its application to colorings. We then focus on the permanent in section 4. We begin by presenting the simulated annealing algorithm for the permanent in section 4. In section 5 we explain the background techniques for analyzing the Markov chain. We present our new inequalities in section 6. Finally, in section 7 we use these new inequalities for bounding the mixing time of the Markov chain. We then conclude the analysis of the permanent algorithm for $0/1$ matrices in section 8.

2. Preliminaries.

2.1. Colorings and Potts model. Let $G = (V, E)$ be the input graph and k be the number of colors. A (valid) k-coloring of G is an assignment of colors from $[k]$ to the vertices of G such that no two adjacent vertices are colored by the same color (i.e., $\sigma(u) \neq \sigma(v)$ for every $(u, v) \in E$). Let $\Omega = \Omega(G)$ denote the set of all k-colorings of G. For input parameters ϵ, δ, our goal is to approximate $|\Omega|$ within a multiplicative factor $1 \pm \epsilon$ with probability $\geq 1 - \delta$. This is commonly known as a fully polynomial randomized approximation scheme (fpras) for counting colorings.

The colorings problem corresponds to the zero-temperature version of the antiferromagnetic Potts model. In addition to the underlying graph G and the number of colors k, the Potts model is also specified by an activity[1] λ. The configuration space of the Potts model, denoted $[k]^V$, is the set of all labelings $\sigma : V \to [k]$. The partition function of the Potts model counts the number of configurations weighted by their distance from a k-coloring. More precisely, for activity $\lambda \geq 0$, the partition function of the Potts model is

$$Z(\lambda) = \sum_{\sigma \in [k]^V} \lambda^{M(\sigma)},$$

where $M(\sigma) = M_G(\sigma) = |(u, v) \in E : \sigma(u) = \sigma(v)|$ is the number of monochromatic edges of the labeling σ. For $\lambda = 0$ we also define $0^0 = 1$, and thus $Z(0) = |\Omega|$. In section 3.1 we will consider simulated annealing algorithms for estimating the partition function of the Potts model.

An elementary component of the upcoming algorithms is the following approach for estimating the ratio of the partition function at a pair of temperatures. In particular, for a sequence $1 = \lambda_0 > \lambda_1 > \cdots > \lambda_\ell > \lambda_{\ell+1} = 0$ we will estimate the ratios $\alpha_i := Z(\lambda_{i+1})/Z(\lambda_i)$ for all $0 \leq i \leq \ell$. Assuming $1/2 \leq \alpha_i \leq 1$, we can approximate α_i efficiently using the following unbiased estimator. Let $X_i \sim \pi_i$ denote a random labeling chosen from the distribution π_i defined by $Z(\lambda_i)$ (i.e., the probability of a labeling σ is $\pi_i(\sigma) = {\lambda_i}^{M(\sigma)}/Z(\lambda_i)$). Let $Y_i = (\lambda_{i+1}/\lambda_i)^{M(X_i)}$. Then Y_i is an unbiased estimator for α_i:

$$\mathbf{E}(Y_i) = \mathbf{E}_{X_i \sim \pi_i}\left((\lambda_{i+1}/\lambda_i)^{M(X_i)}\right) = \sum_{\sigma \in [k]^V} \frac{(\lambda_{i+1})^{M(\sigma)}}{Z(\lambda_i)} = \frac{Z(\lambda_{i+1})}{Z(\lambda_i)} = \alpha_i. \tag{2.1}$$

The expected value of $Y = Y_0 Y_1 \ldots Y_\ell$ is

$$\mathbf{E}(Y) = \prod_{i=0}^{\ell} \mathbf{E}(Y_i) = \frac{Z(0)}{Z(1)},$$

where $Z(1)$ is easy to compute. Thus, our goal of estimating $Z(0) = |\Omega|$ can be reduced to estimating $\mathbf{E}(Y)$.

Assume that we have an algorithm for generating labelings X_i from π_i. We draw $64(\ell + 1)/\varepsilon^2$ samples of X_i and take the mean $\overline{Y}_i$ of their corresponding estimators Y_i. We have

$$\frac{\mathbf{Var}\left(\overline{Y}_i\right)}{\mathbf{E}\left(\overline{Y}_i\right)^2} = \frac{\varepsilon^2}{64(\ell + 1)} \frac{\mathbf{Var}(Y_i)}{\mathbf{E}(Y_i)^2} \leq \frac{\varepsilon^2}{16(\ell + 1)}.$$

[1] The activity corresponds to the temperature of the system. Specifically, the temperature is $-1/\ln \lambda$; thus $\lambda = 1$ corresponds to the infinite temperature and $\lambda = 0$ corresponds to the zero temperature.

Hence for $\overline{Y} = \overline{Y}_0\overline{Y}_1 \dots \overline{Y}_\ell$ we have

$$\frac{\mathbf{Var}\left(\overline{Y}\right)}{\mathbf{E}\left(\overline{Y}\right)^2} = \left(1 + \frac{\mathbf{Var}\left(\overline{Y}_0\right)}{\mathbf{E}\left(\overline{Y}_0\right)^2}\right) \dots \left(1 + \frac{\mathbf{Var}\left(\overline{Y}_\ell\right)}{\mathbf{E}\left(\overline{Y}_\ell\right)^2}\right) - 1 \le \mathrm{e}^{\varepsilon^2/16} - 1 \le \varepsilon^2/8,$$

where in the last two inequalities we used $1 + x \le \mathrm{e}^x$ (true for all x), and $\mathrm{e}^x - 1 \le 2x$ (true for $x \in [0, 1]$). Now, by Chebyshev's inequality, with probability at least 7/8 we have that the value of $\overline{Y}$ is in the interval $[(1-\varepsilon)\mathbf{E}\left(Y\right), (1+\varepsilon)\mathbf{E}\left(Y\right)]$.

Of course, we will not be able to obtain perfect samples from π_i. Assume now that we have X_i' which are from a distribution with a variation distance $\le \varepsilon^2/(512(\ell+1)^2)$ of π_i (we choose the variation distance to be 1/8th of the reciprocal of the number of all samples). Let $\overline{Y}'$ be defined as $\overline{Y}$ above, but instead of X_i we will use X_i'. If we couple X_i with X_i' optimally, then with probability $\ge 7/8$ we have $\overline{Y} = \overline{Y}'$. Hence, $\overline{Y}'$ is in the interval $[(1-\varepsilon)\mathbf{E}\left(Y\right), (1+\varepsilon)\mathbf{E}\left(Y\right)]$ with probability $\ge 3/4$.

2.2. Markov chain basics. For a pair of distributions μ and π on a finite space Ω we will measure their distance using *variation distance*, defined to be

$$d_{\mathrm{TV}}(\mu, \pi) = \frac{1}{2}\sum_{x\in\Omega} |\mu(x) - \pi(x)|.$$

For an ergodic Markov chain with finite state space Ω, transition matrix P, and unique stationary distribution π, we are interested in the *mixing time*, defined to be

$$\tau(\delta) = \max_{x\in\Omega} \tau_x(\delta),$$

where

$$\tau_x(\delta) = \min\{t : d_{\mathrm{TV}}(P^t(x, \cdot), \pi) \le \delta\}.$$

In the case of the permanent, we will bound the mixing time by the canonical paths method. For some $S \subseteq \Omega$, for each $(I, F) \in \Omega\times S$, we will define a *canonical path* from I to F, denoted $\gamma(I, F)$, which is of length $\le \ell$. The path is along transitions of the Markov chain (i.e., along pairs $(x, y) \in \Omega^2$ where $P(x, y) > 0$). We then bound the weighted sum of canonical paths (or "flow") through any transition. More precisely, for a transition $T = x \to y$, let

$$\rho(T) = \sum_{\substack{(I,F)\in\Omega\times\mathcal{P}:\\ T\in\gamma(I,F)}} \frac{\pi(I)\pi(F)}{\pi(x)P(x, y)}$$

denote the *congestion* through the transition T.

Let

$$\rho = \max_T \rho(T).$$

Then (see [17, 4]), for any initial state $x \in \Omega$, the mixing time is bounded as

$$\tau_x(\delta) \le \frac{7\ell\rho}{\pi(S)}\left(\ln \pi(x)^{-1} + \ln \delta^{-1}\right).$$

The factor $1/\pi(S)$ comes from restricting to $F \in S$; see Lemma 9 in [10].

3. Improved cooling schedule.

We begin by motivating the simulated annealing framework in the context of colorings. We then present a general method for obtaining improved cooling schedules and show how it can be applied to colorings. We conclude with the proofs of technical lemmas for improved cooling schedules.

3.1. Basic cooling schedule for counting colorings.

Our focus in this section is obtaining an fpras for counting all k-colorings of a given graph G. Let $\Omega = \Omega(G)$ denote the set of k-colorings of G. We are of course considering only cases where $|\Omega| \geq 1$. There are various situations where a polynomial-time algorithm for approximating $|\Omega|$ exists; see, e.g., [5] for a survey and [6] for a more recent result when $k = \Omega(\Delta/\log \Delta)$ for planar graphs. Our aim is to improve the running time of these approximate counting algorithms.

For the purposes of reducing the approximation of $|\Omega|$ to sampling from Ω, we will define a sequence of activities of the antiferromagnetic Potts model (defined in section 2.1). We will express $|\Omega|$ as a telescoping product over instances of the Potts model where we slowly move from the original k-colorings (corresponding to activity $\lambda = 0$) to a trivial instance of the Potts model, namely, $\lambda = 1$, since $Z(1) = k^n$. We specify a sequence of activities so that the partition functions do not change by more than a constant factor between successive activities. This allows us to reduce the activity to an almost zero value while being able to estimate the ratios of two consecutive partition functions.

The partition function $Z(\lambda)$ can be viewed as a polynomial in λ. Notice that its constant coefficient equals $|\Omega|$, the number of k-colorings of G. Moreover, $Z(1) = |\Omega(G_m)| = k^n$ is the sum of the coefficients of Z. It can be shown that for $k > \Delta$, the number of k-colorings of G is bounded from below by $(k/e)^n$ (i.e., $|\Omega| \geq (k/e)^n$). For completeness, we prove this lower bound in the appendix in Corollary A.2. If we used the trivial lower bound of $|\Omega| \geq 1$, we would introduce an extra factor of $O(\log k)$ in the final running time. Observe that the value of the partition function at $\lambda = 1/e^n$ is at most $2|\Omega|$:

$$|\Omega| \leq Z(1/e^n) \leq |\Omega| + Z(1)(1/e^n) \leq |\Omega| + k^n/e^n \leq 2|\Omega|. \tag{3.1}$$

This will be sufficiently close to $|\Omega|$ so that we can obtain an efficient estimator for $|\Omega|$.

We will define a sequence, called a *cooling schedule*,

$$\lambda_0 = 1, \lambda_1, \ldots, \lambda_\ell \leq 1/e^n, \lambda_{\ell+1} = 0,$$

where $\ell = O(n \log n)$, and, for all $0 \leq i \leq \ell$,

$$\frac{1}{2} \leq \frac{Z(\lambda_{i+1})}{Z(\lambda_i)} \leq 1.$$

Notice that for $i = \ell$ the inequality follows from (3.1), so we need to take care of $i < \ell$. We estimate the number of k-colorings of G via the telescoping product

$$|\Omega| = k^n \prod_{0 \leq i \leq \ell} \alpha_i,$$

where $\alpha_i = Z(\lambda_{i+1})/Z(\lambda_i)$. We will estimate α_i by sampling from the probability distribution corresponding to Z_i as described in section 2.1.

A straightforward cooling schedule sets $\lambda_{i+1} = 2^{-1/m}\lambda_i$. Then,

$$Z(\lambda_{i+1}) \geq (2^{-1/m})^m Z(\lambda_i) = Z(\lambda_i)/2,$$

as required. This specifies a uniform cooling schedule with a rate of decrease $2^{-1/m}$. Note that once $\lambda_\ell \le k^{-n}$ we can set $\lambda_{\ell+1} = 0$ since we then have $Z(\lambda_\ell) \le |\Omega| + 1 \le 2\Omega$, assuming $|\Omega| \ge 1$. Therefore, this uniform cooling schedule is of length $\ell = O(nm \log k)$. We present a new cooling schedule which is only of length $O(n \log n)$.

Our goal is to obtain a general cooling schedule which applies to all instances of the colorings problem and which will also apply to many other combinatorial problems. If we restrict our attention to certain regions of k versus Δ, sometimes a straightforward telescoping product is more efficient for colorings. In particular, assume $k \ge (1+\varepsilon)\Delta$, where $\varepsilon > 0$ is a constant; then the removal of all at most Δ edges adjacent to one of the vertices increases the number of colorings by a factor at most $k/(k-\Delta) \le (1+\varepsilon)/\varepsilon$. Hence, in this case one can obtain a schedule of length $O(n)$, but such a schedule does not apply, for example, to the previously mentioned results of [6], which hold for $k = \Omega(\Delta/\log\Delta)$ for planar graphs.

3.2. New cooling schedule. Note that if we had $Z(\lambda) = k^n\lambda^m$, we could not decrease λ faster than $2^{-1/m}$. Fortunately, in our case the constant term of $Z(\lambda)$ is at least one. To illustrate the idea of nonuniform decrease, let $f_i(\lambda) = \lambda^i$. As we decrease λ, the polynomial f_m will always decrease faster (in a relative sense) than Z. At first (for values of λ close to 1) this difference will be small; however, as λ goes to 0, the rate of decrease of Z slows because of its constant term. Thus, at a certain point f_{m-1} will decrease faster than Z. Once λ reaches this point, we can start decreasing λ by a factor of $2^{-1/(m-1)}$. As time progresses, the rate of Z will be bounded by the rate of polynomials f_m; then $f_{m-1}, f_{m-2}, \ldots$, all the way down to f_1 for λ close to 0. When the polynomial f_i "dominates" we can decrease λ by a factor of $2^{-1/i}$. Note that the rate of decrease increases with time; i.e., the schedule is accelerating.

Now we formalize the accelerated cooling approach. We state our results in a general form which proves useful in other contexts, e.g., for the permanent later in this paper, and binary contingency tables [1].

Let $Z(\lambda)$ be the partition function polynomial. Let s be the degree of $Z(\lambda)$ (note that $s = m$ for colorings). Our goal is to find $1 = \lambda_1 > \lambda_2 > \cdots > \lambda_\ell > \lambda_{\ell+1} = 0$ such that $Z(\lambda_i)/Z(\lambda_{i+1}) \le c$ (e.g., for colorings we took $c = 2$). The important property of $Z(\lambda)$ for colorings is $Z(0) \ge 1$ (e.g., $Z(\lambda)$ has positive constant term). In fact, when $k > \Delta$, we have $Z(0) \ge (k/e)^n$, which will save a factor of $O(\log k)$ in the final result. For completeness, we prove this lower bound in the appendix in Corollary A.2.

For some applications it will not be possible to make the constant term positive; instead we will show that a coefficient a_D of λ^D is large (for some small D). Finally, let γ be an upper bound on $Z(1)/a_D$. For colorings we can take $\gamma = e^n$. The γ measures how small λ needs to get for $Z(\lambda)$ to be within constant factor of $Z(0)$. Now we present a general algorithm in terms of parameters s, c, γ, D.

Algorithm for computing the cooling schedule λ, given parameters s, c, γ, and D:

```
Set $\lambda_0 = 1$, $i = s$, and $j = 0$.
While $\lambda_j > 1/\gamma$ do
    Set $\lambda_{j+1} = c^{-1/i}\lambda_j$.
    If $i > D + 1$ and $\lambda_{j+1} < (s/\gamma)^{1/(i-D)}$,
        Set $\lambda_{j+1} = (s/\gamma)^{1/(i-D)}$ and decrement $i = i - 1$.
    Increment $j = j + 1$.
Set $\ell = j$.
```

The following lemmas prove that the above algorithm produces a short schedule. We prove the lemmas in section 3.3. The first lemma bounds the number of intermediate temperatures in the above cooling schedule, i.e., the length ℓ of the λ sequence.

Lemma 3.1. *Let $c, \gamma > 0$, $D \geq 0$, and let $\lambda_0, \dots, \lambda_\ell$ be the sequence computed by the above algorithm. Then $\ell = O([(D+1)\log(s-D) + s/(s-D)]\log_c(s\gamma))$. If c and D are constants independent of s, then $\ell = O((\log s)\log(s\gamma))$.*

Note that for colorings $\ell = O(n(\log n)\log k)$ (assuming $Z(0) \geq 1$), and when $k > \Delta$, we have $\ell = O(n \log n)$ since $Z(0) \geq (k/e)^n$.

The following lemma shows that for the sequence of the λ_i the value of $Z(\lambda)$ changes by a factor $\leq c$ for consecutive λ_i and λ_{i+1}. For the later application to the permanent, we will need to simultaneously consider a collection of polynomials. Therefore, we state the following lemma in this more general context.

Lemma 3.2. *Let $c, \gamma, D \geq 0$, and let $Z_1, \dots, Z_q$ be a collection of polynomials of degree s. Suppose that for every $i \in [q]$, the polynomial Z_i satisfies the following conditions*:

(i) *Z_i has nonnegative coefficients.*

(ii) *There exists $d \leq D$ such that the coefficient of x^d in Z_i is at least $Z_i(1)/\gamma$.*

Let $\lambda_0, \lambda_1, \dots, \lambda_\ell$ be the sequence constructed by the above algorithm. Then

$$Z_i(\lambda_j) \leq cZ_i(\lambda_{j+1}) \qquad \textit{for every } i \in [q] \textit{ and } j \in [\ell].$$

Recall from section 2.1 that to estimate $\prod_{i=0}^{\ell} \alpha_i$ within a factor $(1 \pm \varepsilon)$ with probability $\geq 3/4$ we need to generate $O(\ell/\varepsilon^2)$ samples from within variation distance $O(\varepsilon^2/\ell^2)$ of π_i for all $i = 0, \dots, \ell$. To illustrate the application of the shorter cooling schedule, recall that for colorings, when $k > \Delta$ we have $\ell = O(n \log n)$. Hence, we need a total of $O(n^2\varepsilon^{-2}\log^2 n)$ samples. For $k > 2\Delta$, for all activities $1 \leq \lambda \leq 0$, there is a Markov chain with mixing time $T(\varepsilon) = \frac{k}{k-2\Delta} n \log(n/\varepsilon)$ [3, 8]. Consequently, we can approximate $|\Omega|$ within a multiplicative factor $1 \pm \varepsilon$ with probability $\geq 3/4$ in $O(\frac{k}{k-2\Delta}\frac{n^3 \log^2 n}{\varepsilon^2}\ln(n/\varepsilon))$ time.

For $k \leq 2\Delta$ there are a variety of results showing fast convergence of Markov chains for generating a random k-coloring [5]. These results are proved for k-colorings, but they can most likely be extended to the nonzero temperature. One particular example is the previously mentioned result of [6] which for planar graphs shows $O^*(n)$ mixing time of a Markov chain when $k = \Omega(\Delta/\log \Delta)$ for all activities. (The $O^*()$ notation hides logarithmic factors and the dependence on ϵ.) Consequently, in this case we again obtain an $O^*(n^3)$ time algorithm for approximating the number of k-colorings.

3.3. Proof of Lemmas 3.1 and 3.2.

The rest of this section is devoted to the proof of Lemmas 3.1 and 3.2.

Proof of Lemma 3.1. We define the following intervals:

$$I_i = \begin{cases} [(s/\gamma)^{1/(s-D)}, \infty) & \text{for } i = s, \\ [(s/\gamma)^{1/(i-D)}, (s/\gamma)^{1/(i+1-D)}] & \text{for } D+1 < i < s, \\ (0, (s/\gamma)^{1/2}] & \text{for } i = D+1. \end{cases}$$

Let ℓ_i be the number of λ values lying in the interval I_i. For $i \in \{D+2, \dots, s-1\}$ we have the estimate

$$\ell_i \leq \log_c\left(\frac{[(s/\gamma)^{1/(i+1-D)}]^i}{[(s/\gamma)^{1/(i-D)}]^i}\right) \leq \frac{D+1}{i-D}\log_c \gamma.$$

Similarly,

$$\ell_s \leq \log_c\left(\frac{\gamma}{[(s/\gamma)^{1/(s-D)}]^s}\right) \leq \frac{2s-D}{s-D}\log_c \gamma,$$

and

$$\ell_{D+1} \leq \log_c \left(\frac{[(s/\gamma)^{1/2}]^{D+1}}{[1/\gamma]^{D+1}} \right) = \frac{D+1}{2} \log_c(s\gamma).$$

Recall that $s \geq 1$. Putting it all together, we get the bound

$$\ell \leq \sum_{i=D+1}^{s} \ell_i \leq \left((D+1)H_{s-D} + \frac{2s-D}{s-D} + \frac{D+1}{2} \right) \log_c(s\gamma),$$

where $H_i = \sum_{j=1}^{i} 1/j = O(\log i)$ is the harmonic sum. Therefore,

$$\ell = O([(D+1)\log(s-D) + s/(s-D)] \log_c(s\gamma)). \qquad \square$$

We now present a few preliminary lemmas before proving Lemma 3.2. The *log-derivative* of a function f is $(\log f)' = f'/f$. The log-derivative measures how quickly a function increases.

Definition 3.3. *We say that a polynomial f is* dominant *over a polynomial g on an interval I if $f'(x)/f(x) \geq g'(x)/g(x)$ for every $x \in I$.*

Lemma 3.4. *Let $f, g : I \to \mathbf{R}^+$ be two nondecreasing polynomials. If f dominates over g on I, then $f(y)/f(x) \geq g(y)/g(x)$ for every $x, y \in I$, $x \leq y$.*

We partition the interval $(0, \infty)$ into subintervals $I_{D+1}, \ldots, I_s$ such that x^i dominates over every Z-polynomial on the interval I_i. The λ_j in I_i will be such that x^i decreases by a factor c between consecutive λ. Therefore, the Z-polynomials decrease by at most a factor of c.

Lemma 3.5. *Let $g(x) = \sum_{j=0}^{s} a_j x^j$ be a polynomial with nonnegative coefficients. Then x^s dominates over g on the interval $(0, \infty)$.*

Proof. It suffices to verify that $(x^s)'/x^s \geq g'(x)/g(x)$ for every $x > 0$. $\square$

Lemma 3.6. *Let $g(x) = \sum_{j=0}^{s} a_j x^j$ be a polynomial with nonnegative coefficients such that $g(1) \leq \gamma$ and at least one of $a_0, a_1, \ldots, a_D$ is ≥ 1. Then for any $i \geq D+1$ the polynomial x^i dominates over g on the interval $(0, (s/\gamma)^{1/(i+1-D)}]$.*

Proof. The log-derivative of x^i is i/x. Hence we need to prove that $ig(x) \geq xg'(x)$ for $x \leq (s/\gamma)^{1/(i+1-D)}$.

Let d be the smallest integer such that $a_d \geq 1$. From the assumptions of the lemma, $d \leq D$. For $x \leq (s/\gamma)^{1/(i+1-D)}$ the following holds:

$$\sum_{j=i+1}^{s} j a_j x^{j-d} \leq \sum_{j=i+1}^{s} s a_j x^{j-D} \leq \sum_{j=i+1}^{s} s a_j \left(\frac{s}{\gamma}\right)^{(j-D)/(i+1-D)} \leq \sum_{j=i+1}^{s} s a_j \left(\frac{s}{\gamma}\right) \leq 1.$$

Since $i > d$, for $x \leq (s/\gamma)^{1/(i+1-D)}$ we have

$$xg'(x) = \sum_{j=0}^{i} j a_j x^j + \sum_{j=i+1}^{s} j a_j x^j \leq \sum_{j=d}^{i} j a_j x^j + a_d x^d \leq \sum_{j=d}^{i} i a_j x^j \leq ig(x). \qquad \square$$

Proof of Lemma 3.2. Let $I_{D+1}, \ldots, I_s$ be as in the proof of Lemma 3.1. Let $Q_q(\lambda) = \gamma Z_q(\lambda)/Z_q(1)$. Notice that the Q_q satisfy the conditions required of g by Lemma 3.6. Therefore, x^i dominates over every Q_q (and hence also every Z_q) on the interval I_i for $i < s$. Moreover, Lemmas 3.6 and 3.4 imply that x^s dominates over every Q_q (and hence every Z_q) on the interval I_s. Notice that if $\lambda_j, \lambda_{j+1} \in I_i$, then $c\lambda_{j+1}^i \geq \lambda_j^i$ (where inequality happens only if $\lambda_{j+1} = (s/\gamma)^{1/(i-D)}$). Therefore, all of the Z_q-polynomials decrease by a factor of at most c between consecutive values of λ. $\square$

4. Permanent algorithm. Here we describe the simulated annealing algorithm for the permanent. For simplicity we consider the case of a 0/1 matrix A. The generalization to nonnegative matrices proceeds as in [10]. We assume $per(A) > 0$; i.e., there is at least one perfect matching.

We show the application of our improved cooling schedule and our improvement in the mixing time bound for the Markov chain underlying the simulated annealing algorithm. We present the new inequalities which are key to the improved mixing time result in section 6.

4.1. Preliminaries. Let $G = (V_1, V_2, E)$ be a bipartite graph with $|V_1| = |V_2| = n$. We will let $u \sim v$ denote the fact that $(u, v) \in E$. For $u \in V_1$, $v \in V_2$ we will have a positive real number $\lambda(u, v)$ called the *activity* of (u, v). If $u \sim v$, $\lambda(u, v) = 1$ throughout the algorithm, and otherwise, $\lambda(u, v)$ starts at 1 and drops to $1/n!$ as the algorithm evolves. Once nonedges have activity $\leq 1/n!$ we have that the total activity of perfect matchings containing at least one nonedge is < 1, and hence they alter the permanent by at most a factor of 2. The activities allow us to work on the complete bipartite graph between V_1 and V_2.

Let $\mathcal{P}$ denote the set of perfect matchings (recall that we are working on the complete graph now), and let $\mathcal{N}(u, v)$ denote the set of near-perfect matchings with holes (or unmatched vertices) at u and v. Similarly, let $\mathcal{N}(x, y, w, z)$ denote the set of matchings that have holes only at the vertices x, y, w, z. Let $\mathcal{N}_i$ denote the set of matchings with exactly i unmatched vertices. The set of states of the Markov chain is $\Omega = \mathcal{P} \cup \mathcal{N}_2$. For any matching M, denote its activity as

$$\lambda(M) := \prod_{(u,v)\in M} \lambda(u, v).$$

For a set S of matchings, let $\lambda(S) := \sum_{M\in S} \lambda(M)$. For $u \in V_1$, $v \in V_2$ we will have a positive real number $w(u, v)$ called the *weight* of the hole pattern u, v. Given weights w, the weight of a matching $M \in \Omega$ is

$$w(M) := \begin{cases} \lambda(M)w(u, v) & \text{if } M \in \mathcal{N}(u, v), \text{ and} \\ \lambda(M) & \text{if } M \in \mathcal{P}. \end{cases}$$

The weight of a set S of matchings is

$$w(S) := \sum_{M\in S} w(M).$$

For given activities, the *ideal weights* on hole patterns are the following:

$$w^*(u, v) := \frac{\lambda(\mathcal{P})}{\lambda(\mathcal{N}(u, v))}. \tag{4.1}$$

Note that for the ideal weights all the $\mathcal{N}(u, v)$ and $\mathcal{P}$ have the same weight; i.e., $w^*(\mathcal{P}) = w^*(\mathcal{N}(u, v))$ for all u, v. Hence, since $w^*(\mathcal{P}) = \lambda(\mathcal{P})$, we have $w^*(\Omega) = (n^2 + 1)\lambda(\mathcal{P})$.

For the purposes of the proof, we need to extend the weights to 4-hole matchings. Let

$$w^*(x, y, w, z) := \frac{\lambda(\mathcal{P})}{\lambda(\mathcal{N}(x, y, w, z))},$$

and for $M \in \mathcal{N}(x, y, w, z)$, let

$$w^*(M) := \lambda(M)w^*(x, y, w, z).$$

4.2. Markov chain definition. At the heart of the algorithm lies a Markov chain MC, which was used in [10], and a slight variant was used in [2, 9]. Let $\lambda : V_1 \times V_2 \rightarrow \mathbb{R}_+$ be the activities and $w : V_1 \times V_2 \rightarrow \mathbb{R}_+$ be the weights. The state space is Ω, the set of all perfect and near-perfect matchings of the complete bipartite graph on V_1, V_2. The stationary distribution π is proportional to w; i.e., $\pi(M) = w(M)/Z$, where $Z = \sum_{M \in \Omega} w(M)$.

The transitions $M_t \rightarrow M_{t+1}$ of the Markov chain MC are defined as follows:

1. If $M_t \in \mathcal{P}$, choose an edge e uniformly at random from M_t. Set $M' = M_t \setminus e$.
2. If $M_t \in \mathcal{N}(u,v)$, choose vertex x uniformly at random from $V_1 \cup V_2$.
    - (a) If $x \in \{u, v\}$, let $M' = M \cup (u,v)$.
    - (b) If $x \in V_2$ and $(w,x) \in M_t$, let $M' = M \cup (u,x) \setminus (w,x)$.
    - (c) If $x \in V_1$ and $(x,z) \in M_t$, let $M' = M \cup (x,v) \setminus (x,z)$.
3. With probability $\min\{1, w(M')/w(M_t)\}$, set $M_{t+1} = M'$; otherwise, set $M_{t+1} = M_t$.

Note that cases 1 and 2(a) move between perfect and near-perfect matchings, whereas cases 2(b) and 2(c) move between near-perfect matchings with different hole patterns. Case 3 applies the Metropolis filter, which ensures that the stationary distribution of the Markov chain is proportional to w.

The key technical theorem is that the Markov chain quickly converges to the stationary distribution π if the weights w are close to the ideal weights w^*. The mixing time $\tau(\delta)$ is the time needed for the chain to be within variation distance δ from the stationary distribution.

THEOREM 4.1. *Assuming the weight function w satisfies inequality*

$$w^*(u,v)/2 \leq w(u,v) \leq 2w^*(u,v) \tag{4.2}$$

for every $(u,v) \in V_1 \times V_2$ with $\mathcal{N}(u,v) \neq \emptyset$, then the mixing time of the Markov chain MC is bounded above by $\tau_M(\delta) = O(n^4(\ln \pi(M)^{-1} + \ln \delta^{-1}))$.

This theorem improves the mixing time bound by $O(n^2)$ over the corresponding result in [10]. The theorem will be proved in section 7.

4.3. Bootstrapping ideal weights. We will run the chain with weights w close to w^*, and then we can use samples from the stationary distribution to redefine w so that they are arbitrarily close to w^*. For the Markov chain run with weights w, note that

$$\pi(\mathcal{N}(u,v)) = \frac{w(u,v)\lambda(\mathcal{N}(u,v))}{Z} = \frac{w(u,v)\lambda(\mathcal{P})}{Zw^*(u,v)} = \pi(\mathcal{P})\frac{w(u,v)}{w^*(u,v)}.$$

Rearranging, we have

$$w^*(u,v) = \frac{\pi(\mathcal{P})}{\pi(\mathcal{N}(u,v))} w(u,v). \tag{4.3}$$

Given weights w which are a rough approximation to w^*, identity (4.3) implies an easy method to recalibrate weights w to an arbitrarily close approximation to w^*. We generate many samples from the stationary distribution and observe the number of perfect matchings in our samples versus the number of near-perfect matchings with holes u, v. By generating sufficiently many samples, we can estimate $\pi(\mathcal{P})/\pi(\mathcal{N}(u,v))$ within an arbitrarily close factor, and hence we can estimate $w^*(u,v)$ (via (4.3)) within an arbitrarily close factor.

More precisely, recall that for $w = w^*$, the stationary distribution of the chain satisfies $\pi(\mathcal{N}(u,v)) = 1/(n^2+1)$. For weights w that are within a factor of 2 of the ideal weights w^*, it follows that $\pi(\mathcal{N}(u,v)) \geq 1/4(n^2+1)$. Then, by Chernoff bounds, $S = O(n^2 \log(1/\hat{\eta}))$ samples of the stationary distribution of the chain suffice to approximate $\pi(\mathcal{P})/\pi(\mathcal{N}(u,v))$ within a factor $\sqrt{2}$ with probability $\geq 1-\hat{\eta}$. Thus, by (4.3) we can also approximate w^* within a factor $\sqrt{2}$ with the same bounds.

Theorem 4.1 (with $\delta = \Theta(1/n^2)$) implies that $T = O(n^4 \log n)$ time is needed to generate each sample (we will choose $\hat{\eta}$ so that the failure probability of the entire algorithm is small; e.g., $\hat{\eta} = \Theta(1/n^4)$ suffices). To be precise, this requires the use of "warm start" samples in which the initial matching for the Markov chain simulation is a reasonable approximation to the stationary distribution. In particular, after the initial sample from (close to) the stationary distribution, the initial matching for each simulation is the final matching from the previous simulation. (The application of warm starts in our work is identical to their use in [10]; hence we refer the interested reader to [10] for further details.)

4.4. Simulated annealing with new cooling schedule. In this section we present an $O^*(n^7)$ algorithm for estimating the ideal weights w^*. The algorithm will be used in section 4.5 to approximate the permanent of a 0/1 matrix. The algorithm can be generalized to compute the permanent of general nonnegative matrices; see section 9.

The algorithm runs in phases, each characterized by a parameter λ. In every phase,

$$\lambda(e) := \begin{cases} 1 & \text{for } e \in E, \\ \lambda & \text{for } e \notin E. \end{cases} \tag{4.4}$$

We start with $\lambda = 1$ and slowly decrease λ until it reaches its target value $1/n!$.

At the start of each phase we have a set of weights within a factor 2 of the ideal weights for all u, v, with high probability. Applying Theorem 4.1 we generate many samples from the stationary distribution. Using these samples and (4.3), we refine the weights to within a factor $\sqrt{2}$ of the ideal weights:

$$\frac{w^*(u,v)}{\sqrt{2}} \leq w(u,v) \leq \sqrt{2}w^*(u,v). \tag{4.5}$$

This allows us to decrease λ so that the current estimates of the ideal weights for λ_i are within a factor of 2 of the ideal weights for λ_{i+1}.

In [10], $O(n^2 \log n)$ phases are required. A straightforward way to achieve this is to decrease λ by a factor of $2^{-1/n}$ between phases as considered in section 3.1 for colorings.

We use only $\ell = O(n \log^2 n)$ phases by progressively decreasing λ by a larger amount per phase. Initially we decrease λ by a factor of $2^{-1/n}$ per phase, but during the final phases we decrease λ by a constant factor per phase.

Here is the pseudocode of our algorithm. The algorithm outputs w, which is a 2-approximation of the ideal weights w^* with probability $\geq 1-\eta$. Recall from the last paragraphs of the previous section that $S = O(n^2(\log n + \log \eta^{-1}))$ since $\eta = O(\ell\hat{\eta})$, and $T = O(n^4 \log n)$.

Algorithm for approximating ideal weights of 0/1 matrices:

```
  Initialize $\lambda = 1$ and $i = n$.
  Initialize $w(u,v) \leftarrow n$ for all $(u,v) \in V_1 \times V_2$.
  While $\lambda > 1/n!$ do:
      Take $S$ samples from $MC$ with parameters $\lambda, w$, using a warm start simulation
```

```
            (in particular, initial matchings for the simulation are the final matchings
            from the previous simulation). We use $T$ steps of the $MC$ per sample,
            except for the first sample which needs $O(Tn \log n)$ steps.
        Use the samples to obtain estimates $w'(u, v)$ satisfying condition (4.5)
            for all $u, v$. The algorithm fails (i.e., (4.5) is not satisfied) with small
            probability.
        Set $\lambda = 2^{-1/(2i)}\lambda$.
        If $i > 2$ and $\lambda < (n-1)!^{-1/(i-1)}$,
            Set $\lambda = (n-1)!^{-1/(i-1)}$ and decrement $i = i - 1$.
        If $\lambda < 1/n!$, set $\lambda = 1/n!$.
        Set $w(u, v) = w'(u, v)$ for all $u \in V_1$, $v \in V_2$.
    Output the final weights $w(u, v)$.
```

By Lemma 3.1, the above algorithm consists of $O(n \log^2 n)$ phases. This follows from setting $s = n$, $c = \sqrt{2}$, $\gamma = n!$, and $D = 1$ (the choice of D becomes clear in section 8). In section 8 we show that Lemma 3.2 implies that our weights at the start of each phase satisfy (4.2) assuming that the estimates w' satisfied condition (4.5) throughout the execution of the algorithm. Therefore, the total running time is $O(STn \log^2 n) = O(n^7 \log^4 n)$.

4.5. Reduction from counting to sampling. Let $\lambda_0 = 1 > \lambda_1 > \cdots > \lambda_\ell = 1/n!$, $\ell = O(n \log^2 n)$, be the sequence of λ used in the weight-estimating algorithm from the previous section. Assume that the algorithm did not fail, i.e., the hole weights $w_0, \dots, w_\ell$ computed by the algorithm are within a factor of $\sqrt{2}$ from the ideal weights $w_0^*, \dots, w_\ell^*$.

It remains to use these (constant factor) estimates of the ideal weights to obtain a $(1 \pm \varepsilon)$-approximation of the permanent. This is done by expressing the permanent as a telescoping product as was done for colorings in section 3.1. We refer the interested reader to section 5 from [10] for details of the argument. The only difference from [10] is that the number of intermediate temperatures is $\ell = O(n \log^2 n)$ as opposed to $O(n^2 \log n)$. The total running time of this part of the algorithm is $O(\ell^2/\varepsilon^2 n^4 \log n) = O(n^6 \log^5 n \varepsilon^{-2})$. This completes the description of the algorithm for 0/1 matrices.

5. Canonical paths for proving Theorem 4.1. Recall the canonical paths method from section 2.2. We will use this approach with $S = \mathcal{P}$. To prove Theorem 4.1 we need to define canonical paths $\gamma(I, F)$ for all initial $I \in \Omega$ and final $F \in \mathcal{P}$. These paths will have length $\ell \leq n$, and hence we need to show that the congestion satisfies $\rho(T) = O(n)$ for every transition T. The canonical paths we use are identical to those considered in [10] (and in the earlier work of [9]).

We define the canonical paths now and defer the bound on the congestion to section 7, after presenting some combinatorial lemmas in section 6. We will assume that the vertices of G are numbered. If $I \in \mathcal{P}$, then $I \oplus F$ consists of even length cycles, where $\oplus$ denotes the symmetric difference. Let us assume that the cycles are numbered according to the smallest numbered vertex contained in them. The path $\gamma(I, F)$ "corrects" these cycles in order. Let $v_0, v_1, \dots, v_{2k-1}$ be a cycle C, where v_0 is the smallest numbered vertex in C and $(v_0, v_1) \in I$. The path starts by unmatching edge (v_0, v_1) and successively interchanging edge (v_{2i}, v_{2i+1}) for edge (v_{2i-1}, v_{2i}) for $1 \leq i \leq k-1$. Finally, it adds edge (v_{2k-1}, v_0) to the matching.

If $I \in \mathcal{N}(w, z)$, then there is an augmenting path from w to z in $I \oplus F$. The canonical path starts by augmenting I along this path by first exchanging edges and finally adding the last edge. It then "corrects" the even cycles in order. Figure 5.1 shows an intermediate transition on the canonical path from I to F.

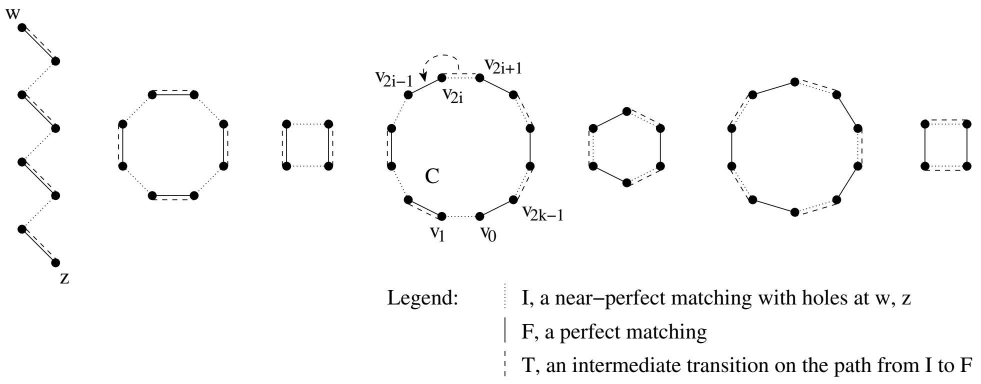

w
z
v2i−1
v2i
v2i+1
C
v1
v0
v2k−1
Legend:
I, a near−perfect matching with holes at w, z
F, a perfect matching
T, an intermediate transition on the path from I to F

FIG. 5.1. *This figure illustrates a near-perfect matching I and a perfect matching F, along with a transition T on the canonical path from I to F. The transition is "sliding" an edge in the cycle denoted as C. The components of $I \oplus F$ are shown in increasing order (from left to right). The alternating path and the cycles to the left of C have been already "corrected," whereas the cycles on the right still need to be "corrected." The unfinished cycle is partially corrected: From v_0 to v_{2i-1} the cycle is the same as F, whereas from v_{2i+1} to v_{2k-1} the cycle is the same as I. (A similar picture arises on the canonical path from a perfect matching to a perfect matching, except in that case there is no alternating path.)*

This completes the definition of the canonical paths, and it remains to bound their associated congestion. To this end, in the following section we present several inequalities which are used to improve the analysis of the congestion.

6. Key technical lemmas. The following lemma contains the new combinatorial inequalities which are the key to our improvement of $O(n^2)$ in Theorem 4.1. These inequalities will be used to bound the total weight of (I, F) pairs whose canonical path passes through a specified transition. In [10] weaker inequalities were proved without the sum in the left-hand side and were a factor of 2 smaller in the right-hand side. The proof of Lemma 6.2 below improves on Lemma 7 in [10] by constructing more efficient mappings. We first present our mappings in the simpler setting of Lemma 6.1 and later use them to prove Lemma 6.2. Using these new inequalities to bound the congestion requires more work than the analysis of [10].

LEMMA 6.1. *Let $u, w \in V_1$, $v, z \in V_2$ be distinct vertices. Then,*

1.
$$\sum_{x,y:(u,y),(x,v)\in E} |\mathcal{N}(u,v)||\mathcal{N}(x,y)| \le 2|\mathcal{P}|^2,$$

2.
$$\sum_{x:(x,v)\in E} |\mathcal{N}(u,v)||\mathcal{N}(x,z)| \le 2|\mathcal{N}(u,z)||\mathcal{P}|,$$

3.
$$\sum_{x,y:(u,y),(x,v)\in E} |\mathcal{N}(u,v)||\mathcal{N}(x,y,w,z)| \le 2|\mathcal{N}(w,z)||\mathcal{P}|.$$

The basic intuition for the proofs of these inequalities is straightforward. For example, consider the first inequality. Take matchings $M \in \mathcal{N}(u,v)$, $M' \in \mathcal{N}(x,y)$.

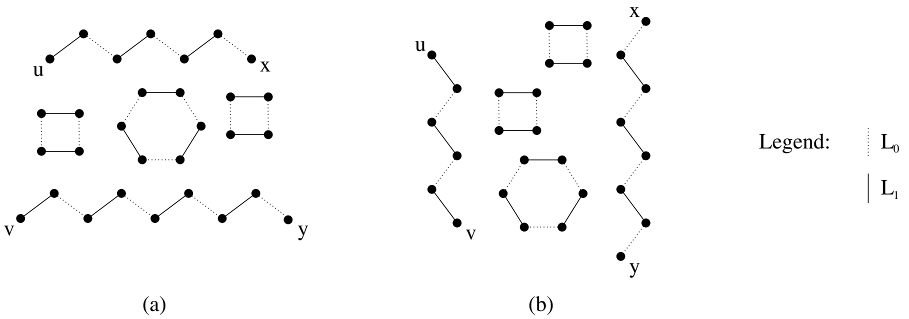


FIG. 6.1. *Proof of Lemma* 6.1, *part* 1. *Two different possibilities for* $L_0 \oplus L_1$.

The set $M \cup M' \cup (u, y) \cup (x, v)$ consists of a set of alternating cycles. Hence, this set can be broken into a pair of perfect matchings. One of the perfect matchings contains the edge (u, y), and one matching contains the edge (x, v). Hence, given the pair of perfect matchings, we can deduce the original unmatched vertices (by guessing which of the two edges is incident to u) and thereby reconstruct M and M'. This outlines the approach for proving Lemma 6.1.

Proof. 1. We will construct a one-to-one map:

$$f_1 : \mathcal{N}(u, v) \times \bigcup_{x,y:(u,y),(x,v)\in E} \mathcal{N}(x, y) \to \mathcal{P} \times \mathcal{P} \times b,$$

where b is a bit, i.e., b is 0/1.

Let $L_0 \in \mathcal{N}(u, v)$ and $L_1 \in \cup_{x,y:(u,y),(x,v)\in E}\mathcal{N}(x, y)$. In $L_0 \oplus L_1$ the four vertices u, v, x, y each have degree one, and the remaining vertices have degree zero or two. Hence these four vertices are connected by two disjoint paths. Now there are three possibilities:

- If the paths are u to x and v to y, they must both be even length, as seen in Figure 6.1(a).
- If the paths are u to v and x to y, they must both be odd length, as seen in Figure 6.1(b).
- The third possibility, u to y and v to x, is ruled out since such paths start with an L_0 edge and end with an L_1 edge and hence must be even length; on the other hand, they connect vertices across the bipartition and hence must be of odd length.

Now, the edges (u, y) and (v, x) are in neither matching, and so $(L_0 \oplus L_1) \cup \{(u, y), (v, x)\}$ contains an even cycle, say, C, containing (u, y) and (x, v). We will partition the edges of $L_0 \cup L_1 \cup \{(u, y), (v, x)\}$ into two perfect matchings as follows. Let M_0 contain the edges of L_0 outside C and alternate edges of C starting with edge (u, y). M_1 will contain the remaining edges. Bit b is set to 0 if $(x, v) \in M_0$ and to 1 otherwise. This defines the map f_1.

Next, we show that f_1 is one-to-one. Let M_0 and M_1 be two perfect matchings and b be a bit. If u and v are not in one cycle in $M_0 \oplus M_1$, then (M_0, M_1, b) is not mapped onto by f_1. Otherwise, let C be the common cycle containing u and v. Let y be the vertex matched to u in M_0. If $b = 0$, denote by x the vertex that is matched to v in M_0; else denote by x the vertex that is matched to v in M_1. Let L_0 contain the edges of M_0 outside C, and let it contain the near-perfect matching in C that leaves u and v unmatched. Let L_1 contain the edges of M_1 outside C, and let it contain the near-perfect matching in C that leaves x and y unmatched. It is easy to see that $f_1(L_0, L_1) = (M_0, M_1, b)$.

2. We will construct a one-to-one map:

$$f_2 : \mathcal{N}(u,v) \times \bigcup_{x:(x,v)\in E} \mathcal{N}(x,z) \to \mathcal{N}(u,z) \times \mathcal{P} \times b.$$

Let $L_0 \in \mathcal{N}(u,v)$ and $L_1 \in \cup_{x:(x,v)\in E}\mathcal{N}(x,z)$. As before, u, v, x, z are connected by two disjoint paths of the same parity in $L_0 \oplus L_1$ and $(v,x) \notin L_0 \cup L_1$. Hence, $L_0 \cup L_1 \cup \{(x,v)\}$ contains an odd length path from u to z, say, P. Construct $M_0 \in \mathcal{N}(u,z)$ by including all edges of L_0 not on P and alternate edges of P, leaving u, z unmatched. Let $M_1 \in \mathcal{P}$ consist of the remaining edges of $L_0 \cup L_1 \cup \{(x,v)\}$. Let $b = 0$ if $(v,x) \in M_0$, and $b = 1$ otherwise. Clearly, path P appears in $M_0 \oplus M_1$, and as before, L_0 and L_1 can be retrieved from (M_0, M_1, b).

3. We will construct a one-to-one map:

$$f_3 : \mathcal{N}(u,v) \times \bigcup_{x,y:(u,y),(x,v)\in E} \mathcal{N}(x,y,w,z) \to \mathcal{N}(w,z) \times \mathcal{P} \times b.$$

Let $L_0 \in \mathcal{N}(u,v)$ and $L_1 \in \cup_{x,y:(u,y),(x,v)\in E}\mathcal{N}(x,y,w,z)$. Consider $L_0 \oplus L_1$. There are two cases. If there are two paths connecting the four vertices u, v, x, y (and a separate path connecting w and z), then the mapping follows using the construction given in case 1. Otherwise, by parity considerations the only possibilities are

- u to w and v to y are even length paths and x to z is an odd length path;
- u to x and v to z are even length paths and w to y is an odd length path;
- u to w and v to z are even length paths and x to y is an odd length path; and
- u to v, x to z, and w to y are odd length paths.

Now, $L_0 \cup L_1 \cup \{(u,y),(v,x)\}$ contains an odd length path, say, P, from w to z. Now, the mapping follows using the construction given in case 2. □

The following lemma is an extension of the previous lemma, which served as a warm-up. This lemma is used to bound the congestion.

LEMMA 6.2. *Let $u, w \in V_1$, $v, z \in V_2$ be distinct vertices. Then,*

1.

$$\sum_{x\in V_1, y\in V_2} \lambda(u,y)\lambda(x,v)\lambda(\mathcal{N}(u,v))\lambda(\mathcal{N}(x,y)) \le 2\lambda(\mathcal{P})^2,$$

2.

$$\sum_{x\in V_1} \lambda(x,v)\lambda(\mathcal{N}(u,v))\lambda(\mathcal{N}(x,z)) \le 2\lambda(\mathcal{N}(u,z))\lambda(\mathcal{P}),$$

3.

$$\sum_{x\in V_1, y\in V_2} \lambda(u,y)\lambda(x,v)\lambda(\mathcal{N}(u,v))\lambda(\mathcal{N}(x,y,w,z)) \le 2\lambda(\mathcal{N}(w,z))\lambda(\mathcal{P}).$$

Proof. We will use the mappings f_1, f_2, f_3 constructed in the proof of Lemma 6.1. Observe that since mapping f_1 constructs matchings M_0 and M_1 using precisely the edges of L_0, L_1 and the edges $(u,y), (x,v)$, it satisfies

$$\lambda(u,y)\lambda(x,v)\lambda(L_0)\lambda(L_1) = \lambda(M_0)\lambda(M_1).$$

Summing over all pairs of matchings in

$$\mathcal{N}(u,v) \times \bigcup_{x,y:(u,y),(x,v)\in E} \mathcal{N}(x,y),$$

we get the first inequality. The other two inequalities follow in a similar way using mappings f_2 and f_3. □

7. Bounding congestion: Proof of Theorem 4.1.

We bound the congestion separately for transitions which move between near-perfect matchings (cases 2(b) and 2(c) in the definition of chain MC in section 4.2) and transitions which move between a perfect and near-perfect matching (case 1). Our goal for this section will be to prove for every transition $T = M \to M'$,

$$\sum_{\substack{(I,F)\in\Omega\times\mathcal{P}:\\ T\in\gamma(I,F)}} \frac{w^*(I)w^*(F)}{w^*(M)} = O(w^*(\Omega)). \tag{7.1}$$

At the end of this section we will prove that this easily implies the desired bound on the congestion.

For a transition $T = M \to M'$, we need to bound the number of canonical paths passing through T. We partition these paths into $2n^2+1$ sets,

$$cp_T = \{(I,F)\in\mathcal{P}\times\mathcal{P} : \gamma(I,F)\ni T\},$$

and, for all w, z,

$$cp_T^{w,z} = \{(I,F)\in\mathcal{N}(w,z)\times\mathcal{P} : \gamma(I,F)\ni T\}.$$

The following lemma converts into a more manageable form the weighted sum of I, F pairs which contain a transition of the first type.

Lemma 7.1. *Let $T = M \to M'$ be a transition which moves between near-perfect matchings (i.e., case* 2(b) *or* 2(c)*). Let $M\in\mathcal{N}(u,v)$, $M'\in\mathcal{N}(u,v')$, $u\in V_1$, $v,v'\in V_2$, and $M' = M\setminus(x,v')\cup(x,v)$ for some $x\in V_1$. Then, the following hold*:

1.

$$\sum_{(I,F)\in cp_T} \lambda(I)\lambda(F) \le \sum_{y\in V_2} \lambda(\mathcal{N}(x,y))\lambda(u,y)\lambda(x,v)\lambda(M).$$

2. *For all $z\in V_2$,*

$$\sum_{(I,F)\in cp_T^{u,z}} \lambda(I)\lambda(F) \le \lambda(\mathcal{N}(x,z))\lambda(x,v)\lambda(M).$$

3. *For all $w\in V_1$, $w\ne u$, and $z\in V_2$, $z\ne v,v'$,*

$$\sum_{(I,F)\in cp_T^{w,z}} \lambda(I)\lambda(F) \le \sum_{y\in V_2} \lambda(\mathcal{N}(w,z,x,y))\lambda(u,y)\lambda(x,v)\lambda(M).$$

Proof. 1. We will first construct a one-to-one map:

$$\eta_T : cp_T \to \bigcup_{x,y:(u,y),(x,v)\in E} \mathcal{N}(x,y).$$

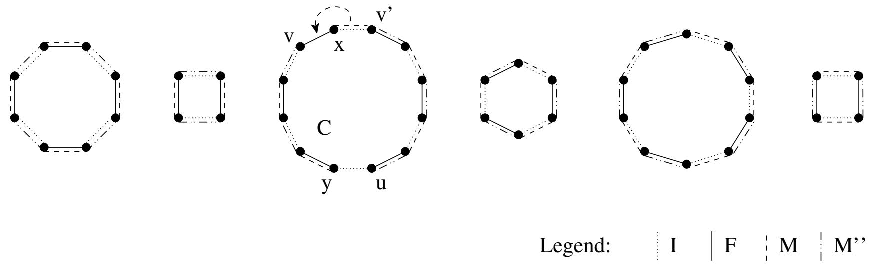


FIG. 7.1. *Proof of Lemma* 7.1, *part* 1.

Let $I, F \in \mathcal{P}$ and $(I, F) \in cp_T$. Let S be the set of cycles in $I \oplus F$. Order the cycles in S using the convention given in section 5. Clearly, u, v, x lie on a common cycle, say, $C \in S$, in $I \oplus F$. Since T lies on the canonical path from I to F, M has already corrected cycles before C and not yet corrected cycles after C in S. Let y be a neighbor of u on C. Define $M'' \in \mathcal{N}(x, y)$ to be the near-perfect matching that picks edges as follows: outside C, it picks edges $(I \cup F) - M$, and on C it picks the near-perfect matching leaving x, y unmatched. Figure 7.1 shows the definition of M''. Define $\eta_T(I, F) = M''$.

Clearly, $(M \oplus M'') \cup \{(u, v), (x, y)\}$ consists of the cycles in S, and I and F can be retrieved from M, M'' by considering the order defined on S. This proves that the map constructed is one-to-one. Since the union of edges in I and F equals the edges in $M \cup M'' \cup \{(u, v), (x, y)\}$,

$$\lambda(I)\lambda(F) = \lambda(M)\lambda(M'')\lambda(u, y)\lambda(x, v).$$

Summing over all $(I, F) \in cp_T$, we get the first inequality.

2. For all $z \in V_2$, we will again construct a one-to-one map:

$$\eta_T^{u,z} : cp_T^{u,z} \to \mathcal{N}(x, z).$$

Let $I \in \mathcal{N}(u, z)$, $F \in \mathcal{P}$, and $(I, F) \in cp_T^{u,z}$. Let S be the set of cycles and P be the augmenting path from u to z in $I \oplus F$. Clearly, x and v lie on P. M has "corrected" part of the path P and none of the cycles in S. It contains the edges of I from z to v and the edges of F from x to u. Also, it contains the edges of I from the cycles in S as well as the edges in $I \cap F$.

Construct matching $M'' \in \mathcal{N}(x, z)$ as follows. It contains the edges of F from the cycles in S, the edges $I \cap F$, and $(P - \{(x, v)\} - M)$. Define $\eta_T^{u,z}(I, F) = M''$. It is easy to see that $M \cup M'' = I \cup F \cup \{(x, v)\}$. Therefore,

$$\lambda(I)\lambda(F) = \lambda(M)\lambda(M'')\lambda(x, v).$$

Furthermore, I, F can be retrieved from M, M''. Hence, summing over all $(I, F) \in cp_T^{u,z}$, we get the second inequality.

3. For all $w \in V_1$, $w \neq u$, and $z \in V_2$, $z \neq v, v'$, we will construct a one-to-one map:

$$\eta_T^{w,z} : cp_T^{w,z} \to \bigcup_{y: u \sim y} \mathcal{N}(w, z, x, y).$$

Let $I \in \mathcal{N}(w, z)$, $F \in \mathcal{P}$, and $(I, F) \in cp_T^{w,z}$. Let S be the set of cycles and P be the augmenting path from w to z in $I \oplus F$. Clearly, u, v, x lie on a common cycle, say,

$C \in S$, in $I \oplus F$. Therefore, M has already "corrected" P, and so it looks like F on P. Construct $M'' \in \mathcal{N}(w,z,x,y)$ as follows. On P, it looks like I. Outside $P \cup C$, it picks edges $(I \cup F) - M$, and on C it picks the near-perfect matching leaving x, y unmatched. Define $\eta_T^{w,z}(I,F) = M''$. It is easy to see that $M \cup M'' = I \cup F \cup \{(u,y),(x,v)\}$. Therefore,

$$\lambda(I)\lambda(F) = \lambda(M)\lambda(M'')\lambda(u,y)\lambda(x,v).$$

Furthermore, I, F can be retrieved from M, M''. Hence, summing over all $(I,F) \in cp_T^{w,z}$, we get the third inequality. □

We now prove (7.1) for the first type of transitions. The proof applies Lemma 7.1 and then Lemma 6.2. We break the statement of (7.1) into two cases depending on whether I is a perfect matching or a near-perfect matching.

LEMMA 7.2. *For a transition $T = M \to M'$ which moves between near-perfect matchings (i.e., case 2(b) or 2(c)), the congestion from $(I,F) \in \mathcal{P} \times \mathcal{P}$ is bounded as*

$$\sum_{(I,F)\in cp_T} \frac{w^*(I)w^*(F)}{w^*(M)} \le \frac{2w^*(\Omega)}{n^2}. \tag{7.2}$$

And, the congestion from $(I,F) \in \mathcal{N}_2 \times \mathcal{P}$ is bounded as

$$\sum_{w\in V_1, z\in V_2} \sum_{(I,F)\in cp_T^{w,z}} \frac{w^*(I)w^*(F)}{w^*(M)} \le 3w^*(\Omega). \tag{7.3}$$

Proof. The transition T is sliding an edge; let x denote the pivot vertex, and let $M \in \mathcal{N}(u,v)$ and $M' \in \mathcal{N}(u,v')$, where $u \in V_1$, $v, v' \in V_2$. Thus, $M' = M \setminus (v',x) \cup (x,v)$ for some $x \in V_1$.

We begin with the proof of (7.2):

$$\begin{aligned}
&\sum_{(I,F)\in cp_T} \frac{w^*(I)w^*(F)}{w^*(M)} \\
&= \sum_{(I,F)\in cp_T} \lambda(I)\lambda(F)\frac{\lambda(\mathcal{N}(u,v))}{\lambda(M)\lambda(\mathcal{P})} \\
&\le \sum_{y\in V_2} \frac{\lambda(\mathcal{N}(x,y))\lambda(u,y)\lambda(x,v)\lambda(\mathcal{N}(u,v))}{\lambda(\mathcal{P})} \qquad \text{(by Lemma 7.1)} \\
&\le 2\lambda(\mathcal{P}) \qquad \text{(by Lemma 6.2)} \\
&= \frac{2w^*(\Omega)}{n^2+1}.
\end{aligned}$$

Note that the application of Lemma 6.2 uses only the summation over y and does not require the summation over x. We have now completed the proof of (7.2). We now prove (7.3) in two parts. This first bound covers the congestion due to the first part of the canonical paths from a near-perfect matching to a perfect matching, unwinding the augmenting path. The second bound covers the second part of these canonical paths when we unwind the alternating cycle(s). During the unwinding of the augmenting path, one of the holes of the transition is the same as one of the holes of the initial near-perfect matching. This is what characterizes the first versus the second part of the canonical path.

$$\begin{aligned}
&\sum_{z\in V_2}\sum_{(I,F)\in cp_T^{u,z}}\frac{w^*(I)w^*(F)}{w^*(M)}\\
&\quad=\sum_{z\in V_2}\sum_{(I,F)\in cp_T^{u,z}}\lambda(I)\lambda(F)\frac{\lambda(\mathcal{N}(u,v))}{\lambda(M)\lambda(\mathcal{N}(u,z))}\\
&\quad\le\sum_{z\in V_2}\frac{\lambda(\mathcal{N}(x,z))\lambda(x,v)\lambda(\mathcal{N}(u,v))}{\lambda(\mathcal{N}(u,z))}\qquad\text{(by Lemma 7.1)}\\
&\quad\le\sum_{z\in V_2}2\lambda(\mathcal{P})\qquad\text{(by Lemma 6.2)}\\
&\quad=\frac{2n}{n^2+1}w^*(\Omega)\\
&\quad\le w^*(\Omega).
\end{aligned}$$

Finally, bounding the congestion from the unwinding of the alternating cycle(s) on canonical paths from near-perfect matchings to perfect matchings,

$$\begin{aligned}
&\sum_{\substack{w\in V_1,z\in V_2:\\ w\ne u}}\sum_{(I,F)\in cp_T^{w,z}}\frac{w^*(I)w^*(F)}{w^*(M)}\\
&=\sum_{\substack{w\in V_1,z\in V_2:\\ w\ne u}}\sum_{(I,F)\in cp_T^{w,z}}\lambda(I)\lambda(F)\frac{\lambda(\mathcal{N}(u,v))}{\lambda(M)\lambda(\mathcal{N}(w,z))}\\
&\le\sum_{\substack{w\in V_1,z\in V_2:\\ w\ne u}}\sum_{y\in V_2}\frac{\lambda(\mathcal{N}(w,z,x,y))\lambda(u,y)\lambda(x,v)\lambda(\mathcal{N}(u,v))}{\lambda(\mathcal{N}(w,z))}\qquad\text{(by Lemma 7.1)}\\
&\le\sum_{\substack{w\in V_1,z\in V_2:\\ w\ne u}}2\lambda(\mathcal{P})\qquad\text{(by Lemma 6.2)}\\
&\le 2w^*(\Omega).\qquad\square
\end{aligned}$$

We now follow the same approach as Lemmas 7.1 and 7.2 to prove (7.1) for transitions moving between a perfect and a near-perfect matching. The proofs in this case are easier.

Lemma 7.3. *For a transition $T=M\to M'$ which removes an edge (i.e., case* 1*) or adds an edge (i.e., case* 2(a)*), let (u,v) be the removed/added edge, and let N be the near-perfect matching from the pair M,M' (i.e., if adding an edge $N=M$, and if removing an edge $N=M'$). Then,*

$$\sum_{(I,F)\in cp_T}\lambda(I)\lambda(F)\le\lambda(\mathcal{P})\lambda(u,v)\lambda(N).$$

And, for all $w\in V_1$, $z\in V_2$,

$$\sum_{(I,F)\in cp_T^{w,z}}\lambda(I)\lambda(F)\le\lambda(\mathcal{N}(w,z))\lambda(u,v)\lambda(N).$$

Proof. Let P denote the perfect matching from the pair M,M'. Define $\eta=\eta_T^{w,z}$: $cp_T^{w,z}\to\mathcal{N}(w,z)$ as

$$\eta(I,F)=I\cup F\setminus P.$$

The mapping satisfies $\lambda(I)\lambda(F) = \lambda(P)\lambda(\eta(I,F))$. Note that $\lambda(P) = \lambda(N)\lambda(u,v)$. Since the mapping is one-to-one, summing over all $N' \in \mathcal{N}(w,z)$ proves the lemma for all w, z. The proof is identical for cp_T with the observation that when $I \in \mathcal{P}$, we have that $I \cup F \setminus P$ is in $\mathcal{P}$. □

Lemma 7.4. *For a transition $T = M \to M'$ which adds or subtracts an edge (i.e., case 1 or 2(a)), the congestion from $(I,F) \in \Omega \times \mathcal{P}$ is bounded as*

$$\sum_{w,z} \sum_{(I,F)\in cp_T^{w,z}} \frac{w^*(I)w^*(F)}{w^*(M)} \le w^*(\Omega)$$

and

$$\sum_{(I,F)\in cp_T} \frac{w^*(I)w^*(F)}{w^*(M)} \le \frac{w^*(\Omega)}{n^2}.$$

Proof. Let $M \in \mathcal{N}(u,v)$ and $M' \in \mathcal{P}$; thus the transition adds the edge (u,v). The proof for the transition which subtracts the edge will be analogous. The proof is a simplified version of Lemma 7.2, using Lemma 7.3.

Observe that for any x, y,

$$\lambda(x,y)\lambda(\mathcal{N}(x,y)) \le \lambda(\mathcal{P}). \tag{7.4}$$

We begin with the proof of the first inequality in Lemma 7.4.

$$\begin{aligned}\sum_{w,z} \sum_{(I,F)\in cp_T^{w,z}} \frac{w^*(I)w^*(F)}{w^*(M)} &= \sum_{w,z} \sum_{(I,F)\in cp_T^{w,z}} \lambda(I)\lambda(F)\frac{\lambda(\mathcal{N}(u,v))}{\lambda(M)\lambda(\mathcal{N}(w,z))}\\ &\le \sum_{w,z} \lambda(u,v)\lambda(\mathcal{N}(u,v)) \qquad \text{(by Lemma 7.3)}\\ &\le w^*(\Omega) \qquad \text{(by (7.4))}.\end{aligned}$$

We now prove the second inequality in Lemma 7.4.

$$\begin{aligned}\sum_{(I,F)\in cp_T} \frac{w^*(I)w^*(F)}{w^*(M)} &= \sum_{(I,F)\in cp_T} \lambda(I)\lambda(F)\frac{\lambda(\mathcal{N}(u,v))}{\lambda(M)\lambda(\mathcal{P})}\\ &\le 2\lambda(u,v)\lambda(\mathcal{N}(u,v)) \qquad \text{(by Lemma 7.3)}\\ &\le \lambda(\mathcal{P}) \qquad \text{(by (7.4))}. \qquad \square\end{aligned}$$

We now restate Theorem 4.1 and then present its proof.

Theorem 4.1. *Assuming the weight function w satisfies inequality*

$$w^*(u,v)/2 \le w(u,v) \le 2w^*(u,v) \tag{4.2}$$

for every $(u,v) \in V_1 \times V_2$ with $\mathcal{N}(u,v) \ne \emptyset$, then the mixing time of the Markov chain MC is bounded above by $\tau_M(\delta) = O(n^4(\ln \pi(M)^{-1} + \ln \delta^{-1}))$.

Proof of Theorem 4.1. Inequality (4.2) implies for any set of matchings $S \subset \Omega$ that the stationary distribution $\pi(S)$ under w is within a factor of 4 of the distribution under w^*. Therefore, to prove Theorem 4.1 it suffices to consider the stationary distribution with respect to w^*. In other words, we need to prove, for every transition T, $\rho(T) = O(n)$ where, for $M \in \Omega$, $\pi(M) = w^*(M)/w^*(\Omega)$. Then for weights satisfying (4.2) the congestion increases by at most a constant factor. Thus, we need to prove

$$\sum_{\substack{(I,F)\in\Omega\times\mathcal{P}:\\ T\in\gamma(I,F)}} \frac{w^*(I)w^*(F)}{w^*(M)P(M,M')} = O(nw^*(\Omega)).$$

Recall case 3 in the definition of the Markov chain MC (section 4.2), where the Metropolis filter is applied.

In particular, from M_t, a new matching N is proposed with probability $1/4n$, and then the proposed new matching is accepted with probability $\min\{1, w^*(N)/w^*(M_t)\}$. Hence, for the transition $T = M \to M'$,

$$w^*(M)P(M,M') = \frac{1}{4n}\min\{w^*(M), w^*(M')\}.$$

The chain is reversible; thus for every transition $T = M \to M'$, there is a reverse transition $T' = M' \to M$. Hence, to prove Theorem 4.1, it suffices to prove that for every transition $T = M \to M'$,

$$\sum_{\substack{(I,F)\in\Omega\times\mathcal{P}:\\ T\in\gamma(I,F)}} \frac{w^*(I)w^*(F)}{w^*(M)} = O(w^*(\Omega)). \tag{7.5}$$

Lemmas 7.2 and 7.4 imply (7.5) which completes the proof of the theorem. □

8. Phases in the permanent algorithm. In this section we show that the choice of λ from the weight-estimating algorithm ensures that (4.2) is satisfied in each phase. Recall that we can obtain a refined estimate of the ideal weights in each phase; see (4.5). We need to guarantee that the weights of two consecutive phases do not differ too much. Namely, if they are within a $\sqrt{2}$ factor of each other, together with (4.5) we have (4.2) for the next phase. As we will see shortly, for our choice of activities the ideal weights $w^*(u,v)$ are a ratio of two polynomials of degree $\leq n$ evaluated at λ. This observation will allow us to use Lemma 3.2.

DEFINITION 8.1. *We say that a matching $M \in \mathcal{P}$ of a complete bipartite graph* covers *k edges of a graph G if the size of $M \cap E(G)$ is k. Let*

$$R_G(x) = \sum_{k=0}^{n} p_k x^{n-k},$$

where p_k is the number of matchings in $\mathcal{P}$ covering k edges of G.

Note that the ideal weights w^*, defined by (4.1), for activities given by (4.4) can be expressed as

$$w^*_\lambda(u,v) = \frac{R_G(\lambda)}{R_{G\setminus\{u,v\}}(\lambda)}. \tag{8.1}$$

First we observe that every R-polynomial has a positive low-degree coefficient (and consequently in the application of Lemma 3.2 we will have that D is small). In particular, the coefficient of either x^0 or x^1 is positive in each of the polynomials R_G, $R_{G\setminus\{u,v\}}$ for every $u \in V_1$, $v \in V_2$. This follows from the fact that G contains a perfect matching. Let M be a perfect matching of G. The existence of M implies that the constant term in R_G is positive. Similarly, if $(u,v) \in M$, then the constant term in $R_{G\setminus\{u,v\}}$ is positive because $M \setminus \{(u,v)\}$ is a perfect matching in $G \setminus \{u,v\}$. If $(u,v) \notin M$, let u', respectively, v', be the vertices matched to u and v in M, and let $M' = M \cup \{(v',u')\} \setminus \{(u,u'),(v,v')\}$. Depending on (v',u') being an edge in G, the cardinality of M' is either $n-1$ or $n-2$. Therefore, the coefficient of either x^0 or x^1 in $R_{G\setminus\{u,v\}}$ is positive.

Now we are ready to apply Lemma 3.2. Let $c = \sqrt{2}$, $\gamma = n!$, $D = 1$, $s = n$, and $Z_1 = R_G$, and the polynomials $Z_2, \dots, Z_{n^2+1}$ are the $R_{G\setminus\{u,v\}}$ polynomials for $u \in V_1, v \in V_2$. Let $\lambda_0, \dots, \lambda_\ell$ be the sequence obtained from the algorithm in section 3.2. Notice that we obtain the same sequence in the algorithm for estimating weights of the permanent. Then

$$\begin{array}{ll} R_G(\lambda_k) \geq R_G(\lambda_{k+1}) \geq R_G(\lambda_k)/\sqrt{2}, & \text{and} \\ R_{G\setminus\{u,v\}}(\lambda_k) \geq R_{G\setminus\{u,v\}}(\lambda_{k+1}) \geq R_{G\setminus\{u,v\}}(\lambda_k)/\sqrt{2} & \text{for every } u,v. \end{array} \tag{8.2}$$

Equations (8.1) and (8.2) imply the $w^*_{\lambda_k}$ and $w^*_{\lambda_{k+1}}$ are within a $\sqrt{2}$ factor. Moreover, if the weight-estimating algorithm does not fail, i.e., the w_{λ_k} satisfy (4.5), then w_{λ_k} also satisfy (4.2), as required by Theorem 4.1.

9. Nonnegative matrices.

The extension to nonnegative matrices follows identically as in section 7 of [10]; hence we refer the interested reader to [10].

10. Discussion.

10.1. Recent improvements.

In this paper, we have presented a near-optimal cooling schedule subject to the constraint that each of the ratios Z_i/Z_{i-1} in (1.1) is bounded. However, in order to estimate the ratio efficiently, it suffices to have an unbiased estimator with bounded variance and the ratio itself might be large. A recent paper [18] presents a general cooling schedule achieving the bounded variance property. As a consequence, for many combinatorial problems, such as colorings or matchings, [18] achieves a cooling schedule of length $O^*(\sqrt{n})$, whereas in this paper we present a schedule of length $O^*(n)$. Therefore, the improved schedule of [18] reduces (compared with this paper) the overall running time by a factor of $O^*(n)$ for many combinatorial counting problems. For the permanent, the improved cooling schedule of [18] does not appear to apply, since the algorithm for approximating the permanent needs to consider multiple polynomials simultaneously.

10.2. Permanent application.

With the improvement in running time of the approximation algorithm for the permanent, computing permanents of $n \times n$ matrices with $n \approx 100$ now appears feasible. Further improvement in the running time is an important open problem.

Some avenues for improvement are the following. We expect that the mixing time of the underlying chain is better than $O(n^4)$. Some slack in the analysis is in the application of the new inequalities to bound the congestion. In their application we simply use a sum over y, whereas the inequalities hold for a sum over x and y as stated in Lemma 6.2.

Another direction is reducing the number of samples needed per phase. It is possible that fewer samples are needed at each intermediate activity for estimating the ideal weights w^*. Perhaps the w^* satisfy relations which allow for fewer samples to infer them.

Appendix. A lower bound of the number of *k*-colorings.

We will use $a^{\underline{b}}$ to denote $a!/(a-b)!$. Let G be a graph, and for each vertex i of G let L_i be a list of colors. A valid list-coloring of G is a coloring such that each i has a color from L_i, and no two neighbors have the same color.

LEMMA A.1. *Let G be a graph with n vertices. Let d_j be the number of vertices of degree j in G. Let s be an integer, $s \geq 1$. Let $L_1, \dots, L_n$ be sets such that $|L_i| \geq s + \deg i$ for each vertex i of G. Let Ω be the set of valid list-colorings of G. Then*

$$|\Omega| \geq \prod_{j=0}^{n} c_j^{d_j},$$

where $c_j = ((s+j)^{\underline{j+1}})^{1/(j+1)}$.

Proof. We will use induction on $d_0 + \cdots + d_k$. Let v be the vertex of minimum degree ℓ. We have $|L_v| \geq \ell + s$. Let $j_1, \dots, j_\ell$ be the degrees of the neighbors of v. Note that $j_i \geq \ell$ for $i = 1, \dots, \ell$.

By the induction hypothesis

(A.1)
$$|\Omega| \geq (\ell+s)\left(\prod_{j=0}^{n} c_j^{d_j}\right)\frac{1}{c_\ell}\left(\prod_{i=1}^{\ell}\frac{c_{j_i-1}}{c_{j_i}}\right) \geq (\ell+s)\left(\prod_{j=0}^{n} c_j^{d_j}\right)\frac{1}{c_\ell}\left(\frac{c_{\ell-1}}{c_\ell}\right)^{\ell} = \prod_{j=0}^{n} c_j^{d_j},$$

where in the second inequality we used the inequality $c_j/c_{j+1} \geq c_{j-1}/c_j$, which we prove next.

Let $T = (s+j)^{\underline{j+1}}$. We want to show

$$T^{2/(j+1)} \geq \left(\frac{T}{s+j}\right)^{1/j}(T(s+j+1))^{1/(j+2)}.$$

After raising both sides to $-j(j+1)(j+2)/2$ and multiplying by $T^{\binom{j+1}{2}+\binom{j+2}{2}}$, we obtain an equivalent inequality

$$T \leq \frac{(s+j)^{\binom{j+2}{2}}}{(s+j+1)^{\binom{j+1}{2}}}. \tag{A.2}$$

Using the inequality between the arithmetic and geometric means,

$$\left((s+j)^{\underline{j+1}}(s+j+1)^{\binom{j+1}{2}}\right)^{1/\binom{j+2}{2}} \leq s+j,$$

which implies (A.2). Therefore, $c_j^2 \geq c_{j+1}c_{j-1}$, and hence the induction step (A.1) is proved. □

For k-colorings we obtain the following result.

COROLLARY A.2. *Let G be a graph with n vertices and maximum degree Δ. Let $k > \Delta$. Let Ω be the set of valid k-colorings of G. Then*

$$|\Omega| \geq \left(k^{\underline{\Delta+1}}\right)^{n/(\Delta+1)} \geq \left(k - \Delta\left(1-\frac{1}{e}\right)\right)^n \geq \left(\frac{k}{e}\right)^n.$$

Proof. Let $s = k - \Delta$. The first inequality follows from Lemma A.1 with the $L_i = [k]$.

The second inequality is equivalent to

$$(s+\Delta)^{\underline{\Delta+1}} \geq (s+\Delta/e)^{\Delta+1}. \tag{A.3}$$

The inequality (A.3) is true for $\Delta = 0$, and hence from now on we assume $\Delta \geq 1$.

Let $f(s,\Delta) = \sum_{i=0}^{\Delta} \ln \frac{s+i}{s+\Delta/e}$. Inequality (A.3) is equivalent to $f(s,\Delta) \geq 0$.

Claim.

$$f(1,\Delta) > 0. \tag{A.4}$$

Proof of the claim. We need to show that (A.3) holds with strict inequality for $s = 1$. Let $n = \Delta + 1$. We want to show $n! > (1 + (n-1)/e)^n$. The inequality $n! > \sqrt{2\pi n}(n/e)^n$ implies that it is enough to show $2\pi n \geq ((n+e-1)/n)^{2n}$, which (using $1 + x \leq e^x$) is implied by $2\pi n \geq e^{2(e-1)}$. Hence we proved (A.3) for $s = 1$ and $n \geq 5$. For $n \leq 4$ and $s = 1$ the (strict version of) inequality (A.3) is easily verified by hand. □

Each term in the definition of f goes to zero as s goes to infinity. Hence we have

$$\lim_{s\to\infty} f(s,\Delta) = 0. \tag{A.5}$$

Note that

$$f'(s,\Delta) = \frac{\partial f}{\partial s}(s,\Delta) = \frac{1}{s+\Delta/e} \sum_{i=0}^{\Delta} \frac{\Delta/e - i}{s+i}.$$

From $\Delta(\Delta+1)/e < \Delta(\Delta+1)/2$ it follows that for every Δ there exists s_Δ such that

$$f'(s,\Delta) < 0 \text{ for all } s > s_\Delta. \tag{A.6}$$

Let $g(s,\Delta,y) = \sum_{i=0}^{\Delta} \frac{y-i}{s+i}$. We have $g(s,\Delta,y) = 0$ iff

$$y = y_\Delta(s) := \left(\sum_{i=0}^{\Delta} \frac{i}{s+i}\right) \Bigg/ \left(\sum_{i=0}^{\Delta} \frac{1}{s+i}\right).$$

We will show that $y_\Delta(s)$ is an increasing function of s. This will imply that the equation $\Delta/e = y_\Delta(s)$ has at most one solution for any fixed Δ. Note that $f'(s,\Delta) = g(s,\Delta,\Delta/e)$. Hence we will obtain that $f'(s,\Delta) = 0$ has at most one solution for any fixed Δ. This, together with (A.4), (A.5), and (A.6), implies $f(s,\Delta) \geq 0$.

It remains to show that

$$(\partial y_\Delta/\partial s)(s) > 0. \tag{A.7}$$

The sign of $(\partial y_\Delta/\partial s)(s)$ is the same as the sign of

$$h(s,\Delta) := \left(\sum_{i=0}^{\Delta} \frac{i}{s+i}\right)\left(\sum_{i=0}^{\Delta} \frac{1}{(s+i)^2}\right) - \left(\sum_{i=0}^{\Delta} \frac{1}{s+i}\right)\left(\sum_{i=0}^{\Delta} \frac{i}{(s+i)^2}\right).$$

For $\Delta = 0$ we have $h(s,\Delta) = 0$. To show (A.7) it is enough to show that for $\Delta \geq 1$ the following quantity is positive:

$$h'(s,\Delta) := h(s,\Delta) - h(s,\Delta-1) = \frac{1}{s+\Delta}\left(\sum_{i=0}^{\Delta} \frac{\Delta - i}{(s+i)^2}\right) + \frac{1}{(s+\Delta)^2}\left(\sum_{i=0}^{\Delta} \frac{i-\Delta}{s+i}\right).$$

For $\Delta = 0$ we have $(s+\Delta)^2 h'(s,\Delta) = 0$. To show $h'(s,\Delta) > 0$ for $\Delta \geq 1$ it is enough to show that for $\Delta \geq 1$ the following quantity is positive:

$$h''(s,\Delta) := (s+\Delta)^2 h'(s,\Delta) - (s+\Delta-1)^2 h'(s,\Delta-1) = \sum_{i=0}^{\Delta-1} \frac{2\Delta - 2i - 1}{(s+i)^2}.$$

We have that $h''(s,\Delta)$ is a sum of positive numbers and hence $h''(s,\Delta) > 0$ for $\Delta \geq 1$. This implies $h'(s,\Delta) > 0$ for $\Delta > 0$ and this in turn implies $h(s,\Delta) > 0$ for $\Delta \geq 1$. We just proved (A.7), which was all that remained to be proved. □

REFERENCES

[1] I. Bezáková, N. Bhatnagar, and E. Vigoda, *Sampling binary contingency tables with a greedy start*, in Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), ACM, New York, SIAM, Philadelphia, 2006, pp. 414–423.

[2] A. Z. Broder, *How hard is it to marry at random? (On the approximation of the permanent)*, in Proceedings of the 18th Annual ACM Symposium on Theory of Computing (STOC), ACM, New York, 1986, pp. 50–58. Erratum in Proceedings of the 20th Annual ACM Symposium on Theory of Computing, ACM, New York, 1988, p. 551.

[3] R. Bubley and M. Dyer, *Path coupling: A technique for proving rapid mixing in Markov chains*, in Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society, Los Alamitos, CA, 1997, pp. 223–231.

[4] P. Diaconis and D. Stroock, *Geometric bounds for eigenvalues of Markov chains*, Ann. Appl. Probab., 1 (1991), pp. 36–61.

[5] A. Frieze and E. Vigoda, *A survey on the use of Markov chains to randomly sample colorings*, in Combinatorics, Complexity and Chance, Grimmett and McDiarmid, eds., Oxford University Press, Oxford, UK, 2007, pp. 53–71.

[6] T. Hayes, J. Vera, and E. Vigoda, *Randomly coloring planar graphs with fewer colors than the maximum degree*, in Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC), ACM, New York, 2007, pp. 450–458.

[7] T. Hayes and E. Vigoda, *A non-Markovian coupling for randomly sampling colorings*, in Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society, Los Alamitos, CA, 2003, pp. 618–627.

[8] M. Jerrum, *A very simple algorithm for estimating the number of k-colorings of a low-degree graph*, Random Structures Algorithms, 7 (1995), pp. 157–166.

[9] M. Jerrum and A. Sinclair, *Approximating the permanent*, SIAM J. Comput., 18 (1989), pp. 1149–1178.

[10] M. Jerrum, A. Sinclair, and E. Vigoda, *A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries*, J. ACM, 51 (2004), pp. 671–697.

[11] A. Kalai and S. Vempala, *Simulated annealing for convex optimization*, Math. Oper. Res., 31 (2006), pp. 253–266.

[12] P. W. Kasteleyn, *The statistics of dimers on a lattice* I. *The number of dimer arrangements on a quadratic lattice*, Physica, 27 (1961), pp. 1664–1672.

[13] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, *Optimization by simulated annealing*, Science, 220 (1983), pp. 671–680.

[14] L. Lovász and S. Vempala, *Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithm*, in Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society, Los Alamitos, CA, 2003, pp. 650–659.

[15] P. McCullagh, *On Prediction and Density Estimation*, preprint, University of Chicago, Chicago, IL, 2004. Available online from http://www.stat.uchicago.edu/~pmcc/reports/.

[16] H. Pasula, S. J. Russell, M. Ostland, and Y. Ritov, *Tracking many objects with many sensors*, in Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI), Stockholm, Sweden, 1999, pp. 1160–1171.

[17] A. Sinclair, *Improved bounds for mixing rates of Markov chains and multicommodity flow*, Combin. Probab. Comput., 1 (1992), pp. 351–370.

[18] D. Štefankovič, S. Vempala, and E. Vigoda, *Adaptive simulated annealing: A near-optimal connection between sampling and counting*, in Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society, Los Alamitos, CA, 2007. Preprint available from arXiv at http://arxiv.org/abs/cs.DS/0612058.

[19] L. G. Valiant, *The complexity of computing the permanent*, Theoret. Comput. Sci., 8 (1979), pp. 189–201.

SIAM J. Comput.
Vol. 37, No. 5, pp. 1455–1471

IMPROVED DYNAMIC REACHABILITY ALGORITHMS FOR DIRECTED GRAPHS*

LIAM RODITTY† AND URI ZWICK†

Abstract. We obtain several new dynamic algorithms for maintaining the transitive closure of a directed graph and several other algorithms for answering reachability queries without explicitly maintaining a transitive closure matrix. Among our algorithms are: (i) A decremental algorithm for maintaining the transitive closure of a directed graph, through an arbitrary sequence of edge deletions, in $O(mn)$ *total* expected time, essentially the time needed for computing the transitive closure of the initial graph. Such a result was previously known only for *acyclic* graphs. (ii) Two fully dynamic algorithms for answering reachability queries. The first is deterministic and has an amortized insert/delete time of $O(m\sqrt{n})$, and worst-case query time of $O(\sqrt{n})$. The second is randomized and has an amortized insert/delete time of $O(m^{0.58}n)$ and worst-case query time of $O(m^{0.43})$. This significantly improves the query times of algorithms with similar update times. (iii) A fully dynamic algorithm for maintaining the transitive closure of an acyclic graph. The algorithm is deterministic and has a worst-case insert time of $O(m)$, *constant* amortized delete time of $O(1)$, and a worst-case query time of $O(n/\log n)$. Our algorithms are obtained by combining several new ideas, one of which is a simple *sampling* idea used for detecting decompositions of strongly connected components, with techniques of Even and Shiloach [*J. ACM*, 28 (1981), pp. 1–4], Italiano [*Inform. Process. Lett.*, 28 (1988), pp. 5–11], Henzinger and King [*Proceedings of the* 36*th Annual Symposium on Foundations of Computer Science*, Milwaukee, WI, 1995, pp. 664–672], and Frigioni et al. [*ACM J. Exp. Algorithmics*, 6 (2001), (electronic)].

Key words. dynamic algorithms, transitive closure, strongly connected components

AMS subject classifications. 68W40, 68W20, 68W05, 68Q25

DOI. 10.1137/060650271

1. Introduction.

The problem of maintaining the transitive closure of a dynamic directed graph, i.e., a directed graph that undergoes a sequence of edge insertions and deletions, is a well studied and well motivated problem. Demetrescu and Italiano [5], improving an algorithm of King [15], recently obtained an algorithm for dynamically maintaining the transitive closure under a sequence of edge insertions and deletions with an amortized insert/delete time of $O(n^2)$, where n is the number of vertices in the graph. King and Thorup [17] reduced the space requirements of these algorithms. All these algorithms support *extended* insert and delete operations in which an arbitrary set of edges, all touching the same vertex, may be inserted, and a completely arbitrary set of edges may be deleted, all in one update operation.

When the transitive closure of a graph is explicitly maintained, it is of course possible to answer every reachability query, after each update, in $O(1)$ time. As the insertion or deletion of a single edge may change $\Omega(n^2)$ entries in the transitive closure matrix, an amortized update time of $O(n^2)$, in the worst-case, is essentially optimal. When the number of queries after each update operation is relatively small, it is desirable to have a dynamic algorithm with a smaller update time, at the price

*Received by the editors January 18, 2006; accepted for publication (in revised form) March 15, 2007; published electronically January 30, 2008. A preliminary version of this paper appeared in Proceedings of the 43rd Annual Syposium on Foundations of Computer Science, Vancouver, BC, Canada, 2002. This work was supported by grant I-792-136.6/2003 from the German-Israeli Foundation for Scientific Research and Development.

http://www.siam.org/journals/sicomp/37-5/65027.html

†School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel (liamr@tau.ac.il, zwick@tau.ac.il).

of a nonconstant query time. Such algorithms can escape the $\Omega(n^2)$ lower bound on the amortized update time by *implicitly* maintaining the transitive closure matrix.

Several dynamic algorithms for answering reachability queries, without explicitly maintaining the transitive closure, were developed. Most recently, Demetrescu and Italiano [5, 6] gave such a Monte Carlo algorithm with an amortized update time of $O(n^{1.58})$ and worst-case query time of $O(n^{0.58})$. They exhibit, in fact, a tradeoff between the update and query times. Smaller query times may be obtained at the cost of higher update times. However, their algorithm can handle only *acyclic* graphs, and can insert or delete only one edge at a time. Furthermore, it relies on fast rectangular matrix multiplication and thus may not be very efficient in practice. Earlier, Henzinger and King [10] gave two Monte Carlo algorithms for answering reachability queries. The first algorithm has an amortized update time of $O(m\sqrt{n}\log^2 n)$ and a worst-case query time of $O(n/\log n)$, where m is the number of edges in the graph. The second one has an amortized update time of $O(m^{0.58}n)$ and a query time of $O(n/\log n)$.

We present two new fully dynamic reachability algorithms for general graphs that improve upon the results of Henzinger and King [10]. The first is a *deterministic* algorithm that has an amortized update time of $O(m\sqrt{n})$ and a worst-case query time of $O(\sqrt{n})$. The update time of this algorithm is faster by a polylogarithmic factor than the update time of the first algorithm of Henzinger and King [10] while the query time is reduced from $O(n/\log n)$ to $O(\sqrt{n})$. Furthermore, we can obtain a tradeoff between the update and query times. For every $t \leq \sqrt{n}$, we can get an update time of $O(mn/t)$ and query time $O(t)$. This algorithm is purely combinatorial and does not use fast matrix multiplication algorithms.

Our second algorithm is a randomized algorithm with an amortized update time of $O(m^{0.58}n)$ and worst-case query time of $O(m^{0.43})$. This improves the query time of the second algorithm of Henzinger and King [10] from $O(n/\log n)$ to $O(m^{0.43})$. This algorithm does use fast matrix multiplication. We again get a tradeoff. For every $t \leq (m\log n)^{1/\omega}$, we can get an update time of $O(mn\log n/t)$ and a query time of $O(t)$, where $\omega < 2.376$ is the matrix multiplication exponent (see Coppersmith and Winograd [3]). Note that this is essentially the same tradeoff as that of the first algorithm. But, when $m \geq n^{\omega/2}\log n$, larger values of t may be chosen, giving lower update times.

We also obtain a fully dynamic reachability algorithm for acyclic graphs. This algorithm is deterministic and has a *linear* amortized update time of $O(m)$ and a worst-case query time of $O(n/\log n)$. A comparison between our dynamic reachability algorithms and the previously available ones is given in Table 1.1.

In the time bounds given above for decremental algorithms, m stands for the *initial* number of edges in the graph. In time bounds given above for fully dynamic algorithms, m stands for the *maximum* number of edges in the graph during the *phase* in which the update operation is performed. (Phases will be defined later.)

One of the ingredients used in obtaining the improved fully dynamic reachability algorithms is an improved *decremental* algorithm for maintaining the transitive closure. A decremental algorithm is an algorithm that can handle deletions but not insertions. Italiano [14] obtained a decremental algorithm for *acyclic* graphs that processes any sequence of deletions in $O(mn)$ time. Slower algorithms for general, i.e., not necessarily acyclic, graphs were obtained by La Poutré and van Leeuven [19], Frigioni et al. [8], Demetrescu and Italiano [5], and by Baswana, Hariharan, and Sen [1]. A summary of previous decremental algorithms for maintaining the transitive closure, and for answering reachability queries is given in Table 1.2. (All the algo-

Table 1.1
Fully dynamic reachability algorithms.

Graphs	Algorithm	Query	Amortized update time	Reference
DAGs	Monte Carlo	$O(1)$	$O(n^2)$	[16]
DAGs	Monte Carlo	$O(n^{0.58})$	$O(n^{1.58})$	[5]
DAGs	**Deterministic**	$\boldsymbol{O(\frac{n}{\log n})}$	$\boldsymbol{O(m)}$	**This paper**
General	Monte Carlo	$O(\frac{n}{\log n})$	$O(m\sqrt{n}\log^2 n)$	[10]
General	Monte Carlo	$O(\frac{n}{\log n})$	$O(m^{0.58}n)$	[10]
General	Monte Carlo	$O(1)$	$O(n^{2.26})$	[16]
General	Deterministic	$O(1)$	$O(n^2\log n)$	[15]
General	Deterministic	$O(1)$	$O(n^2)$	[5]
General	**Deterministic**	$\boldsymbol{O(\sqrt{n})}$	$\boldsymbol{O(m\sqrt{n})}$	**This paper**
General	**Monte Carlo**	$\boldsymbol{O(m^{0.43})}$	$\boldsymbol{O(m^{0.58}n)}$	**This paper**

Table 1.2
Decremental reachability algorithms.

Graphs	Algorithm	Query	Total update time	Reference
DAGs	Deterministic	$O(1)$	$O(mn)$	[14]
General	Monte Carlo	$O(\frac{n}{\log n})$	$O(mn\log^2 n)$	[10]
General	Deterministic	$O(1)$	$O(m^2)$	[19, 8]
General	Deterministic	$O(1)$	$O(n^3)$	[5]
General	Monte Carlo	$O(1)$	$\tilde{O}(mn^{4/3})$	[1]
General	**Las Vegas**	$\boldsymbol{O(1)}$	$\boldsymbol{O(mn)}$	**This paper**

rithms there, except that of Henzinger and King [10], explicitly maintain the transitive closure matrix.)

We obtain a new randomized decremental algorithm for maintaining the transitive closure of arbitrary, not necessarily acyclic, graphs. It processes *any* sequence of edge deletions in a total expected time of $O(mn)$. The algorithm is a Las Vegas algorithm, i.e., its answers are always correct. This matches the time bound of Italiano [14] for acyclic graphs, and answers an open problem raised there. As mentioned in the abstract, a time bound of $O(mn)$ is essentially optimal for the problem, as $\Omega(mn)$ time is needed just for computing the transitive closure of the initial graph using the currently best matrix multiplication-free algorithm. The new decremental algorithm is based on a very simple sampling idea.

Next, we adapt the results of Cohen [2] on estimating the size of the transitive closure to the dynamic setting. In particular, we obtain an incremental algorithm that can process any sequence of edge insertions and requests to estimate the number of vertices reachable from a certain vertex in $O(m\log n + q)$ time, where m is the total number of edges inserted and q is the number of queries. We also obtain such a decremental algorithm for acyclic graphs. In the fully dynamic setting, we can provide such estimates at the cost of $O(\log n)$ reachability queries.

The rest of this extended abstract is organized as follows. In the next section we present a new decremental algorithm for maintaining the strongly connected components of a directed graph. This algorithm is used in section 3 to obtain the $O(mn)$ decremental algorithm for maintaining the transitive closure of general directed graphs. In section 4 we then describe *three* new fully dynamic reachability algorithms for general graphs. (Only two of them were mentioned above.) In section 5 we describe a new fully dynamic reachability algorithm for acyclic graphs. In section 6 we sketch our dynamic size estimation results. We end in section 7 with some concluding remarks and open problems.

Recent developments. Since the appearance of the preliminary version of this paper, some additional dynamic algorithms for maintaining the transitive closure and for answering reachability queries were obtained. None of them, however, supersedes the results presented in this paper. Roditty [20] obtained another fully dynamic algorithm, with an amortized update time of $O(n^2)$ and a worst-case query time of $O(1)$, for maintaining the transitive closure matrix of a general graph. Sankowski [24] obtained a randomized algorithm with a worst-case update time of $O(n^2)$ and a worst-case query time of $O(1)$ for maintaining, with high probability, the transitive closure matrix. We [22] obtained yet another algorithm for answering reachability queries that has an amortized update time of $O(m + n \log n)$ and a query time of $O(n)$. Finally, Krommidas and Zaroliagis [18] have implemented some of the algorithms presented in this paper and they report that they work fairly well in practice.

2. Decremental maintenance of strongly connected components.

In this section we consider the dynamic maintenance of the strongly connected components (SCCs) of a directed graph under a sequence of edge deletions. This is a seemingly easier problem than the maintenance of the transitive closure of a graph. In section 3, however, we use the results of this section to obtain an improved decremental algorithm for the maintenance of the transitive closure, and in section 4 we use this decremental algorithm as a building block in our new fully dynamic reachability algorithms.

The new algorithm is given in Figure 2.1. It handles any sequence of edge deletions and queries in $O(mn+q)$ total *expected* time, where q is the number of queries. Each query is answered correctly in $O(1)$ worst-case time. The expected amortized time per edge deletion, if all edges are eventually deleted, is $O(n)$.

The algorithm starts by computing the SCCs of the graph using any linear time algorithm (see Tarjan [26], Sharir [25], Gabow [9], or Chapter 22 of Cormen et al. [4]). In each SCC C of the graph it then constructs and maintains a shortest-paths in-tree $In(w)$ and a shortest-paths out-tree $Out(w)$ rooted at a *random* representative w of this SCC. These shortest-paths trees are maintained using the decremental algorithm of Even and Shiloach [7], as adapted to directed graphs by Henzinger and King [10]. If C is composed of n' vertices and m' edges, then the total cost of maintaining these two shortest-paths trees, over any sequence of edge deletions, is $O(m'n')$.

The algorithm also maintains an array A of length n that holds for every vertex v the representative vertex of the SCC containing v. Using this array it is easy to answer any strong connectivity query in $O(1)$ time.

Edge deletions are handled as follows. If the edge $e = (u, v)$ is not contained in an SCC, i.e., if $A(u) \neq A(v)$, then nothing needs to be updated. If e is contained in an SCC C with representative vertex w, i.e., $A(u) = A(v) = w$, and e is not contained in the trees $In(w)$ and $Out(w)$, then again, the SCCs of the graph do not change and we need only record that the edge e was deleted.

The difficult case, of course, is when e is contained in one of the trees $In(w)$ or $Out(w)$. In this case, we use the decremental algorithm to update the shortest-paths trees $In(w)$ and $Out(w)$. If after this update we have $u \in In(w)$ and $v \in Out(w)$, then there is still a directed path from u to v in the graph. Thus C is still an SCC, and the partition of the graph into SCCs did not change.

```
init(V):
    1. Allocate an array $A$ of size $n$.
    2. Choose a random vertex $w \in V$.
    3. Call $findSCC(V, w)$.

findSCC(C, w):
    1. Find the SCCs $C_1, C_2, \ldots, C_k$ of the graph $G[C]$.
    2. In each SCC $C_j$, where $1 \le j \le k$, do:
        (a) If $w \in C_j$, then let $w_j \leftarrow w$.
            Otherwise, choose a random representative $w_j \in C_j$.
        (b) For every $v \in C_j$, let $A(v) \leftarrow w_j$.
        (c) Initialize decremental data structures for maintaining a
            shortest-paths in-tree $In(w_j)$ and a shortest-paths out-tree
            $Out(w_j)$ of $G[C_j]$ rooted at $w_j$.

query(u, v):
    1. If $A(u) = A(v)$ then "yes," otherwise "no."

delete(u, v):
    1. If $A(u) \neq A(v)$, i.e., $u$ and $v$ are not in the same SCC, do nothing.
    2. Otherwise, let $w \leftarrow A(u)$, and let $C$ be the vertices of the SCC
       containing $w$.
    3. Delete the edge $(u, v)$, if necessary, from the trees $In(w)$ and
       $Out(w)$ using the appropriate decremental data structures.
    4. If $u \notin In(w)$ or $v \notin Out(w)$, i.e., $C$ decomposed, then call
       $findSCC(C, w)$.
```

FIG. 2.1. *A randomized decremental algorithm for maintaining strongly connected components.*

If $u \notin In(w)$ or $v \notin Out(w)$, then clearly C is no longer a SCC. We construct, in $O(m' + n')$ time, the new SCCs $C_1, C_2, \ldots, C_k$ to which C decomposed. Let C_i be the new SCC containing w. We let $w_i = w$ be the representative of C_i. In every other SCC C_j, for $j \neq i$, we choose a *random* representative $w_j \in C_j$.

By removing from $In(w)$ and $Out(w)$ the vertices that do *not* belong to C_i, we obtain shortest-paths trees that span C_i. It is crucial for the analysis of the algorithm to note that the decremental data structures maintaining these two shortest-paths trees do not have to be reinitialized.

From each random representative w_j, for $j \neq i$, we build from scratch shortest-paths trees $In(w_j)$ and $Out(w_j)$ that span C_j, and initialize the data structure of Even and Shiloach [7] for maintaining them. Finally, we update the array A accordingly. We now claim the following theorem.

THEOREM 2.1. *The algorithm of Figure* 2.1 *correctly handles any sequence of deletions and strong connectivity queries. Each query is answered in* $O(1)$ *time. The expected running time of the algorithm, for any sequence of deletions and queries is* $O(mn + q)$, *where* q *is the number of queries.*

Proof. The correctness of the algorithm follows easily from the above discussion. (Note that the random choices of the representatives affect only the running time, not the answers given.) It remains, therefore, to show that the expected time spent in processing *all* edge deletions is only $O(mn)$.

Let $f(m, n)$ be an upper bound on the expected running time of the algorithm on the worst possible strongly connected graph with m edges and n vertices, and for the worst sequence of edge deletions. (If the initial graph is not strongly connected, we repeat the analysis in each strongly connected component.) In the upper bound $f(m,n)$ we charge $m'n'$ units of time for the decremental maintenance of the in-tree and out-tree of an SCC containing, initially, n' vertices and m' edges, even if the actual cost of maintaining these trees is smaller.

We claim that

$$f(m,n) \ \leq \ mn + \sum_{i=1}^{k} \left(f(m_i,n_i) - \frac{m_i n_i^2}{n} \right),$$

for some $k \geq 2$ and $m_1, m_2, \ldots, m_k \geq 0$, $n_1, n_2, \ldots, n_k \geq 1$ such that $\sum_{i=1}^{k} m_i \leq m$ and $\sum_{i=1}^{k} n_i = n$. Here, k is the number of SCCs to which the graph breaks when it is no longer strongly connected, and m_i and n_i, respectively, are the number of edges and vertices in the ith SCC. (Note that k, the n_i's and the m_i's do not depend on the random choices made by the algorithm.)

The term mn covers the initialization cost of the algorithm and the cost of *all future maintenance operations* performed on the shortest-paths trees $In(w)$ and $Out(w)$. When the graph breaks into the k SCCs, the algorithm continues independently on each one of them. So we clearly have $f(m,n) \leq mn + \sum_{i=1}^{k} f(m_i,n_i)$.

This naive estimate fails, however, to take advantage of the following fact. The new component that contains w, the representative of the original component, *inherits* the shortest-paths trees $In(w)$ and $Out(w)$, and does not have to pay for their construction and maintenance. Furthermore, as w was randomly chosen, the larger a new component is, the more likely it is to receive this "gift." The probability that a new component of n_i vertices will contain w is n_i/n. Thus, with a probability of n_i/n, the term $m_i n_i$, incorporated into $f(m_i,n_i)$, can be dispensed with, giving the desired relation. We now claim the following lemma.

LEMMA 2.2. *$f(m,n) \leq 2mn$.*

Proof. The proof is by induction. The basis of the induction is easily established. Suppose now that the claim holds for any (m',n') with $m' < m$ and $n' < n$. We show that it also holds for (m,n). We have to verify that

$$mn + 2\sum_{i=1}^{k} m_i n_i - \sum_{i=1}^{k} \frac{m_i n_i^2}{n} \ \leq \ 2mn.$$

Letting $x_i = m_i/m$ and $y_i = n_i/n$, so that $x_i, y_i \geq 0$, $\sum_{i=1}^{k} x_i \leq 1$, and $\sum_{i=1}^{k} y_i = 1$, we get after a simple manipulation that we have to verify that

$$2\sum_{i=1}^{k} x_i y_i - \sum_{i=1}^{k} x_i y_i^2 \ \leq \ 1.$$

We show that in fact

$$2\sum_{i=1}^{k} x_i y_i - \sum_{i=1}^{k} x_i y_i^2 \ \leq \ \sum_{i=1}^{k} x_i \ \leq \ 1.$$

This follows as we have

$$\sum_{i=1}^{k} x_i - 2\sum_{i=1}^{k} x_i y_i + \sum_{i=1}^{k} x_i y_i^2 \ = \ \sum_{i=1}^{k} x_i (1-y_i)^2 \ \geq \ 0.$$

This completes the proof of the lemma. □

This completes the proof of the theorem. □

3. Decremental maintenance of the transitive closure. Our goal in this section is to prove the following two theorems.

THEOREM 3.1. *There is a randomized algorithm for maintaining the transitive closure matrix of a graph that undergoes a sequence of edge deletions whose total expected running time is $O(mn)$.*

THEOREM 3.2. *There is a deterministic algorithm for maintaining the transitive closure matrix of a graph that undergoes a sequence of edge deletions whose total running time is $O(mn+del{\cdot}m)$, where del is the number of delete operations performed on the graph. Each delete operation may remove an arbitrary set of edges from the graph.*

The first result is obtained by combining the algorithm for the decremental maintenance of the strongly connected components of a graph, described in the previous section, with an algorithm of Frigioni et al. [8] for the decremental maintenance of the transitive closure matrix. The second result is obtained by a small modification of the algorithm of Frigioni et al. For completeness, we sketch the operation of the algorithm of Frigioni et al. and describe the modifications needed to obtain our results.

Italiano [14] describes a deterministic algorithm, with a total running time of $O(mn)$, for the decremental maintenance of the transitive closure matrix of an *acyclic* directed graph. Frigioni et al. [8] extend Italiano's algorithm so that it could handle general, not necessarily acyclic, graphs. Frigioni et al. [8] report that their algorithm works well in practice, though its worst-case time complexity is $O(m^2)$.

We begin with a sketch of the operation of Italiano's algorithm. The algorithm maintains, in addition to the transitive closure matrix M, a collection of reachability trees, one rooted at every vertex of the graph. The tree of a vertex v, denoted by $T(v)$, contains all the vertices in the current version of the graph that are reachable from v. Note that $M(v,u)=1$ if and only if $u \in T(v)$. Every vertex u has two linked lists $E_{in}(u)$ and $E_{out}(u)$ of its incoming and outgoing edges that were not yet deleted. If $u \in T(v)$, then we let $p(v,u)$ be a pointer to the edge (u',u) in $E_{in}(u)$ such that u' is the parent of u in $T(v)$. If $u \notin T(v)$, then $p(v,u)=null$.

When an edge (u,w) is deleted from the graph, the algorithm performs the following operations. For every vertex v, it checks whether (u,w) is an edge of the tree $T(v)$. (This is done by checking whether $p(v,w)$ points to (u,w).) If (u,w) is not an edge of $T(v)$, then nothing needs to be done. Otherwise, if (u,w) is a tree edge, the algorithm tentatively sets $p(v,w)$ to point to the next edge (u',w) in $E_{in}(w)$, or to *null*, if (u,w) is the last edge of $E_{in}(w)$. Note that if $u' \in T(v)$, then the new edge (u',w) reconnects w to $T(v)$. As this condition still needs to be checked, the algorithm sets $R(v) \leftarrow \{w\}$. If (u,w) is not an edge of $T(v)$, or if (u,w) is the last edge in $E_{in}(w)$, it lets $R(v) \leftarrow \phi$. The set $R(v)$ is thus the set of vertices with tentative parent pointers that might or might not connect them to the remaining part of $T(v)$. After these operations, the edge (u,w) is removed from the graph, i.e., from the lists $E_{out}(u)$ and $E_{in}(w)$.

For every vertex v, the algorithm now needs to check the tentative pointers of the vertices in $R(v)$. While there is a vertex $w \in R(v)$, the algorithm scans the edges of $E_{in}(w)$, starting from the edge pointed to by $p(v,w)$, until an edge (u',w) for which $p(v,u') \neq null$ is found, or until the list $E_{in}(w)$ is exhausted. If such an edge is found, then w is removed from $R(v)$. If the list $E_{in}(w)$ is exhausted before finding such an edge, the algorithm sets $p(v,w)$ to *null* and $M(v,w)$ to 0. It then scans all the outgoing edges of w. If (w, w') is a tree edge, then it adds w' to $R(v)$.

As shown by Italiano [14], the algorithm sketched previously correctly maintains the transitive closure of an *acyclic* graph that undergoes a sequence of edge deletions, and its total running time is $O(mn)$. To see that the total running time of the algorithm is indeed $O(mn)$, note that the lists $E_{in}(u)$ and $E_{out}(u)$ are examined only once per reachability tree.

The algorithm of Frigioni et al. [8] maintains the strongly connected components of the graph, and the *skeleton* of the graph, i.e., the acyclic graph induced on the strongly connected components. The skeleton is maintained using Italiano's algorithm [14].

For each SCC, the algorithm of Frigioni et al. [8] maintains a *sparse certificate* composed of an in-tree and an out-tree rooted at an arbitrary vertex. When an edge from this certificate is deleted, their algorithm may have to spend $O(m + n)$ time to check whether the SCC decomposed. As this may happen every time an edge is deleted, the worst case total running time of the algorithm may be $\Omega(m^2)$.

However, the total running time of the algorithm of Frigioni et al. [8], *excluding* the time needed to detect decompositions of SCCs is only $O(mn)$. Thus, combining their algorithm with our algorithm for maintaining the SCCs yields a decremental algorithm for maintaining the transitive closure of general graphs with a total expected time of $O(mn)$, matching the time bound of Italiano [14] for acyclic graphs. This proves Theorem 3.1.

To provide a proof of Theorem 3.2 we need to sketch the operation of the algorithm of Frigioni et al. [8] in more detail. The algorithm of Frigioni et al. [8] is similar to the algorithm of Italiano [14], with SCCs playing the role played by vertices in the algorithm of Italiano. Some of the details, however, are slightly more involved. A sketch of (a variant of) the algorithm of Frigioni et al. [8] follows.

Every vertex u has a pointer $C(u)$ to the component containing it. For every component C, the algorithm maintains three linked lists $E_{in}(C)$, $E_{out}(C)$, and $E_{int}(C)$ of the incoming, outgoing, and the internal edges of the component C. An edge (u, w) belongs to $E_{in}(C)$ if $u \notin C$ while $w \in C$, to $E_{out}(C)$ if $u \in C$ while $w \notin C$, and to $E_{int}(C)$ if $u, w \in C$. For every component C the algorithm maintains a tree $T(C)$ of all the components reachable from C. If $C_2 \in T(C_1)$, then we let $p(C_1, C_2)$ be a pointer to the edge (u', u) in $E_{in}(C_2)$ such that $C(u')$ is the parent of $C(u) = C_2$ in $T(C_1)$. If $C_2 \notin T(C_1)$, then $p(C_1, C_2) = null$.

The algorithm handles the deletion of a set of edges E' in the following way. First it lets $E' = E'_{int} \cup E'_{ext}$, where E'_{int} are edges that connect two vertices that are in the same component, while E'_{ext} are edges that connect vertices in two different components. The algorithm first removes all the edges of E'_{int} and then all the edges of E'_{ext}.

The first step is the computation of the new SCCs. This can be done *deterministically* in $O(m+n)$ time by simply recomputing the SCCs from scratch. (To get an algorithm that satisfies the conditions of Theorem 3.1 we replace this step with the randomized algorithm of the previous section.) Suppose that a component C breaks into k new components $C_1, C_2, \ldots, C_k$. The first step is to split the lists $E_{in}(C)$, $E_{out}(C)$, and $E_{int}(C)$ into new lists $E_{in}(C_i)$, $E_{out}(C_i)$, and $E_{int}(C_i)$, for $1 \leq i \leq k$, and to replace the pointers $p(D, C)$, for every component D by new pointers $p(D, C_i)$. This step also constructs for each component D a set $R(D)$ of the components that need to be reconnected, if possible, to $T(D)$. We also initialize new reachability trees for the new components $C_1, C_2, \ldots, C_k$.

The incoming edges of C are now split between the new components $C_1, C_2, \ldots, C_k$. The splitting of $E_{in}(C)$ is done as follows. We scan the edges $E_{in}(C)$ and move each edge (u, w), where $w \in C_i$, to the list $E_{in}(C_i)$. If $p(D, C) = (u, w)$, then we let $p(D, C_i) = (u, w)$, while $p(D, C_j)$, for each $j \neq i$, is set to the next edge to be added to $E_{in}(C_j)$. In the latter case, we also add C_j to $R(D)$, as C_j lost its link to $T(D)$. (Note that $p(D, C_j)$ now is not the edge connecting C_j to $T(D)$ but rather the first edge that should be checked.) The list $E_{out}(C)$ is split in a similar manner. Finally, each edge $(u, w) \in E_{int}(C)$ with $u \in C_i$ and $w \in C_j$ is moved into $E_{int}(C_i)$, if $i = j$, and to $E_{out}(C_i)$ and $E_{in}(C_j)$, otherwise. It is important to note that the edges of $E_{int}(C)$ are placed at the end of the lists $E_{out}(C_i)$ and $E_{in}(C_j)$. We can now remove the edges of E'_{int} from the graph.

We now deal with the deletion of the edges of E'_{ext}. Suppose that $(u, w) \in E'_{ext}$. If $p(D, C) = (u, w)$, we move $p(D, C)$ to point to the next edge in $E_{in}(C)$, or to null if there is no such edge, and add C to $R(D)$.

Finally, we try to repair the trees. For every component D, while $R(D)$ is not empty, we choose $C \in R(D)$ and scan the edges of $E_{in}(C)$, starting from $p(D, C)$, until an edge (u, w) for which $p(D, C(u)) \neq \mathit{null}$ is found. If such an edge is found, we let $p(D, C)$ point to this edge and remove C from $R(D)$. Otherwise, we find all the components C' for which $p(D, C') = (w, v)$ with $w \in C$ and add them to $R(D)$.

This completes the sketch of (a variant of the) algorithm of Frigioni et al. [8]. Frigioni et al. [8] show that the algorithm correctly maintains the transitive closure matrix of the graph. It is easy to check that the worst case total running time of the algorithm is indeed $O(mn + \mathit{del} \cdot m)$, where del is the number of delete operations performed. This completes the proof of Theorem 3.2.

4. Fully dynamic reachability algorithms.

4.1. The first fully dynamic algorithm. Our first fully dynamic reachability algorithm is given in Figure 4.1. It is essentially a combination of an algorithm of Henzinger and King [10] with our improved decremental reachability algorithm, or with the somewhat slower, but deterministic algorithm of Frigioni et al. [8] described in the previous section.

The algorithm works in *phases*. In the beginning of each phase, a decremental reachability data structure is initialized. We let S be the set of vertices that were centers of insertions during the phase. Initially $S = \phi$. When a set of edges E_v, all touching v, is inserted, we add v to S and construct reachability trees $In(v)$ and $Out(v)$ rooted at v. When the size of S, the set of insertion centers, reaches t, a parameter fixed in advance, the phase is declared over, and all the data structures are reinitialized.

The deletion of an arbitrary set E' of edges is handled as follows. First, the edges of E' are removed from the decremental data structure. Next, for every $w \in S$, the shortest-paths trees $In(w)$ and $Out(w)$ are rebuilt from scratch.

A query $query(u, v)$ is answered as follows. First the decremental data structure is queried to see whether there is a directed path from u to v composed solely of edges that were present in the graph at the start of the current phase. If not, it is checked whether there exists $w \in S$ such that $u \in In(w)$ and $v \in Out(w)$. If such a vertex w exists, then the answer is "yes."

It is easy to check that the answer given for each query is always correct. Clearly, if $query(u, v)$ returns "yes," then there is indeed a path from u to v in the graph. Suppose now that there is a path p from u to v in the graph. If this path uses only "old" edges, i.e., edges that were not inserted in the current phase, then the decremental data structure would signal that out. Otherwise, let w be the *last* vertex on a path from u to v that was the center of an insert operation during the current phase. This

```
init:
    1. Initialize a decremental reachability data structure.
    2. Let $S \leftarrow \phi$.

query(u, v):
    1. Query the decremental reachability data structure.
    2. For each $w \in S$ check if $u \in In(w)$ and $v \in Out(w)$.

delete(E'):
    1. Let $E \leftarrow E - E'$.
    2. Delete $E'$ from the decremental data structure.
    3. For every $w \in S$, rebuild the trees $In(w)$ and $Out(w)$.

insert(E_v):
    1. Let $E \leftarrow E \cup E_v$.
    2. Let $S \leftarrow S \cup \{v\}$.
    3. If $|S| > t$, then call init.
    4. Otherwise, construct the trees $In(v)$ and $Out(v)$.
```

FIG. 4.1. *The first fully dynamic reachability algorithm for general graphs.*

insert operation added w to S and constructed the trees $In(w)$ and $Out(w)$. At the time of this insertion all the edges of the path p were already present in the graph, so $u \in In(w)$ and $v \in Out(w)$. Some edges from these trees may be subsequently deleted, but as the path p remains in the graph, the vertex u would stay in $In(w)$, and similarly v would stay in $Out(w)$. This completes the proof of correctness. We claim the following theorem.

THEOREM 4.1. *For any $t \leq \sqrt{n}$, the algorithm of Figure 4.1 handles each insert or delete operation in $O(mn/t)$ amortized time, and answers each query correctly in $O(t)$ worst-case time. In particular, when $t = \sqrt{n}$, the amortized update time is $O(m\sqrt{n})$, and the worst-case query time is $O(\sqrt{n})$.*

Proof. Assume, at first, that our decremental reachability algorithm is used. The expected complexity of setting up the decremental data structure in the beginning of each phase, and of handling all subsequent delete operations on it, is only $O(mn)$. As each phase, except possibly the last phase, is composed of at least t update operations, we can cover this cost by charging $O(mn/t)$ of these operations to each update.

Each delete operation involves the recomputation of up to $2t$ trees. This is easily done in $O(mt)$ time. An insert operation is even cheaper as only two trees need to be constructed.

The total expected amortized cost per insert or delete operation is therefore $O(mn/t + mt)$. When $t \leq \sqrt{n}$, the first term dominates the second and the expected cost per operation is $O(mn/t)$, assuming that at least t update operations are performed. The query time is clearly $O(t)$.

As presented, the algorithm is randomized (Las Vegas). We can get a deterministic version of the algorithm, with the same time bounds, by using the variant of the decremental algorithm of Frigioni et al. [8] described in section 3. □

4.2. The second fully dynamic algorithm. Our second fully dynamic reachability algorithm is given in Figure 4.2. It is essentially a combination of a second algorithm of Henzinger and King [10] with our decremental reachability algorithm, or with the algorithm of Frigioni et al. [8].

```
init:
    1. Initialize a decremental reachability data structure.
    2. Let $S$ be a random set of $t$ vertices.
    3. For every $w \in S$, construct shortest-paths trees $In(w)$ and $Out(w)$
       of depth at most $(cn \ln n)/t$, and initialize decremental data structure for them.
    4. Call $build(S)$.
```

```
build(S):
    1. Construct Boolean matrices $A_1, A_2$, and $B$ of sizes $n \times |S|$, $|S| \times n$,
       and $|S| \times |S|$:
        (a) $A_1(u, w) = 1$ iff $u \in In(w)$, for every $u \in V$ and $w \in S$.
        (b) $A_2(w, v) = 1$ iff $v \in Out(w)$, for every $w \in S$ and $v \in V$.
        (c) $B(w_1, w_2) = 1$ iff $w_1 \in In(w_2)$, for every $w_1, w_2 \in S$.
    2. Compute $B^*$, the transitive closure of $B$, and $A_1^* = A_1 B^*$.
```

```
query(u, v):
    1. Query the decremental data-structure.
    2. Check whether there exists $w \in S$ such that $A_1^*(u, w) = A_2(w, v) = 1$.
```

```
delete(E'):
    1. $E \leftarrow E - E'$.
    2. Delete $E'$ from the decremental data structure.
    3. For every $w \in S$, update the shortest-paths trees $In(w)$ and $Out(w)$
       of depth at most $(cn \ln n)/t$ using the decremental algorithm for
       maintaining shortest-paths trees.
    4. Call $build(S)$.
```

```
insert(E_v):
    1. $E \cup E \cup E_v$.
    2. Let $S \leftarrow S \cup \{v\}$.
    3. If $|S| > 2t$, then call init.
    4. Otherwise, construct shortest-paths trees $In(v)$, $Out(v)$ of depth
       at most $(cn \ln n)/t$, and initialize a data structure for maintaining
       them under a sequence of edge deletions.
    5. Call $build(S)$.
```

FIG. 4.2. *The second fully dynamic reachability algorithm for general graphs.*

The algorithm again works in *phases.* In the beginning of each phase, a decremental reachability data structure is again initialized. The algorithm again maintains a set S of *special* vertices. For each vertex $w \in S$, the algorithm maintains an in-tree $In(w)$ and an out-tree $Out(w)$. These trees are *shortest-paths* trees that contain all vertices that are at distance at most $(cn \ln n)/t$ from w, where c is some fixed constant. (For concreteness, we choose $c = 10$.) These trees are maintained using the algorithm of Even and Shiloach [7]. In the beginning of each phase, t *random* vertices are placed in S.

The algorithm also maintains two Boolean matrices, A_1^* of size $n \times |S|$, and A_2 of size $|S| \times n$. The columns of A_1^* and the rows of A_2 are indexed by the elements of S. For every $u \in V$ and $w \in S$, we have $A_1^*(u, w) = 1$ if and only if there is a path (of arbitrary length) in the graph from u to w, and $A_2(w, u) = 1$ if and only if there is a path of length at most $(cn \ln n)/t$ from w to u.

When a set of edges E_v touching v is inserted, we add v to the set S of special vertices and construct shortest-paths trees $In(v)$ and $Out(v)$ of depth at most $(cn \ln n)/t$ rooted at v. When the size of the set S reaches $2t$, a parameter fixed in advance, the phase is over, and all data structures are reinitialized.

The deletion of an arbitrary set E' of edges is handled as follows. First the edges of E' are removed from the decremental data structure. Next, for every $w \in S$, the shortest-paths trees $In(w)$ and $Out(w)$ are updated using the algorithm of Even and Shiloach [7].

A query $query(u, v)$ is answered as follows. First the decremental data structure is queried to see whether there is a directed path from u to v composed solely of "old" edges. If not, it is checked whether there exists $w \in S$ such that $A_1^*(u, w) = A_2(w, v) = 1$. The correctness of the algorithm relies on the following observation of Ullman and Yannakakis [27].

LEMMA 4.2. *Let $G = (V, E)$ be a directed graph on n vertices. Let S be a set of $(cn \ln n)/t$ random vertices. Then, with a probability of at least $1 - n^{-(c-3)}$, for every two vertices $u, v \in V$, if there is a path from u to v in G, then there is also such a path that among any t consecutive vertices on it there is a vertex from S.*

As stated, the lemma applies to a fixed graph. However, it is easy to adapt it to our dynamic setting.

COROLLARY 4.3. *Let $G_1, G_2, \ldots, G_\ell$ be directed graphs on the same set of n vertices. Let S be a set of $(cn \ln n)/t$ random vertices. Then, with a probability of at least $1 - \ell n^{-(c-3)}$, for every $1 \le i \le \ell$ and every $u, v \in V$, if there is a path from u to v in G_i, then there is also such a path in G_i that among any t consecutive vertices on it there is a vertex from S.*

The random set S may be chosen, of course, without knowing the sequence of graphs. We note in passing that similar ideas are also used by Zwick [28] and King [15].

THEOREM 4.4. *For any $t \le (m \log n)^{1/\omega}$, the algorithm of Figure 4.1 handles each insert or delete operation in $O(mn \log n/t)$ amortized time and answers each query correctly, with very high probability, in $O(t)$ worst-case time. In particular, when $t = (m \log n)^{1/\omega}$, the expected amortized update time is $O((m \log n)^{1-1/\omega} n)$, and the worst-case query time is $O((m \log n)^{1/\omega})$.*

Proof. The correctness of the algorithm follows easily from Corollary 4.3. As with the previous algorithm, the $O(mn)$ complexity of setting up and maintaining the decremental data structure is split among the at least t updates operations of a phase.

In the beginning of each phase, the algorithm also sets up $2t$ shortest-paths trees of depth at most $(cn \ln n)/t$. The cost of setting up and maintaining these trees throughout the phase, using the algorithm of Even and Shiloach [7], is $O(2t \cdot m \cdot \frac{cn \ln n}{t}) = O(mn \log n)$. This cost is again split among the update operations of the phase.

Each delete operation updates the decremental data structure and the shortest-paths trees. These operations are already accounted for. It also involves the call $build(S)$. As $|S| \le 2t$, the complexity of this procedure is $O(\frac{n}{t} \cdot t^\omega) = O(nt^{\omega-1})$, where $\omega < 2.376$ is the exponent of matrix multiplication.

Each insert operation constructs two new shortest-paths trees of depth at most $(cn \ln n)/t$. The total cost of maintaining these trees throughout the phase, using the algorithm of Even and Shiloach [7] is $O(mn \log n/t)$. The cost of calling $build(S)$ is

```
init:
    1. Initialize a decremental reachability data structure.
    2. Let $S$ be a random set of $t$ vertices.
    3. For every $w \in S$, construct shortest-paths trees $In(w)$ and $Out(w)$ of
       depth at most $(cn \ln n)/t$, and initialize decremental data structures
       for maintaining them.
    4. Construct a directed graph $H = (S, F)$, where $F = \{(w_1, w_2) \in S^2 \mid
       w_1 \in In(w_2)\}$.
    5. Initialize a fully dynamic algorithm for maintaining the transitive
       closure $B^*$ of $H$.
query(u, v):
    1. Query the decremental data-structure.
    2. Check whether there exist $w_1, w_2 \in S$ such that $u \in In(w_1)$,
       $B^*(w_1, w_2) = 1$, and $v \in Out(w_2)$.
delete(E'):
    1. $E \cup E - E'$.
    2. Delete $E'$ from the decremental data structure.
    3. For every $w \in S$, update the shortest-paths trees $In(w)$ and $Out(w)$
       of depth at most $(cn \ln n)/t$ using the decremental algorithm.
    4. Update the transitive closure $B^*$ of the graph $H$ after the removal
       of edges from $F$.
insert(E_v):
    1. $E \cup E \cup E_v$.
    2. Let $S \leftarrow S \cup \{v\}$.
    3. If $|S| > 2t$, then call init.
    4. Otherwise, construct shortest-paths trees $In(v)$ and $Out(v)$ of depth
       at most $(cn \ln n)/t$ and initialize decremental data structures for
       maintaining them.
    5. Update the transitive closure $B^*$ of the graph $H$ after the addition
       of $v$ to it.
```

FIG. 4.3. *A third fully dynamic reachability algorithm for general graphs.*

again $O(nt^{\omega-1})$.

Thus, the total amortized cost per each update operation is $O(\frac{mn \log n}{t} + nt^{\omega-1})$. When $t \leq (m \log n)^{1/\omega}$, the first term is the dominant term, and the expected amortized time per update is $O((mn \log n)/t)$. Each query is clearly answered in $O(t)$ worst-case time. □

We note in passing that the tradeoff of an amortized update time of $O((mn \log n)/t)$ and query time of $O(t)$ can be extended to values of t that are slightly larger than $(m \log n)^{1/\omega}$ using the fast rectangular matrix multiplication algorithms of Huang and Pan [13].

4.3. A third fully dynamic reachability algorithm. Our third fully dynamic reachability algorithm for general graphs is given in Figure 4.3. It is somewhat similar to our second algorithm. However, it does not maintain the matrices A_1^* and A_2, and it uses a fully dynamic algorithm, e.g., the algorithm of Demetrescu and Italiano [5], to maintain the matrix B^*. We now claim the following theorem.

THEOREM 4.5. *For any $1 \leq t \leq \sqrt{m}$, the algorithm of Figure* 4.3 *handles each insert or delete operation in $O(mn \log n/t)$ amortized time and answers each query correctly, with very high probability, in $O(t^2)$ worst-case time. In particular, when $t = m^{(1-\epsilon)/2}$, the amortized update time is $O(m^{(1+\epsilon)/2} n \log n)$, and the worst-case query time is $O(m^{1-\epsilon})$.*

Proof. The correctness proof of the algorithm is identical to the correctness proof of the second fully dynamic algorithm. The cost of initializing a phase is $O(mn \log n + t^3)$. The cost of an insert operation is $O(mn \log n/t + t^2)$. (The first term is the cost of constructing and maintaining the trees $In(v)$ and $Out(v)$. The second term is the cost of updating of the matrix B^* using the fully dynamic algorithm for maintaining the transitive closure.) The added cost of a delete operation is only $O(t^2)$, the cost of updating B^*. Thus, the amortized cost of each update operation is $O(mn \log n/t + t^2)$. As $t \leq \sqrt{m}$, the first term is always dominant. The query time is clearly $O(t^2)$. □

5. A very simple fully dynamic reachability algorithm for acyclic graphs. A very simple fully dynamic reachability algorithm for acyclic graphs is presented in Figure 5.1. The algorithm is based on the main idea of King [15]. The acyclicity assumption allows us to greatly simplify the algorithm, and to obtain the first fully dynamic reachability algorithm, for acyclic graphs, with a *linear*, i.e., $O(m)$, amortized update time. The query time of the algorithm, $O(n/\log n)$, is quite large. However, it is still much faster than the $\Omega(m)$ time that may be needed to answer such a query without a dynamic data structure.

```
init(V):
    • For every $v \in V$ construct reachability trees $In(v)$ and $Out(v)$ and
      initialize appropriate decremental data structures for them.
query(u, v):
    • For every $w \in V$, check whether $u \in In(w)$ and $v \in Out(w)$.
delete(E'):
    • Delete $E'$ from all reachability trees and update each one of
      them using the decremental single-source reachability algorithm
      for DAGs.
insert(E_v):
    • Call $init(\{v\})$.
```

FIG. 5.1. *A very simple dynamic reachability algorithm for acyclic graphs.*

Italiano [14] showed that, in acyclic graphs, a forest of reachability trees, one rooted at each vertex, can be decrementaly maintained in $O(mn)$ total time. His result is, in fact, stronger. Each of these trees can be *individually* maintained in $O(m)$ total time. Our algorithm exploits this fact.

THEOREM 5.1. *The algorithm of Figure* 5.1 *handles each insert operation, which keeps the graph acyclic, in $O(m)$ worst-case time, each delete operation in $O(1)$ amortized time, and answers every reachability query correctly in $O(n/\log n)$ worst-case time.*

Proof. The algorithm starts by constructing a forest of in-trees and a forest of out-trees. Each of these trees is individually maintained using the data structure of Italiano [14]. When a set E' of edges is deleted, we simply update each of these trees individually. To insert a set E_v of edges, we simply rebuild the trees $In(v)$

and $Out(v)$. The cost of building these two trees, and of maintaining them through all future delete operations, is only $O(m)$. Thus, the cost of all delete operations is covered by either the initialization cost, of $O(mn)$, or by preceding insert operations.

A query $query(u, v)$ is answered by checking whether there is a $w \in V$ such that $u \in In(w)$ and $v \in Out(w)$. If there is a path p from u to v, then this condition holds when w is the last vertex on the path that was the center of an insert operation, or by u and v themselves, if no such insertions took place. As described, each query would require $O(n)$ time.

However, it is easy to reduce the query time to $O(n/\log n)$. The algorithm essentially maintains two $n \times n$ Boolean matrices A and B such that $A(u, w) = 1$ if and only if $u \in In(w)$, and $B(w, v) = 1$ if and only if $v \in Out(w)$. We can *pack* each row of A and B into $n/\log n$ machine words, and each query would then require only $O(n/\log n)$ time. □

6. Dynamic estimation of the size of reachability sets. Cohen [2] presents an $O(m)$ time randomized algorithm that estimates, for every vertex of a given directed graph, the number of vertices that are reachable from that vertex. We discuss here adaptations of her ideas to the dynamic setting.

One of the variants of the algorithm of Cohen [2] works roughly as follows. It chooses a random permutation on the vertices of the graph and labels the vertices according to it. For every vertex v, it then finds the smallest label $s(v)$ assigned to a vertex reachable from v. In the static setting, this can be easily done in $O(m)$ time. Then, $n/s(v)$ is a reasonable estimate to the number of vertices reachable from v. To obtain higher accuracy and higher confidence, this experiment is repeated several times and the results are combined in several possible ways. See Cohen [2] for exact details.

Here we make the simple observation that a request to estimate the size of a reachability set can be reduced to $O(\log n)$ reachability queries. This is done as follows. Let $\epsilon > 0$. Add $k = \log_{1+\epsilon} n$ new vertices $u_1, u_2, \ldots u_k$ to the graph. For every $1 \leq i \leq k$, add an edge (v, u_i) for every vertex $v \in V$ whose label is in $[(1+\epsilon)^{i-1}, (1+\epsilon)^i]$. Now, for every $v \in V$, the queries $query(v, u_i)$, for $1 \leq i \leq k$, allow us to estimate $s(v)$ with a relative error of ϵ, which is good enough for our purposes. Furthermore, these queries involve only $k = O(\log n)$ destinations. This can be exploited, especially in the semidynamic setting, to obtain more efficient algorithms as there is only a logarithmic number of trees to which we save reachability information. The cost of maintaining a reachability tree while edges are added to the graph is $O(m)$. Thus, this leads to an incremental algorithm whose total running time is $O(m \log n + q)$. The cost of maintaining a reachability tree while edges are removed from a directed acyclic graph is $O(m)$; thus, this leads to a decremental algorithm whose total running time is $O(m \log n + q)$ in directed acyclic graphs.

7. Concluding remarks and open problems. We presented an essentially optimal decremental algorithm for maintaining the transitive closure of a general graph. We also presented several improved fully dynamic algorithms for the reachability problem. There is still a huge gap, however, between the results obtained here, and elsewhere, for *directed* graphs, and the polylogarithmic results available for *undirected* graphs (see Henzinger and King [11] and Holm, de Lichtenberg, and Thorup [12]).

Many open problems still remain. Among them are the following.

1. Is there a decremental algorithm for maintaining the strongly connected components of a directed graph whose total running time is $O(mn)$?

2. Is there a decremental algorithm for maintaining a reachability tree, from a *single* source in a general directed graph whose total running time is $O(mn)$? (Note that the decremental maintenance of a single-source shortest paths tree seems to be a harder task than just maintaining a reachability tree. See [23].)
3. Is there a *deterministic* decremental algorithm for maintaining the transitive closure of a general directed graph whose total running time is $O(mn)$?

REFERENCES

[1] S. Baswana, R. Hariharan, and S. Sen, *Improved decremental algorithms for transitive closure and all-pairs shortest paths*, in Proceedings of the 34th Annual Symposium on Theory of Computing, Montréal, QC, Canada, 2002, pp. 117–123.

[2] E. Cohen, *Size-estimation framework with applications to transitive closure and reachability*, J. Comput. System Sci., 55 (1997), pp. 441–453.

[3] D. Coppersmith and S. Winograd, *Matrix multiplication via arithmetic progressions*, J. Symbolic Comput., 9 (1990), pp. 251–280.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed., MIT Press, Cambridge, MA, 2001.

[5] C. Demetrescu and G. F. Italiano, *Fully dynamic transitive closure: Breaking through the $O(n^2)$ barrier*, in Proceedings of the 41st Annual Symposium on Foundations of Computer Science, Redondo Beach, CA, 2000, pp. 381–389.

[6] C. Demetrescu and G. F. Italiano, *Trade-offs for fully dynamic transitive closure on DAGs: Breaking through the $O(n^2)$ barrier*, J. ACM, 52 (2005), pp. 147–156.

[7] S. Even and Y. Shiloach, *An on-line edge-deletion problem*, J. ACM, 28 (1981), pp. 1–4.

[8] D. Frigioni, T. Miller, U. Nanni, and C. Zaroliagis, *An experimental study of dynamic algorithms for transitive closure*, ACM J. Exp. Algorithmics, 6 (2001), (electronic).

[9] H. N. Gabow, *Path-based depth-first search for strong and biconnected components*, Inform. Process. Lett., 74 (2000), pp. 107–114.

[10] M. Henzinger and V. King, *Fully dynamic biconnectivity and transitive closure*, in Proceedings of the 36th Annual Symposium on Foundations of Computer Science, Milwaukee, WI, 1995, pp. 664–672.

[11] M. Henzinger and V. King, *Randomized fully dynamic graph algorithms with polylogarithmic time per operation*, J. ACM, 46 (1999), pp. 502–516.

[12] J. Holm, K. de Lichtenberg, and M. Thorup, *Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree,* 2*-edge, and biconnectivity*, J. ACM, 48 (2001), pp. 723–760.

[13] X. Huang and V. Y. Pan, *Fast rectangular matrix multiplications and applications*, J. Complexity, 14 (1998), pp. 257–299.

[14] G. F. Italiano, *Finding paths and deleting edges in directed acyclic graphs*, Inform. Process. Lett., 28 (1988), pp. 5–11.

[15] V. King, *Fully dynamic algorithms for maintaining all-pairs shortest paths and transitive closure in digraphs*, in Proceedings of the 40th Annual Symposium on Foundations of Computer Science, New York, 1999, pp. 81–91.

[16] V. King and G. Sagert, *A fully dynamic algorithm for maintaining the transitive closure*, J. Comput. System Sci., 65 (2002), pp. 150–167.

[17] V. King and M. Thorup, *A space saving trick for directed dynamic transitive closure and shortest path algorithms*, in Proceedings of the 7th Annual International Conference, COCOON, Guilin, China, 2001, pp. 269–277.

[18] I. Krommidas and C. D. Zaroliagis, *An experimental study of algorithms for fully dynamic transitive closure*, in Lecture Notes in Comput. Sci. 3669, Springer, Berlin, 2005, pp. 544–555.

[19] J. A. La Poutré and J. van Leeuwen, *Maintenance of transitive closure and transitive reduction of graphs*, in Proc. of the 13th WG, 1987.

[20] L. Roditty, *A faster and simpler fully dynamic transitive closure*, in Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, Baltimore, MD, 2003, pp. 404–412.

[21] L. Roditty and U. Zwick, *Improved dynamic reachability algorithms for directed graphs*, in Proceedings of the 43rd Annual Symposium on Foundations of Computer Science, Vancouver, BC, Canada, 2002, pp. 679–688.

[22] L. Roditty and U. Zwick, *A fully dynamic reachability algorithm for directed graphs with an*

almost linear update time, in Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, 2004, pp. 184–191.

[23] L. Roditty and U. Zwick, *On dynamic shortest paths problems*, in Proceedings of the 12th Annual European Symposium on Algorithms, Bergen, Norway, 2004, pp. 580–591.

[24] P. Sankowski, *Dynamic transitive closure via dynamic matrix inverse (extended abstract)*, in Proceedings of the 45th Annual Symposium on Foundations of Computer Science, Rome, Italy, 2004, pp. 509–517.

[25] M. Sharir, *A strong-connectivity algorithm and its application in data flow analysis*, Comput. Math. Appl., 7 (1981), pp. 67–72.

[26] R. E. Tarjan, *Depth-first search and linear graph algorithms*, SIAM J. Comput., 1 (1972), pp. 146–160.

[27] J. D. Ullman and M. Yannakakis, *High-probability parallel transitive-closure algorithms*, SIAM J. Comput., 20 (1991), pp. 100–125.

[28] U. Zwick, *All pairs shortest paths using bridging sets and rectangular matrix multiplication*, J. ACM, 49 (2002), pp. 289–317.

SIAM J. COMPUT.
Vol. 37, No. 5, pp. 1472–1498

A FASTER, BETTER APPROXIMATION ALGORITHM FOR THE MINIMUM LATENCY PROBLEM*

AARON ARCHER[†], ASAF LEVIN[‡], AND DAVID P. WILLIAMSON[§]

Abstract. We give a 7.18-approximation algorithm for the minimum latency problem that uses only $O(n \log n)$ calls to the prize-collecting Steiner tree (PCST) subroutine of Goemans and Williamson. This improves the previous best algorithms in both performance guarantee and running time. A previous algorithm of Goemans and Kleinberg for the minimum latency problem requires an approximation algorithm for the k-minimum spanning tree (k-MST) problem which is called as a black box for each value of k. Their algorithm can achieve an approximation factor of 10.77 while making $O(n(n + \log C) \log n)$ PCST calls, a factor of 8.98 using $O(n^3(n + \log C) \log n)$ PCST calls, or a factor of $7.18 + \epsilon$ using $n^{O(1/\epsilon)} \log C$ PCST calls, via the k-MST algorithms of Garg, Arya and Ramesh, and Arora and Karakostas, respectively. Here n denotes the number of nodes in the instance, and C is the largest edge cost in the input. In all cases, the running time is dominated by the PCST calls. Since the PCST subroutine can be implemented to run in $O(n^2)$ time, the overall running time of our algorithm is $O(n^3 \log n)$. We also give a faster randomized version of our algorithm that achieves the same approximation guarantee in expectation, but uses only $O(\log^2 n)$ PCST calls, and derandomize it to obtain a deterministic algorithm with factor $7.18 + \epsilon$, using $O(\frac{1}{\epsilon} \log^2 n)$ PCST calls. The basic idea for our improvement is that we do not treat the k-MST algorithm as a black box. This allows us to take advantage of some special situations in which the PCST subroutine delivers a 2-approximate k-MST. We are able to obtain the same approximation ratio that would be given by Goemans and Kleinberg if we had access to 2-approximate k-MSTs for all values of k, even though we have them only for some values of k that we are not able to specify in advance. We also extend our algorithm to a weighted version of the minimum latency problem.

Key words. minimum latency problem, traveling repairman, approximation algorithm, Lagrangian relaxation, prize-collecting Steiner tree

AMS subject classifications. 68W25, 90C27, 90C59

DOI. 10.1137/07068151X

1. Introduction.

Given a metric space with n nodes and a tour starting at some node and visiting all of the others, the latency of node v is defined to be the total distance traveled before reaching v. The *minimum latency problem* (MLP) asks for a tour, starting at a specified root r and visiting all nodes, such that the total latency is minimized. This problem is sometimes called the *traveling repairman problem* or the *deliveryman problem* and has been well studied in both the computer science and the operations research literature. The MLP models routing problems in which one wants to minimize the average time each customer (node) has to wait before being served (visited) rather than the total time to visit all nodes, as in the case of the famous

[*] Received by the editors January 31, 2007; accepted for publication (in revised form) August 29, 2007; published electronically January 30, 2008. An extended abstract of this paper previously appeared in *Proceedings of the* 14*th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003, pp. 88–96. An improved version appeared as Cornell ORIE Technical report 1362, 2003.
http://www.siam.org/journals/sicomp/37-5/68151.html

[†] Algorithms and Optimization Department, AT&T Shannon Research Laboratory, 180 Park Avenue, Florham Park, NJ 07932 (aarcher@research.att.com). Much of this research was performed while visiting the IBM Almaden Research Center, and part was supported by the Fannie and John Hertz Foundation while the author was at Cornell University.

[‡] Department of Statistics, The Hebrew University, Jerusalem 91905, Israel (levinas@mscc.huji.ac.il).

[§] School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY 14853 (dpw@cs.cornell.edu). Most of this research was performed while this author was a Research Staff Member at the IBM Almaden Research Center.

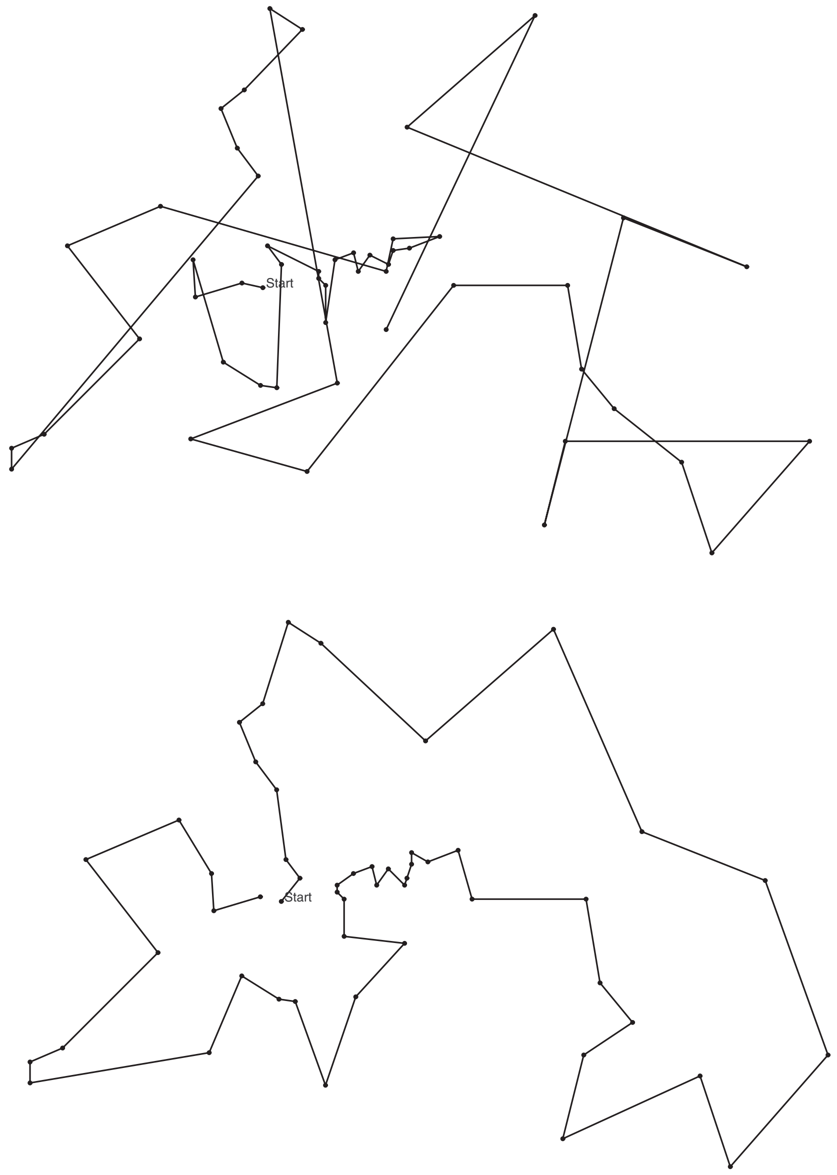


FIG. 1. *Comparison of our algorithm (top) versus the optimal TSP tour (bottom) on the berlin*52 *instance.*

traveling salesman problem (TSP). In this sense, the MLP takes a customer-oriented view, whereas the TSP is server-oriented.

As one illustration of how different the MLP is from the TSP, in Figure 1 we show a comparison of tours for a 2-dimensional Euclidean instance. This is the berlin52 instance, taken from the TSPLIB [33]. At the top we show the MLP tour produced by the algorithm we develop in this paper, and below that we show the optimal TSP tour for this instance. Note that the tour produced by the MLP algorithm attempts to visit most of the nodes near the start node first, before making its way to the more distant nodes. On this instance, the total latency of the optimal TSP tour is 4.48 times as large as that of the tour given by our algorithm.

Koutsoupias, Papadimitriou, and Yannakakis [25] and Ausiello, Leonardi, and Marchetti-Spaccamela [8] motivate the MLP in terms of searching a graph (such as the Web) to find a hidden treasure. If the treasure is equally likely to reside at any node of the graph, then the optimal MLP tour minimizes the expected time to find it. If the location of the treasure is given by a nonuniform probability distribution, the problem reduces to the weighted minimum latency problem that we define below.

The MLP was shown to be NP-hard for general metric spaces by Sahni and Gonzalez [34]. It is also Max-SNP hard, by a reduction from TSP(1,2) (the traveling salesman problem with all distances restricted to be either 1 or 2) [31, 11]. Therefore, there is no polynomial-time approximation scheme for the MLP on general metric spaces unless P = NP. Sitters showed that the problem is NP-hard even for weighted trees [36]. On the positive side, Arora and Karakostas gave quasi-polynomial-time approximation schemes for the MLP on weighted trees and constant dimensional Euclidean spaces [5]. Blum et al. gave the first constant factor approximation algorithm for general metric spaces [11], which was later improved by Goemans and Kleinberg [22]. We elaborate on these results below.

Much work has focused on exact (exponential time) solution approaches to the MLP [35, 16, 10, 28, 42] and on the more general time-dependent traveling salesman problem (TDTSP) [39, 32]. In the TDTSP, the distance between the ith and $(i+1)$st nodes in the traveling salesman tour is multiplied by some weight $w(i)$ in the objective function. The ordinary TSP is the case where all $w(i) = 1$; the MLP is the case where $w(i) = n - i$. The time-dependent orienteering problem (considered in [17]) is dual to the TDTSP—the salesman aims to maximize the number of nodes visited before a given deadline, given that travel times vary as in the TDTSP. Various heuristics for the MLP are evaluated in [38, 40], while [2] analyzes a stochastic version of the problem. In the online variant [15, 26], new nodes appear in the graph as the repairman is traveling. Many authors have considered special cases of the MLP, where the metric is given by an underlying network with some special structure [1, 30, 9, 37, 41]. Fakcharoenphol, Harrelson, and Rao gave a constant factor approximation algorithm for the variant where there are k repairmen who must collectively visit all of the nodes [14].

Because the MLP is NP-hard, we shall consider approximation algorithms for the problem. An *α-approximation algorithm* runs in polynomial time and produces a solution with total latency no more than α times the total latency of a minimum latency tour. The value of α is called the *performance guarantee* of the algorithm.

The first constant factor approximation algorithm for the problem was given by Blum et al. [11]. They also show how to use a β-approximation algorithm for the rooted k-minimum spanning tree (k-MST) problem as a black box and convert it into an 8β-approximation algorithm for the MLP. In the k-MST problem, we are given a graph with costs on the edges and must find the minimum-cost tree spanning at least k nodes. In the *rooted* version, the tree must contain some specified root r.[1] The connection between the k-MST problem and the MLP is that the cost of the optimal k-MST rooted at r is a lower bound on the latency of the kth node visited by the optimal MLP tour. Goemans and Kleinberg (GK) [22] subsequently improved the

[1] Although there has been some confusion in the literature on this point, the rooted and unrooted versions of the k-MST problem can each be reduced to the other as follows. To solve the unrooted version, one can run an algorithm for the rooted version with each of the n possible choices for the root and take the cheapest solution. To solve the rooted version, one can generate an unrooted instance where there are an extra n nodes, each connected to the root by an edge of zero cost, and ask for an $(n+k)$-MST. The reduction works because every tree spanning $n+k$ nodes in the modified instance includes the root.

performance guarantee of the algorithm of Blum et al. to 3.59β. When preliminary versions of our work appeared [4, 3], the best approximation algorithms known for the k-MST problem were a 3-approximation by Garg [19], a 2.5-approximation by Arya and Ramesh [7], and a $(2+\epsilon)$-approximation by Arora and Karakostas [6], yielding MLP guarantees of 10.77, 8.98, and $7.18+\epsilon$, respectively. Each improvement in the approximation guarantee came at the cost of a significant increase in the running time of the algorithm.

In this paper we further explore the connection between the MLP and rooted k-MST problems. We obtain a performance guarantee of 7.18, slightly improving the previous best of $7.18+\epsilon$. Moreover, our algorithm also has a running time that is faster than the GK algorithm using any of the k-MST algorithms above. In each of these algorithms, the running time is dominated by multiple subroutine calls to an algorithm of Goemans and Williamson for the prize-collecting Steiner tree (PCST) problem [21]. The GK algorithm using Garg's 3-approximation as a subroutine requires $O(n(n+\log C)\log n)$ PCST calls, where C is the cost of the most expensive edge. Using Arya and Ramesh it requires $O(mn(n+\log C)\log n)$, if the metric space is the shortest path metric in a graph with m edges. For general metric spaces, the m becomes an n^2. Using Arora and Karakostas it requires $n^{O(\frac{1}{\epsilon})}\log C$ calls. For each of these algorithms, the $\log C$ factor comes from binary searching for a particular parameter, and by using Megiddo's parametric search technique [29] we can obtain a strongly polynomial version where the $\log C$ is replaced by an n^2. Our algorithm requires only $O(n\log n)$ PCST calls, and we also devise a randomized algorithm that achieves the same approximation factor (in expectation) and uses only $O(\log^2 n)$ PCST calls. This randomized algorithm can be derandomized to obtain a deterministic algorithm with factor $7.18+\epsilon$ using only $O(\frac{1}{\epsilon}\log^2 n)$ PCST calls.

Goemans and Williamson showed how to implement their PCST algorithm in $O(n^2\log n)$ time. Later work of Gabow and Pettie [18] improves this to $O(n^2)$. Thus, our deterministic algorithm runs in time $O(n^3\log n)$ overall, our randomized algorithm in time $O(n^2\log^2 n)$, and our deterministic $(7.18+\epsilon)$-approximation in time $O(\frac{1}{\epsilon}n^2\log^2 n)$.

The main idea in achieving our result is that we do not treat the k-MST algorithm as a black box. It can be shown that the PCST algorithm returns k-MSTs of cost no more than twice optimal for some values of k that depend on the instance and cannot be specified by the algorithm [13]. We refer to a tree spanning k nodes with cost no more than α times the optimal k-MST as an *α-approximate* k-MST. If we had 2-approximate k-MSTs for all $k=2,\dots,n$, then we could run the GK algorithm, which uses the costs of the trees to select some subset of them to concatenate into an MLP tour. Our trick is to successfully bluff the GK algorithm. We *pretend* to have trees of all sizes by interpolating the costs of the trees we do have to fill in the tree costs for the missing values of k. We refer to these as "phantom" trees. We then prove that, if the GK algorithm were to be run with both real and phantom trees, it would never choose any of the phantom trees to concatenate, so it never calls our bluff. Since the GK algorithm would never select any of the phantom trees anyway, it suffices to use only the real trees we generated. For the analysis to go through, we must also carefully extend our k-MST lower bounds to the phantom values of k. To do this, we utilize the fact that the PCST problem is a Lagrangian relaxation of the k-MST problem, as observed by Chudak, Roughgarden, and Williamson in [13].

The improvement in our running time derives from two sources. The first is that the k-MST algorithms expend a significant amount of work in producing a near-optimal solution of size *exactly* k. In contrast, the workhorse of our algorithms is a

Table 1
A summary of the running times and approximation ratios of our deterministic and randomized algorithms for the weighted and unweighted MLP. Here $\gamma \approx 3.5912$ is the unique solution to $\gamma \ln \gamma = \gamma + 1$.

MLP version	Approx. ratio	Randomized or deterministic	# PCST calls
Unweighted	2γ	det.	$O(n \log n)$
		rand.	$O(\log^2 n)$
	$2\gamma(1+\epsilon)$	det.	$O(\frac{1}{\epsilon} \log^2 n)$
Weighted	2γ	det.	$O(n \log^2 W)$ or $O(n^3 \log^2 n)$
		rand.	$O(\log^2 W)$ or $O(n^2 \log^2 n)$
	$2\gamma(1+\epsilon)$	det.	$O(\frac{1}{\epsilon} \log^2 W)$ or $O(\frac{1}{\epsilon} n^2 \log^2 n)$

subroutine that generates a pair of near-optimal trees of sizes k_{lo} and k_{hi} sandwiching a given target value k, along with convenient lower bounds on the cost of the optimal k-MST for every $k \in [k_{\text{lo}}, k_{\text{hi}}]$. This sandwiching operation can be done much more rapidly than the k-MST algorithms. Second, the GK algorithm needs to run the k-MST black box $(n-1)$ times, once for each value of k. The Lagrangian relaxation technique we use to generate our k-MST lower bounds allows us to reduce this to $O(\log n)$ sandwiching operations in our randomized algorithm. Our work supplies a nice example where an improved analysis technique has led directly to faster and better algorithms. See Table 1.

While Lagrangian relaxation has a rich history as an effective computational technique (for example, in large scale linear programming), only relatively recently has it also been used as a technique to design and analyze approximation algorithms, starting with [24, 13]. Our work represents another early contribution to this emerging body of research.

Our algorithms also extend to the *weighted* minimum latency problem, previously considered in [5]. In this variant, each node v has an associated positive integer weight w_v, and the goal is to find a tour that minimizes $\sum_v w_v \ell_v$, where ℓ_v is the latency of the node v in the tour. Note that this is equivalent to replacing each node v with a clique of w_v nodes joined by edges of cost zero. Let $W = \sum_v w_v$ (which is always at least n, since each weight is a positive integer). The PCST routine may be trivially modified to run in $O(n^2)$ time (that is, time which depends only on the number of nodes in the original graph and not on W). However, using this reduction, our deterministic algorithm requires $O(W \log W)$ PCST calls, which is only a pseudopolynomial running time. Our randomized algorithm applied to the weighted case makes only $O(\log^2 W)$ PCST calls and hence is weakly polynomial. We show how to alter our randomized algorithm to use $O(n^2 \log^2 n)$ PCST calls, making it strongly polynomial. These algorithms for the weighted MLP have an approximation guarantee of 7.18 in expectation, just like our algorithm for the unweighted version. Each of the algorithms can be derandomized to achieve a deterministic algorithm with the same guarantee, at the expense of blowing up the running time by a factor of n. Alternatively, we can derive a deterministic algorithm with a factor of $7.18 + \epsilon$ while blowing up the running time by a factor of only $\frac{1}{\epsilon}$.

Since the appearance of an extended abstract of this paper [4] and a subsequent improved version [3], Chaudhuri, Godfrey, Rao, and Talwar [12] have given a 3.59-approximation algorithm for the MLP. Instead of k-MSTs, they consider *k-strolls*, which are paths starting at the root and visiting k nodes. Clearly, the length of the shortest k-stroll also gives a lower bound on the latency of the kth node in any tour. They give an algorithm that for some values of k finds a tree spanning k nodes whose cost is no worse than that of the best k-stroll. By invoking our framework above, they are able to reduce the performance guarantee by the factor of 2 that our algorithm incurred by using 2-approximate k-MSTs. Their algorithm uses a slightly modified version of the PCST algorithm as its subroutine but requires guessing the last node in the k-stroll and trying all guesses. Thus, their algorithm requires an extra factor of n in the running time for each tree they generate.

Also after the initial publication of our results, Garg published a 2-approximation algorithm for the k-MST problem [20]. While Garg does not state a running time, the straightforward analysis of his algorithm as he presents it would lead to a running time of $O(\frac{1}{k}mn^2)$ PCST calls. However, we can show that a somewhat different implementation requires only $O(\frac{1}{k}n(n + \log C))$ PCST calls. Using this algorithm as a subroutine for the GK algorithm ties our approximation guarantee but requires $O(n(n + \log C) \log n)$ PCST calls, so our algorithm is still superior.

One might hope that the techniques from [20] and [12] could be combined to obtain a tree spanning k nodes whose cost is no greater than that of the optimal k-stroll, for any value of k chosen by the algorithm rather than just some "lucky" values of k as in [12]. However, as Garg explains in [20], each of these techniques relies on a certain piece of slack in the analysis of the PCST algorithm and each uses up the entire slack, so they cannot be combined. Therefore, even in light of Garg's new result, our techniques are still necessary to obtain the 3.59-approximation algorithm in [12] for the MLP, which is the best approximation ratio achieved to date. Without our techniques, the best approximation ratio that is known would be $3.59 + \epsilon$ from [12], which requires $n^{O(\frac{1}{\epsilon})} \log C$ PCST calls.

The techniques introduced by this paper have also been successfully applied by Hassin and Levin to the minimum latency set cover problem [23] and by Lin, Nagarajan, Rajaraman, and Williamson [27] to develop a framework for designing incremental approximation algorithms. Thus, our techniques have broader applicability than just to the MLP.

The paper is structured as follows. In section 2, we review the main ideas of previous approximation algorithms for the MLP. Section 3 gives our deterministic algorithm, assuming we can generate approximate k-MSTs satisfying certain conditions, and section 4 analyzes the performance guarantee of this algorithm. Section 5 motivates the conditions of section 3 and shows how to satisfy them using something we call our *critical search* primitive, which relies on Lagrangian relaxation. We include a discussion of how one could implement the critical search primitive using Megiddo's parametric search technique [29]. Section 6 shows how to exploit some slack in the previous analysis in order to accelerate the running time, given a slightly different critical search primitive that uses binary search instead of Megiddo's parametric search, although the details of this faster critical search primitive are deferred to section 9. In section 7, we give our faster randomized algorithm, which requires a somewhat different analysis than the deterministic algorithm. We also show how to derandomize it. Section 8 discusses the weighted version of the problem and gives extensions of our deterministic and randomized algorithms to the weighted case. In section 10, we show some experimental results of our algorithm for the unweighted case of MLP. We conclude in section 11 with some thoughts about approaches for further improvements.

2. Intuition and overview.

We now describe the basic ideas behind the Blum et al. [11] and GK [22] algorithms and how our approach departs from them. Both analyses use the cost of the optimal k-MST as a lower bound for the latency of the kth node visited in the optimal MLP tour, and both algorithms start with β-approximate solutions to the k-MST problem rooted at r for $k = 2, 3, \ldots, n$. They then select a subsequence of these trees with geometrically increasing costs and concatenate them to get a solution for the MLP. For the sake of intuition, let us assume throughout this section that the sets of nodes spanned by these trees are nested, which turns out to be the worst case for the analysis.

Without loss of generality, the cost of the k-MSTs increases with k. The Blum et al. algorithm buckets the trees according to their cost—for each integer ℓ, it selects the most expensive tree with cost in $(2^\ell, 2^{\ell+1}]$. It doubles each of the selected trees, shortcuts it to make a cycle rooted at r, and then traverses all of these cycles in order, shortcutting nodes it has already visited. Since the last tree selected spans all of the nodes, so does the resulting MLP tour. They compare the latency of the kth node visited in the tour to the cost of the optimal k-MST. They upper bound the latency of the kth node by the total cost of all of the concatenated cycles up to and including the first one that visits this node. They lose a factor of β because the trees are β-approximate k-MSTs, a factor of 2 from the bucketing ratio, a factor of 2 from doubling the trees to get cycles, and a factor of 2 from the geometric sum. This yields the approximation factor of 8β.

The GK improvement derives from two sources. First, it orients each of the concatenated cycles in the direction that minimizes the total latency of the new nodes visited by that cycle. Second, it applies a random shift to the bucket breakpoints. Using buckets of ratio $\gamma \approx 3.59$ instead of ratio 2, it achieves a performance guarantee of $\gamma\beta$.

Our algorithm departs from these previous ones in that we do not start off with approximate k-MSTs for every value of k. Instead, we obtain $(2 - \frac{1}{n-1})$-approximate n_i-MSTs for some subsequence $n_1 < \cdots < n_\ell$ (where $n_1 = 1$ and $n_\ell = n$) that is not under our control. Let d_k denote the cost of the tree spanning k nodes and b_k denote our lower bound on the optimal k-MST for $k = n_1, \ldots, n_\ell$. We derive these trees using a Lagrangian relaxation technique, which allows us to guarantee that linearly interpolating the b_{n_i} to the missing values of k yields valid lower bounds on the cost of the optimal k-MST. We will obtain our MLP solution by concatenating some subset of these trees, as in Blum et al. and GK.

The GK analysis uses the idea of *modified latency.* Roughly, one can think of the modified latency of node v as the average latency of all of the nodes first visited by the cycle in the concatenation that first visits node v. The total modified latency is an upper bound on the latency of the MLP tour we construct. The GK randomized bucketing procedure yields a solution whose expected total modified latency is at most $\gamma(d_2 + \cdots + d_n) \leq \gamma\beta(OPT_2 + \cdots + OPT_n) \leq \gamma\beta OPT$ (where OPT denotes the value of the optimal MLP tour and OPT_k denotes the optimal k-MST value). Goemans and Kleinberg also observe that one can use a shortest path computation to determine which concatenation of trees minimizes the total modified latency. Since the minimum is no more than the expectation, this yields a deterministic $\gamma\beta$-approximation algorithm. Whereas Goemans and Kleinberg introduce the shortest path calculation merely to derandomize their algorithm, for us the use of the shortest path computation is central to the analysis of our performance guarantee.

3. The algorithm.

Having motivated and outlined the main points of our MLP algorithm, in this section we describe it precisely.

We start by using our tree-generating algorithm of section 5 to produce some set of ℓ trees $T_{n_1}, \ldots, T_{n_\ell}$ rooted at r and spanning $n_1 < \cdots < n_\ell$ nodes, respectively (where $n_1 = 1$ and $n_\ell = n$). For $k = n_1, n_2, \dots, n_\ell$, let d_k denote the cost of tree T_k. Without loss of generality, we assume d_k is increasing with k. Our tree-generating algorithm also establishes lower bounds on OPT_k (the cost of the optimal k-MST rooted at r) and hence on the latency of the kth node visited in the optimal MLP tour for $k = 2, \dots, n$. These lower bounds b_k satisfy the properties:

1. $b_k \le OPT_k$ for $1 \le k \le n$,
2. $d_{n_i} \le \beta b_{n_i}$ for $1 \le i \le \ell$,
3. $b_k = b_{n_{i-1}} + \frac{b_{n_i} - b_{n_{i-1}}}{n_i - n_{i-1}}(k - n_{i-1})$ for $n_{i-1} \le k \le n_i$ and $i = 2, \dots, n$.

That is, b_k is the linear interpolation of $b_{n_{i-1}}$ and b_{n_i}, and each T_{n_i} is a β-approximate n_i-MST. In our algorithm, we can specify any $\beta \ge (2 - \frac{1}{n-1})$. For now, think of β as 2. Later on, we will choose some β in the interval $(2 - \frac{1}{n-1}, 2)$.

Now we use a shortest path calculation described below to select some subcollection of these trees to concatenate. Denote the selected trees by $T_{j_1}, \dots, T_{j_m}$, so $j_1, \dots, j_m$ is the increasing sequence of nodes they span. For each selected tree T_i, double all of the tree edges and traverse an Eulerian tour starting at r, shortcutting nodes already visited, to obtain a cycle $\hat{C}_i$. Now obtain a cycle C_i from $\hat{C}_i$ by shortcutting all nodes (except for r) that are visited by some $\hat{C}_k$, with $k < i$. Because we are in a metric space, this does not increase the length of the cycle. Let S_i denote the set of nodes visited by C_i, excluding the root r. Orient C_i in the direction that minimizes the total latency of the nodes in S_i. To obtain our MLP solution, simply traverse each rooted, oriented cycle $C_{j_1}, \dots, C_{j_m}$ in order, shortcutting the intermediate visits to the root between cycles. Let $C = C_{j_1}, \dots, C_{j_m}$ denote this concatenated tour. Following Goemans and Kleinberg, we define the *modified latency* of the kth node of C to be

$$\pi_k = d_{j_{p(k)}} + 2(d_{j_{p(k)-1}} + \cdots + d_{j_1}), \tag{1}$$

where $p(k)$ is the smallest index such that $k \le j_{p(k)}$. The motivation for this definition is that if the sets of nodes spanned by $T_{j_1}, \dots, T_{j_m}$ are nested, then π_k is an upper bound on the average latency of the nodes first visited by cycle $C_{j_{p(k)}}$. Indeed, in section 4, we repeat an argument of Goemans and Kleinberg that in all cases $\pi_2 + \cdots + \pi_n$ is an upper bound on the total latency of C.

We can now describe the shortest path computation whose solution identifies a tree concatenation with minimum total modified latency. We construct a graph G with nodes $n_1, \dots, n_\ell$ and arcs $i \to k$ for each $i < k$. A path $j_1 \to \cdots \to j_m$ corresponds to selecting trees $T_{j_1}, \dots, T_{j_m}$. Thus, the cost on arc $i \to k$ is

$$(k - i)d_k + 2(n - k)d_k = 2d_k\left(n - \frac{i + k}{2}\right), \tag{2}$$

which corresponds to the contribution made to the total modified latency by traversing tree T_k immediately after traversing T_i. This is because tree T_k contributes d_k to the modified latencies of the $(k - i)$ new nodes it visits and $2d_k$ to each of the remaining $(n - k)$ unvisited nodes. If the shortest path from 1 to n in this graph goes $j_1 \to \cdots \to j_m$, then we select trees $T_{j_1}, \dots, T_{j_m}$ to concatenate.

4. Analyzing the approximation ratio. The analysis of our algorithm proceeds in three steps. First, we demonstrate that $\pi_2 + \cdots + \pi_n$ really is an upper bound on the latency of the tour we construct, even if the trees are not nested. Next, we appeal to a result of Goemans and Kleinberg that upper bounds the total modified latency of the tour given by the shortest path computation in terms of the tree costs, in the event that we have trees of all sizes, $2, \dots, n$. Finally, we show that, if we were to run this shortest path computation with our real trees and some "phantom" interpolated trees, the computation would never select any of the phantom trees. Therefore, we achieve the same performance guarantee that GK would achieve if we actually did have access to the phantom trees.

For completeness, we begin by repeating an argument of Goemans and Kleinberg showing that the modified latencies do upper bound the latency of the concatenated tour.

LEMMA 4.1 (see [22]). *The total latency of the MLP tour obtained by concatenating trees $T_{j_1}, \dots, T_{j_m}$ (where $j_1 = 1$, $j_m = n$) is at most $\pi_2 + \cdots + \pi_n$, where the π_k are given by* (1).

Proof. The following argument essentially says that the worst case for our analysis is when the sets of nodes spanned by the selected trees $T_{j_1}, \dots, T_{j_m}$ are nested. Let us consider the latency of the kth node we visit in the concatenated tour C, where r is considered to be the first node, whose latency is zero. If the kth node in C was encountered as part of cycle C_{j_p}, then we can upper bound its latency by the sum of the costs of cycles $C_{j_1}, \dots, C_{j_{p-1}}$ plus the portion of cycle C_{j_p} that is traversed prior to reaching this node. Since we traverse cycle C_{j_p} in the direction that minimizes the total latency of the new nodes S_{j_p}, the average contribution of this cycle to the latencies of the nodes in S_{j_p} is at most half the cost of the cycle. To see this, notice that, for any node $i \in S_{j_p}$, traversing C_{j_p} in one direction contributes some amount x to the latency of i, and traversing C_{j_p} in the other direction contributes $\mathrm{cost}(C_{j_p}) - x$, so on average it contributes $\mathrm{cost}(C_{j_p})/2$. The cost of C_i is at most $2d_i$, by the triangle inequality. Therefore, the average latency of the nodes in S_{j_p} is at most $2(d_{j_1} + \cdots + d_{j_{p-1}}) + d_{j_p}$, so the total latency of C is at most

$$\sum_{p=1}^{m} |S_{j_p}|(2(d_{j_1} + \cdots + d_{j_{p-1}}) + d_{j_p}). \tag{3}$$

Since $\sum |S_{j_p}| = n$, we can view this as a weighted sum. Clearly, the worst case for this analysis is when the sets of nodes spanned by the trees $T_{j_1}, \dots, T_{j_m}$ are nested, since this puts the greatest weight on the larger terms in (3). In this worst case, our upper bound on the average latency of the $(j_{p-1}+1)$th through j_pth nodes in C becomes $2(d_{j_1} + \cdots + d_{j_{p-1}}) + d_{j_p}$. Thus since $\pi_k = d_{j_{p(k)}} + 2(d_{j_{p(k)-1}} + \cdots + d_{j_1})$, where $p(k)$ is the smallest index such that $k \le j_{p(k)}$, our tour has latency at most $\pi_2 + \cdots + \pi_n$. □

The advantage of the upper bound $\pi_2 + \cdots + \pi_n$ is that it depends only on the costs of the selected trees and the number of nodes they span, not on the structure of the trees. We now state the main theorem of the Goemans and Kleinberg paper.

DEFINITION 4.2. *The number γ denotes the unique solution of $\gamma \ln \gamma = \gamma + 1$, which is approximately* 3.5912.

THEOREM 4.3 (see [22]). *Given $d_2, \dots, d_n \ge 0$, let G be the graph on nodes $1, \dots, n$ including all arcs $i \to k$ for $i < k$, with arc lengths given by* (2)*. Then the shortest path in G from node* 1 *to node n has length at most $\gamma(d_2 + \cdots + d_n)$.*

Recall that our tree-generating procedure of section 5 returns trees of sizes $n_1, \dots, n_\ell$ and costs $d_{n_1}, \dots, d_{n_\ell}$. It also establishes lower bounds b_k on the cost OPT_k of the optimal k-MST for every k, and these bounds satisfy properties 1–3 from section 3. Let us linearly interpolate the tree costs d_{n_i} to the missing values of k. That is, set

$$d_k = d_{n_{i-1}} + \frac{d_{n_i} - d_{n_{i-1}}}{n_i - n_{i-1}}(k - n_{i-1}) \tag{4}$$

for $n_{i-1} \leq k \leq n_i$ and $i = 2, \ldots, n$. Then clearly $d_k \leq \beta b_k$ for all k – it is true for the end points of each interval, so it is true for the linear interpolations. That is, if we actually had a tree of cost d_k spanning k nodes, it would be a β-approximate k-MST, and therefore we could use the GK algorithm to generate a $\beta\gamma$-approximate MLP tour. Unfortunately, we are missing these interpolated trees, so we must find a way to get around this difficulty.

Notice that the GK shortest path computation can be defined independently of trees and MLP tours—it just requires a set of ordered pairs to define the graph G in which the shortest path is computed.

Definition 4.4. *A* tree signature *is an ordered pair* (k, c)*, where* k *is its* size *and* c *is its* cost*. If we denote this tree signature by* T*, then we define* $|T| := k$ *and* $c(T) := c$*. Analogously, if* T *is an actual tree, let* $|T|$ *denote the number of nodes spanned by it and* $c(T)$ *denote the total cost of its edges* $\sum_{e \in T} c_e$*.*

Obviously, a real tree T has an associated tree signature $(|T|, c(T))$. We will find it convenient to consider the behavior of the GK shortest path computation on graphs generated from tree signatures that may or may not correspond to actual trees. When we talk about concatenating a sequence of tree signatures to get a "tour," we just mean to define the kth modified latency π_k with respect to such a concatenation using (1), exactly the same as if these tree signatures represented real trees. Thus, tree signatures, concatenations of tree signatures, and modified latencies are just a bookkeeping mechanism, with the property that if the tree signatures in a concatenation correspond to actual trees, then these trees can be concatenated into an actual MLP tour with latency at most $\pi_2 + \cdots + \pi_n$. When we use a tree signature that may or may not correspond to an actual tree, we may think of it as corresponding to a "phantom tree" that exists only in our minds. Unlike *tree signature*, *phantom tree* is not a technical term but rather just a mnemonic to signify that we *wish* we had a real tree with a particular signature.

In particular, we wish we had a full set of trees with the signatures (k, d_k), $k = 2, \ldots, n$, because this would immediately yield a $\beta\gamma$-approximate MLP tour. Why? Suppose we were to run the GK shortest path computation using the full set of costs $d_2, \ldots, d_n$, i.e., using both the real trees and the phantom trees. Then by Theorem 4.3, the modified latency of the resulting solution would be at most

$$\begin{aligned}\gamma(d_2 + \cdots + d_n) &\leq \beta\gamma(b_2 + \cdots + b_n)\\ &\leq \beta\gamma(OPT_2 + \cdots + OPT_n)\\ &\leq \beta\gamma OPT,\end{aligned}$$

where OPT denotes the optimal MLP value. The difficulty is that the shortest path computation might select one of the phantom trees, in which case we cannot actually construct the MLP tour. Fortunately, we will show that this never occurs, because the shortest path will go only through "corner points."

Definition 4.5. *With respect to a set of tree signatures* $(1, 0), (2, c_2), \ldots, (n, c_n)$*, the signature* (k, c_k) *is a* corner point *if* $k \in \{1, n\}$ *or* $k \in \{2, \ldots, n-1\}$ *and* $c_k \neq \frac{1}{2}(c_{k-1} + c_{k+1})$*.*

The corner points divide up the interval $\{1, \ldots, n\}$ such that the tree signatures between any two consecutive corner points are just linear interpolations of those two corner points.

Theorem 4.6. *The Goemans–Kleinberg shortest path computation in the graph generated from a set of tree signatures* $(1, 0), (2, c_2), \ldots, (n, c_n)$ *visits only corner points (with respect to these signatures).*

Proof. Just to eliminate special cases that we would otherwise have to consider, let us introduce self-loops at every node in our graph, with costs still given by (2). In any case in our argument where we end up using a self-loop, we can obtain an even shorter path by removing it.

Suppose on the contrary that a shortest path visits $i \to j \to k$, where j is not a corner point. Let n_{lo} and n_{hi} be the sizes of the two nearest corner points that sandwich j, where $n_{\text{lo}} < j < n_{\text{hi}}$. Set $\lambda = (c_{n_{\text{hi}}} - c_{n_{\text{lo}}})/(n_{\text{hi}} - n_{\text{lo}})$; i.e., λ is the slope of the line connecting these two corner points. If $\lambda \leq 0$, we can decrease the modified latencies $\pi_{i+1}, \ldots, \pi_n$ by visiting $j+1$ instead of j. (This corresponds to replacing a tree in the concatenation by a larger one of lesser cost.) Thus, we know $\lambda > 0$.

Treating j as a variable now, we show that we can obtain a strictly shorter path by setting j to either $\max(i, n_{\text{lo}})$ or $\min(k, n_{\text{hi}})$, arriving at a contradiction. For j in the interval between the sandwiching corner points, the tree signature (j, c_j) is just the linear interpolation of these corner points, i.e., $c_j = c_{n_{\text{lo}}} + \lambda(j - n_{\text{lo}})$. Thus, by definition of the arc lengths (2), the subpath from $i \to j \to k$ costs

$$(5) \qquad 2(c_{n_{\text{lo}}} + \lambda(j - n_{\text{lo}}))\left(n - \frac{i+j}{2}\right) + 2c_k\left(n - \frac{j+k}{2}\right).$$

This cost is valid for $\max(i, n_{\text{lo}}) \leq j \leq \min(k, n_{\text{hi}})$ and is a quadratic function of j, where the coefficient on the j^2 term is $-\lambda$. Thus, the cost is strictly concave in j, so it attains a strict minimum at one of the end points $\max(i, n_{\text{lo}})$ or $\min(k, n_{\text{hi}})$. This is a contradiction, because we already started with a shortest path. □

We chose phantom trees that were linear interpolations of the real trees we had. Thus, all of the corner points in our set of tree signatures correspond to real trees. Since a shortest path in the graph with the phantom trees included never actually uses any of the phantom trees, we might as well run the shortest path computation using just the actual trees, as in the algorithm description of section 3. Putting Theorem 4.6 together with the discussion preceding it yields our main result. Recall that $\beta < 2$.

THEOREM 4.7. *The algorithm described in section* 3 *yields an MLP tour of cost at most* $\beta\gamma OPT$.

5. Generating trees and lower bounds via Lagrangian relaxation.

In this section, we discuss how to use Lagrangian relaxation to obtain trees and lower bounds satisfying properties 1–3 of section 3.

Lagrangian relaxation is a technique for generating lower bounds on the optimal solution of one minimization problem by relaxing some of the constraints to generate a related but simpler minimization problem, which ideally should be easier to solve. We discuss here how the PCST problem serves as a Lagrangian relaxation of the k-MST problem, a relationship first explored by [13].

Given a graph with a cost c_e on each edge e, a root node r, and a penalty $p_v \geq 0$ for each node v, the PCST problem asks for a tree T rooted at r that minimizes the quantity

$$\sum_{e \in T} c_e + \sum_{v \notin T} p_v.$$

In other words, we ask for a tree that minimizes the sum of the edge costs in the tree plus the penalties of the nodes not spanned by the tree. Throughout this paper, we use PCST to refer to the PCST problem with *uniform penalties*, where all of the

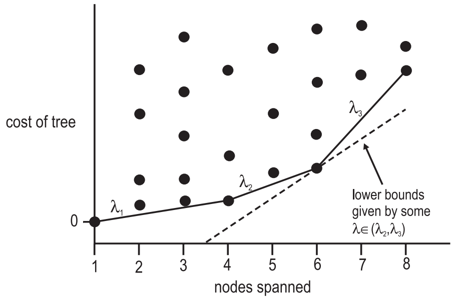


FIG. 2. *The dots represent a signature* $(|T|, c(T))$ *for each possible tree* T *rooted at* r. *The lowest dot in column* k *represents the optimal* k*-MST, while the lines forming the lower convex hull of the dots represent the best lower bounds on* OPT_k *that could be derived via our Lagrangian relaxation.*

penalties are set to the same value $\lambda \geq 0$. In this case, the objective function becomes

$$\sum_{e \in T} c_e + \lambda(n - |T|).$$

We now consider how the PCST problem can generate lower bounds for the k-MST problem. Let $T^*(\lambda)$ be an optimal solution to the PCST problem when the penalty parameter is set to λ. Since all trees spanning $|T^*(\lambda)|$ nodes incur the same penalty term, $T^*(\lambda)$ must be the optimal $|T^*(\lambda)|$-MST. Moreover, since the optimal k-MST is one feasible solution to the PCST problem, we have $c(T^*(\lambda)) + \lambda(n - |T^*(\lambda)|) \leq OPT_k + \lambda(n-k)$. Rearranging gives

$$c(T^*(\lambda)) + \lambda(k - |T^*(\lambda)|) \leq OPT_k. \tag{6}$$

Notice that solving a *single* instance of PCST generates k-MST lower bounds for *all* values of k simultaneously. Since the PCST problem is also NP-hard, this does not help us directly in generating trees and k-MST lower bounds. However, we will continue with this thought experiment, which will guide us in how we use the Goemans–Williamson approximation algorithm for PCST.

The following graphical interpretation of this Lagrangian relaxation is revealing. Consider a plot of the signatures $(|T|, c(T))$ for all possible trees T rooted at r (see Figure 2). Let $\mathcal{T}$ denote this set of points. Fixing the penalty parameter $\lambda \geq 0$, consider a line of slope λ and slide this line up vertically until the first time it hits one of the points in $\mathcal{T}$. This is the point corresponding to $T^*(\lambda)$, and the line $\mathcal{L}(\lambda)$ of slope λ passing through it represents the k-MST lower bounds generated for all values of k. Thus, if we consider the best lower bounds that can be generated by solving PCST for all possible values of λ, this just yields the lower convex hull of the point set $\mathcal{T}$. Let $P_0, \ldots, P_\ell$ denote the sequence of tree signatures along the lower convex hull of $\mathcal{T}$ starting with $P_0 = (1, 0)$ and t_i denote the number of nodes spanned by the tree corresponding to P_i. Let $\lambda_0 = 0$, and for $i = 1, \ldots, \ell$ let λ_i be the slope of the line connecting P_{i-1} to P_i. Consider how the line $\mathcal{L}(\lambda)$ moves as we change λ. As λ increases from 0 to λ_1, the line starts out horizontally and pivots around the point P_0 until it hits P_1. In general, as λ increases from λ_i to λ_{i+1}, the line $\mathcal{L}(\lambda)$ rotates around point P_i until it hits point P_{i+1}. Now consider how the k-MST lower bound changes with λ, for various values of k. If $k = t_i$, then the lower bound on OPT_k strictly increases for $\lambda \in [0, \lambda_i)$, stays constant on $[\lambda_i, \lambda_{i+1}]$, and then decreases on (λ_{i+1}, ∞).

If $k \in (t_{i-1}, t_i)$, then the lower bound increases on $[0, \lambda_i)$, attains its maximum at λ_i, and decreases on (λ_i, ∞). We refer to the values λ_i $(i = 1, \ldots, \ell)$, where the size of the optimal tree changes, as *critical values* with respect to the optimal algorithm.

In general, suppose $\mathcal{A}$ is any PCST algorithm such that the tree generated (as a function of λ) changes only at a finite number of breakpoints and remains constant on the intervals between consecutive breakpoints. Then we say that $\mathcal{A}$ has *critical values* and refer to those breakpoints as *critical values with respect to the algorithm* $\mathcal{A}$. If the algorithm is understood, we will just call them critical values.

Notice that, if we want to generate the best lower bounds for all values of k and also the trees corresponding to the points P_i, it is sufficient to solve the PCST problem just for the critical values of λ. The critical value λ_i generates the best k-MST lower bounds for k in the interval $[t_{i-1}, t_i]$. By using all of the critical values, this collection of intervals covers all values of k in $[1, n]$. Notice that this collection of trees and lower bounds would satisfy conditions 1–3 of section 3, with $\beta = 1$.

Since the PCST problem is NP-hard, we cannot expect to solve it to optimality in polynomial time. Fortunately, Goemans and Williamson [21] gave a $(2 - \frac{1}{n-1})$-approximation algorithm that also has the *Lagrangian multiplier preserving* property. This means that running the algorithm with penalty parameter λ returns a tree $T(\lambda)$ and a lower bound $LB(\lambda)$ such that

$$c(T(\lambda)) \leq \left(2 - \frac{1}{n-1}\right) LB(\lambda) \tag{7}$$

and

$$OPT_k \geq LB(\lambda) + \lambda(k - |T(\lambda)|). \tag{8}$$

In other words, the tree generated is a $(2 - \frac{1}{n-1})$-approximate $|T(\lambda)|$-MST, and the line $\mathcal{L}(\lambda)$ of slope λ through $(|T(\lambda)|, LB(\lambda))$ gives valid k-MST lower bounds for all values of k. We will prove these inequalities in Lemma 9.2 of section 9.

How do these lower bounds change as we vary λ? The inner workings of the Goemans–Williamson PCST algorithm are such that the algorithm has critical values.[2] At a critical value of λ, we can generate the tree corresponding to one bordering interval or the other depending on how we break ties in the algorithm, which does not affect the line of lower bounds $\mathcal{L}(\lambda)$. Thus, if the two trees are T_{lo} and T_{hi} with corresponding lower bounds $LB_{T_{\text{lo}}}$ and $LB_{T_{\text{hi}}}$ (where $|T_{\text{lo}}| < |T_{\text{hi}}|$), then the interpolated k-MST lower bounds for $|T_{\text{lo}}| < k < |T_{\text{hi}}|$ coincide with the line $\mathcal{L}(\lambda)$ and are hence valid lower bounds. We say that this critical value of λ *covers the size interval* $[|T_{\text{lo}}|, |T_{\text{hi}}|]$ and *covers the bound interval* $[LB_{T_{\text{lo}}}, LB_{T_{\text{hi}}}]$. We refer to both the size interval and the bound interval as *critical intervals.*

Suppose that we can find a collection of critical values of λ such that the union of the size intervals they cover contains $\{1, \ldots, n\}$. Each critical value generates a line of lower bounds for the size interval that it covers. Let $\mathcal{T}$ denote the set of lower bound tree signatures generated by all of these critical values. If we take the lower convex hull of $\mathcal{T}$, the interpolated lower bounds would be at most the lower bounds generated by the covering intervals, and hence still valid (see Figure 3). Thus, the desired properties 1–3 of section 3 would hold, with $\beta = 2 - \frac{1}{n-1}$.

How can we find the required critical values of λ? Let $c_{\max}$ denote the distance from the root to the farthest node. Because of the inner workings of the PCST

[2] Unlike in the case of $T^*(\lambda)$, on the intervals where $T(\lambda)$ is constant, $LB(\lambda)$ may not be constant, and $|T(\lambda)|$ is not increasing in λ. However, this does not cause a problem.

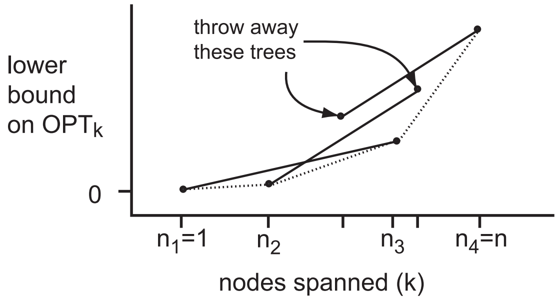


FIG. 3. *Each dot represents a (size of tree, lower bound) pair returned by PCST, and the solid lines are the interpolated lower bounds from Lemma* 9.2. *The lower envelope (dotted line) is still clearly a valid lower bound at each point, since it is below the solid lines. We keep only the trees whose dots are on the lower envelope.*

algorithm, it is a fact that $T(c_{\max})$ spans all n nodes, while $T(0)$ spans only the root. For any arbitrary target value $k \in \{1, \dots, n\}$, there is at least one critical value λ^* that covers a size interval containing k. Moreover, if we maintain an interval $[\lambda_{\text{lo}}, \lambda_{\text{hi}}]$ such that $|T(\lambda_{\text{lo}})| \le k \le |T(\lambda_{\text{hi}})|$, we know that there is some solution $\lambda^* \in [\lambda_{\text{lo}}, \lambda_{\text{hi}}]$. We can initialize the interval to $[0, c_{\max}]$, test some value λ in the interval, and update the appropriate end point to maintain the invariant. It is a fact that all of the decisions in the PCST algorithm involve comparing quantities that are linear in λ, and therefore we can apply the parametric search technique of Megiddo to calculate λ^* exactly [29]. This requires running the PCST algorithm once for each comparison made in the PCST algorithm. Using the $O(n^2)$ implementation of Gabow and Pettie [18], this means that we can compute a single critical value exactly using $O(n^2)$ PCST calls. Since we need to compute at most $(n-1)$ critical values to cover $[1, n]$, we obtain the following theorem.

THEOREM 5.1. *There is a* $(2 - \frac{1}{n-1})\gamma$*-approximation algorithm for the minimum latency problem that runs in time dominated by* $O(n^3)$ *PCST calls.*

Aside from justifying (7) and (8) (which we do in section 9), this ends the discussion of our basic algorithm for the MLP problem. In section 6, we discuss how to speed up this running time by allowing some small error in our computation of a critical value λ^*, and, in section 7, we discuss how to further speed up the algorithm by using randomization to reduce the number of critical values we need to calculate. Before we move on to this, let us describe a slight variant of our algorithm.

In section 3, we discussed creating tree signatures by interpolating the sizes and costs of pairs of real trees. We now describe a slightly different algorithm, based on creating tree signatures from the interpolated lower bounds rather than interpolating real trees. This alternate view will be useful in section 7 when we discuss our randomized algorithm.

Let $b_1, \dots, b_n$ be the lower bounds given by the lower convex hull of $\mathcal{T}$, described above. Suppose we construct the graph G of section 3 using the tree signatures (k, b_k), $k = 1, \dots, n$, and compute the shortest path to select which tree signatures to concatenate. By Theorem 4.3, the shortest path will have modified latency at most $\gamma(b_2 + \cdots + b_n) \le \gamma OPT$, and, by Theorem 4.6, it will use only corner points. Since every corner point is a tree signature corresponding to the end point of an interval we generated, we may replace these phantom trees with real trees, blowing up the tree costs and hence the modified latencies by at most a factor of β. Thus, our tour has total latency at most $\beta\gamma OPT$.

6. Accelerating the running time. The goal of this section is to reduce the number of PCST calls required by our algorithm to $O(n \log n)$ while achieving an approximation ratio of 2γ.

Because the PCST algorithm gives an approximation factor of $(2 - \frac{1}{n-1})$ instead of 2, this allows us a little bit of slack if we just want a 2γ-approximation algorithm for the MLP problem. We can use this slack to significantly speed up our algorithm. Instead of using Megiddo's parametric search technique to calculate a critical value λ^* exactly, we can instead use binary search to find a very small interval $[\lambda_{\text{lo}}, \lambda_{\text{hi}}]$ that contains λ^*. If the interval is small enough, then the lines $\mathcal{L}(\lambda_{\text{lo}})$ and $\mathcal{L}(\lambda_{\text{hi}})$ are close enough together that we can replace them with a single line that gives valid lower bounds b_k on the interval $[|T(\lambda_{\text{lo}})|, |T(\lambda_{\text{hi}})|]$ without degrading the lower bounds by much.

By a *probe*, we mean a single PCST call to narrow the interval $[\lambda_{\text{lo}}, \lambda_{\text{hi}}]$. By a *critical search*, we mean a series of probes used to narrow the interval $[\lambda_{\text{lo}}, \lambda_{\text{hi}}]$ to the desired size by binary search, so that the following three conditions hold:

1. $b_k \le OPT_k$ for $k_{\text{lo}} \le k \le k_{\text{hi}}$,
2. $c(T(\lambda)) \le \beta b_{|T(\lambda)|}$ for $\lambda \in \{\lambda_{\text{lo}}, \lambda_{\text{hi}}\}$,
3. $b_k = b_{k_{\text{lo}}} + \frac{b_{k_{\text{hi}}} - b_{k_{\text{lo}}}}{k_{\text{hi}} - k_{\text{lo}}}(k - k_{\text{lo}})$,

where $k_{\text{lo}} = |T(\lambda_{\text{lo}})|$ and $k_{\text{hi}} = |T(\lambda_{\text{hi}})|$. The bounds $b_{k_{\text{lo}}}$ and $b_{k_{\text{hi}}}$ will take the form $\delta \text{LB}(\lambda_{\text{lo}})$ and $\delta \text{LB}(\lambda_{\text{hi}})$, for some δ less than but close to 1 that we can specify in the algorithm, and $\beta = \frac{1}{\delta}(2 - \frac{1}{n-1})$.

Let $c_{\min}$ denote the distance from the root to the closest other node. In Corollary 9.5 of section 9, we show that $O(\log \frac{n c_{\max}}{c_{\min}})$ probes are sufficient to perform a critical search with $\beta = (2 - \frac{1}{2n})$. We now show how to reduce the problem to the case where $\frac{c_{\max}}{c_{\min}} = \text{poly}(n)$, so each critical search takes only $O(\log n)$ probes. Since we require $O(n)$ critical searches, our overall algorithm takes at most $O(n \log n)$ PCST calls, as advertised.

How do we ensure $\frac{c_{\max}}{c_{\min}} = \text{poly}(n)$? We first delete every node that is within distance $\frac{c_{\max}}{4n^3}$ of the root (aside from the root itself). Clearly, this did not increase the cost of an optimal solution. We are left with an instance where $\frac{c_{\max}}{c_{\min}} \le 4n^3$, and we apply our main algorithm to this instance, with $\beta = (2 - \frac{1}{2n})$. This yields a tour of latency at most $(2 - \frac{1}{2n})\gamma OPT$. To obtain a tour for the original instance, we first visit the deleted nodes in arbitrary order, return to the root, and then visit the remaining nodes in the order given by our main algorithm. By the triangle inequality, the total length of the initial partial tour is at most $\frac{c_{\max}}{2n^2}$. By visiting all of these points first, it contributes at most an extra $\frac{c_{\max}}{2n}$ to the total latency of the entire tour. Since $c_{\max} \le OPT_n \le OPT$, the total latency of our tour is at most $(2 - \frac{1}{2n})\gamma OPT + \frac{c_{\max}}{2n} < 2\gamma OPT$. This yields the following theorem.

THEOREM 6.1. *The algorithm described above is a 2γ-approximation algorithm for the minimum latency problem and runs in time dominated by $O(n \log n)$ PCST calls.*

7. A faster randomized algorithm. In this section, we describe a randomized algorithm that achieves the same approximation ratio of 2γ (in expectation) while reducing the number of PCST calls to $O(\log^2 n)$. We also show how to derandomize it to achieve a deterministic algorithm with the improved running time but a slightly worse approximation guarantee.

In the paper of Goemans and Kleinberg [22], Theorem 4.3 is proven via the probabilistic method. They actually construct a random path in their graph and show that the expected modified latency of their path is at most $\gamma(d_2 + \cdots + d_n)$.

Therefore, the shortest path is at least this short, which yields Theorem 4.3. Thus, instead of Theorem 4.3, a more fundamental statement of their main result is the following.

THEOREM 7.1 (see [22]). *Given $d_2, \dots, d_n \geq 0$, consider the graph G on nodes $1, \dots, n$ including all arcs $i \to k$ for $i < k$, with arc lengths given by* (2). *Let the random variable U be drawn uniformly from* $[0, 1)$, *let $L_0 = \gamma^U$, and for each integer i define $L_i = L_0\gamma^i$. For each i, mark the largest node j such that $d_j \leq L_i$. Then the expected length of the path from 1 through all of the marked nodes (in order) to n is at most $\gamma(d_2 + \cdots + d_n)$.*

We include a proof here, since we will need to generalize this result for our purposes.

Proof. Recall that the path in G represents a concatenation of tree signatures, and the length of the path equals the sum of the modified latencies $\pi_2 + \cdots + \pi_n$. Fix $k \in \{2, \dots, n\}$. Let $\mathcal{S} = \{L_0\gamma^i : i \in \mathbb{Z}\}$ be the set of thresholds, and let $L = \min\{L' \in \mathcal{S} : L' \geq d_k\}$. Then L has the same distribution as $d_k\gamma^U$. Moreover, the first tree signature in the concatenation with size at least k has cost at most L, and the smaller tree signatures have costs at most $L\gamma^{-i}$ for $i = 1, 2, \dots$. Thus,

$$\pi_k \leq L + 2L(\gamma^{-1} + \gamma^{-2} + \cdots) = \frac{\gamma + 1}{\gamma - 1}L,$$

$$E[\pi_k] \leq \frac{\gamma + 1}{\gamma - 1}E[d_k\gamma^U] = \frac{\gamma + 1}{\ln\gamma}d_k = \gamma d_k.$$

The last equality comes because γ satisfies the identity $\gamma \ln\gamma = \gamma + 1$ (from Definition 4.2). Summing over k, the theorem follows by linearity of expectation. □

This result generalizes readily to the following.

THEOREM 7.2. *Given $d_2, \dots, d_n \geq 0$, select random thresholds as in Theorem* 7.1. *For each threshold L, select one arbitrary tree signature $T(L)$ such that $|T(L)| \geq \max\{k : d_k \leq L\}$ and $c(T(L)) \leq L$. The concatenation of these tree signatures has total modified latency at most $\gamma(d_2 + \cdots + d_n)$, in expectation.*

Proof. Fix $k \in \{2, \dots, n\}$, and let L be the smallest threshold such that $d_k \leq L$. By assumption, $|T(L)| \geq k$. Then by the time the concatenation includes tree signature $T(L)$, it has visited at least k nodes, and the sum of the costs of all previous tree signatures in the concatenation is at most $L(\gamma^{-1} + \gamma^{-2} + \cdots)$. Thus, $\pi_k \leq L + 2L(\gamma^{-1} + \gamma^{-2} + \cdots)$, and the rest follows as in the proof of Theorem 7.1. □

We will use this theorem to analyze the following randomized algorithm, which reduces the number of critical searches to $O(\log n)$ down from the $O(n)$ of section 6.

Our algorithm starts by generating random thresholds, as in Theorem 7.2. For each threshold L in the range $[c_{\min}, nc_{\max}]$, we perform a critical search with $\beta = 2 - \frac{1}{2n}$ to cover a bound range that includes L. Each of these $O(\log \frac{nc_{\max}}{c_{\min}})$ critical searches generates two actual trees. We use this collection of trees to create the graph G of section 3 and compute the shortest path to determine which of these trees to concatenate. In order to limit the number of critical searches to $O(\log n)$, we preprocess the original input by deleting every node that is within distance $\frac{c_{\max}}{8n^3}$ of the root (aside from the root itself). We visit all of these nodes first in arbitrary order and then follow the tour given by the tree concatenation. The reason that we consider only thresholds in the range $[c_{\min}, nc_{\max}]$ is that $c_{\min} \leq OPT_2$ and $OPT_n \leq nc_{\max}$, so these thresholds are the only relevant ones.

The tree signatures $T(L)$ mentioned in Theorem 7.2 are used only for the analysis and not for the actual algorithm. They will be derived from the critical searches.

THEOREM 7.3. *The randomized algorithm described above yields a 2γ-approximate solution to the minimum latency problem in expectation and runs in time dominated by $O(\log^2 n)$ PCST calls.*

Proof. The number of thresholds we must consider is $O(\log \frac{nc_{\max}}{c_{\min}}) = O(\log n)$, since we preprocessed the input to ensure $\frac{c_{\max}}{c_{\min}} = \text{poly}(n)$. For each of these, we perform one critical search with $\beta = 2 - \frac{1}{2n}$, each of which requires only $O(\log \frac{nc_{\max}}{c_{\min}}) = O(\log n)$ PCST calls, as we will show in Corollary 9.5. Thus, the total number of PCST calls is $O(\log^2 n)$.

Let us analyze the approximation ratio we achieve on the amended instance. For each threshold L, select one tree signature as follows. Our critical search for L covers some size interval $[k_{\text{lo}}, k_{\text{hi}}]$ and some bound interval $[b_{k_{\text{lo}}}, b_{k_{\text{hi}}}]$ containing L. Thus, there is some $k \in \{k_{\text{lo}}, k_{\text{lo}}+1, \dots, k_{\text{hi}}-1\}$ such that the interpolated bounds satisfy $b_k \le L < b_{k+1}$. Let $T(L)$ be a tree signature of size k and cost b_k. Because b_{k+1} is a valid lower bound on OPT_{k+1}, we get $\max\{j : OPT_j \le L\} \le k = |T(L)|$. Thus, this choice of tree signatures satisfies the hypotheses of Theorem 7.2. Let π denote the total modified latency given by concatenating this collection of tree signatures. Theorem 7.2 gives $E[\pi] \le \gamma(OPT_2 + \cdots + OPT_k) \le \gamma OPT$. Now we need to understand how π compares to the cost of the MLP solution we generated.

Recall that each critical search covers some size interval $[k_{\text{lo}}, k_{\text{hi}}]$ and generates a set of lower bounds on OPT_k for k in this size interval. We can think of each of these lower bounds as a tree signature. Consider taking the lower convex hull of all of these tree signatures generated by all of our critical searches, as in Figure 3. For each tree signature $T(L)$ chosen in the last paragraph, replace it with a tree signature of the same size but cost equal to the corresponding point on the lower convex hull. This can only decrease the total modified latency. By Theorem 4.6, we can further improve the modified latency by selecting the best concatenation using only tree signatures corresponding to the corners of the convex hull. But these tree signatures correspond to end points of the size intervals covered by the critical searches, and therefore we can replace these phantom trees with real trees of cost at most β times as much, which were generated by our critical searches. This increases the modified latency to at most $\beta\pi$. But the modified latency of the concatenation we selected is no larger than this, since we selected the best concatenation from a (perhaps) larger set of trees corresponding to all end points of size intervals covered by critical searches, not just the ones corresponding to the lower convex hull of the lower bounds. Thus, the expected latency of our concatenation is at most $E[\beta\pi] \le \beta\gamma OPT$. All of the nodes within distance $\frac{c_{\max}}{8n^3}$ of the root were visited in arbitrary order first, which increases the total latency of the tour by at most $\frac{c_{\max}}{4n} \le \frac{OPT_n}{4n} \le \frac{OPT}{4n}$. Thus, the final tour has expected latency at most $(2 - \frac{1}{4n})\gamma OPT < 2\gamma OPT$. □

As Goemans and Kleinberg observed, the algorithm above can be derandomized using the following observation. Instead of choosing $L_0 = \gamma^U$, where U is drawn uniformly from $[0, 1)$, fix some positive integer p, and draw U uniformly from $\{\frac{k}{p} : 0 \le k < p\}$. This changes the factor of γ in the approximation guarantee of Theorem 7.2 to

$$\frac{(\gamma+1)\gamma^{\frac{1}{p}}}{p(\gamma^{\frac{1}{p}} - 1)},$$

which we denote by $r(p)$. Therefore, if we run the randomized algorithm for each of the p choices of the discrete random variable U and take the best solution, we get a deterministic algorithm that achieves an approximation factor of $\beta r(p)$, plus the tiny term for the nodes we deleted to create the amended instance.

As we would expect, $r(p) \to \gamma$ as $p \to \infty$. A Taylor series expansion reveals that

$$r(p) = \gamma \left(1 + \frac{\ln \gamma}{2p} + O\left(\frac{1}{p^2}\right)\right).$$

This yields the following result.

THEOREM 7.4. *For any $\epsilon > 0$, there is a deterministic algorithm for the minimum latency problem that gives an approximation factor of $(2 - \frac{1}{4n})\gamma(1+\epsilon)$ and runs in time dominated by $O(\frac{1}{\epsilon}\log^2 n)$ PCST calls.*

Setting $\epsilon = \frac{1}{8n}$, we can recover a deterministic 2γ-approximation algorithm using $O(n \log^2 n)$ PCST calls, which is slower than our old deterministic algorithm. But for any constant $\epsilon > 0$, we save a factor of $\frac{n}{\log n}$ compared to Theorem 6.1.

8. The weighted minimum latency problem. In this section, we consider the weighted MLP, in which each node v has an associated positive integer weight w_v. The goal is to find a tour that minimizes the sum over all nodes of the weight of the node times its latency. As in the unweighted case, the latency of a node in a tour is its distance along the tour from the root node. Since the root node r always has zero latency, its weight is irrelevant. The unweighted case discussed so far is simply the case in which $w_v = 1$ for all nodes v.

8.1. A pseudopolynomial-time algorithm. As we observed in section 1, from a weighted instance we can derive an equivalent unweighted instance by replacing each node v with a clique of w_v nodes at distance zero from each other. This new instance has $W = \sum_v w_v$ nodes. Recall that all weights are positive integers, so $W \geq n$. We can apply our deterministic algorithm of section 3 to obtain a 2γ-approximation while making $O(W \log W)$ PCST calls. In particular, we use our tree-generating algorithm of section 5 (along with the tweaks of section 6 to speed up the running time) to find some set of ℓ trees $T_{t_1}, \dots, T_{t_\ell}$ rooted at r and spanning total weights $t_1 < \cdots < t_\ell$ respectively, where $t_1 = 1$ and $t_\ell = W$. For $i = t_1, t_2, \dots, t_\ell$, we let d_i denote the cost of tree T_i. The tree-generating algorithm also establishes lower bounds b_w on OPT_w, the cost of the optimal tree rooted at r that has total node weight w. We call this a w-MST, as it generalizes the notion of a k-MST. Just as the cost of an optimal k-MST gives a lower bound on the latency of the kth node visited in the optimal MLP tour, the cost OPT_w of an optimal w-MST gives a lower bound on the length of the path in the optimal MLP tour from the root until nodes of total weight at least w have been visited. We claim then that $\sum_{w=1}^{W} OPT_w$ gives a lower bound on the cost of any weighted MLP tour. To see this, observe that, if l_i is the latency of the ith node visited in an optimal MLP tour, w_i is the weight of that node, and $W_i = \sum_{j=1}^{i} w_j$, then the cost of the optimal MLP tour is

$$\sum_{i=1}^{n} w_i l_i \geq \sum_{i=1}^{n} w_i OPT_{W_i} \geq \sum_{i=1}^{n} \sum_{w=W_{i-1}+1}^{W_i} OPT_w = \sum_{w=1}^{W} OPT_w.$$

As in the unweighted case, our lower bounds satisfy the properties that

1. $b_w \leq OPT_w$ for $1 \leq w \leq W$,
2. $d_{t_i} \leq \beta b_{t_i}$ for $1 \leq i \leq \ell$,
3. $b_w = b_{t_{i-1}} + \frac{b_{t_i} - b_{t_{i-1}}}{t_i - t_{i-1}}(w - t_{i-1})$ for $t_{i-1} \leq w \leq t_i$ and $i = 2, \dots, \ell$

for some $\beta > 2 - \frac{1}{n-1}$ that we will specify in the algorithm. We will show in section 9 that we can run the PCST algorithm on the weighted instance without creating extra nodes, so that each invocation of the PCST algorithm runs in $O(n^2)$ time. The algorithm and analysis of section 6 carry over, so we obtain a 2γ-approximation algorithm for the weighted MLP that uses $O(W \log W)$ PCST calls and takes overall time $O(\max(n^2 W \log W, W^2))$. The max comes because the shortest path computation is on a directed acyclic graph with W nodes and $\Theta(W^2)$ edges. This might dominate the running time of the PCST calls if W is large.

8.2. Weakly polynomial-time algorithms. In section 7, we converted our deterministic algorithm for the unweighted MLP to a faster randomized version. The exact same method applies to the weighted MLP. The only difference is in which nodes we delete to create the amended instance. This time we delete all nodes that are within distance $\frac{c_{\max}}{8n^2 W}$ of the root, so $\frac{c_{\max}}{c_{\min}} \leq 8n^2 W$ in the amended instance. We still set $\beta = 2 - \frac{1}{2n}$ when we perform our critical searches. By Corollary 9.5, each one requires $O(\log \frac{W c_{\max}}{c_{\min}}) = O(\log(nW)) = O(\log W)$ PCST calls (since $W \geq n$). The number of relevant threshold values is $O(\log \frac{n c_{\max}}{c_{\min}}) = O(\log W)$. At the end, our shortest path computation is on a directed acyclic graph with $O(\log^2 W)$ edges, so the running time is dominated by the $O(\log^2 W)$ PCST calls. The weighted latency of our tour on the amended instance is at most $(2 - \frac{1}{2n})\gamma OPT$, in expectation. The at most n nodes that were deleted to create the amended instance can be visited in arbitrary order at the beginning of the tour, which adds at most $\frac{c_{\max}}{4nW}$ to the latency of each node and so at most $\frac{c_{\max}}{4n} \leq \frac{OPT_W}{4n} \leq \frac{OPT}{4n}$ to the total latency of the tour.

THEOREM 8.1. *The randomized algorithm described above yields an approximation guarantee of 2γ in expectation for the weighted minimum latency problem and runs in time dominated by $O(\log^2 W)$ PCST calls.*

The algorithm above actually gives an approximation factor of less than $2(1 - \frac{1}{8n})\gamma$. Thus, if we derandomize it as in section 7, choosing p large enough so that $r(p) \leq 1 + \frac{1}{8n}$, we get a deterministic algorithm with a factor of at most 2γ. If we only want a factor of $2\gamma(1+\epsilon)$, we can save on the running time.

COROLLARY 8.2. *The deterministic algorithm described above for the weighted MLP gives an approximation factor of 2γ using $O(n \log^2 W)$ PCST calls, or a factor of $2\gamma(1+\epsilon)$ using only $O(\frac{1}{\epsilon} \log^2 W)$ PCST calls, for any $\epsilon > 0$.*

Proof. As in Theorem 7.4, setting $p = O(n)$ is sufficient to make $r(p) \leq 1 + \frac{1}{8n}$, and setting $p = O(\frac{1}{\epsilon})$ is sufficient to make $r(p) \leq (1+\epsilon)$. □

8.3. Strongly polynomial-time algorithms. Using one more clever trick, we can make the running time depend only on n and not on W. As in section 8.2, we do this first for our randomized algorithm, and then we derandomize it.

Corollary 9.5 of section 9 will show that we can perform each critical search using at most $O(\log \frac{nW c_{\max}}{w_{\min} c_{\min}})$ probes (i.e., PCST calls), where $w_{\min} = \min_{v \neq r} w_v$. Again, the number of relevant threshold values in the randomized algorithm of section 7 is $O(\log \frac{n c_{\max}}{c_{\min}})$. Thus, if we can limit ourselves to solving only amended instances where both $\frac{W}{w_{\min}}$ and $\frac{c_{\max}}{c_{\min}}$ are $\exp(\text{poly}(n))$, then we will achieve a strongly polynomial running time. In our weakly polynomial algorithm of section 8.2, we had $\frac{c_{\max}}{c_{\min}} = O(W \text{poly}(n))$, which yielded a dependence on $\log W$ in the running time.

We have already taken advantage of the fact that we can first visit all of the nodes very close to the root without increasing the total latency by much. Since we need to decrease $\frac{c_{\max}}{c_{\min}}$, we will need to refine this reasoning. Since the weighted latency of the node sitting at distance $c_{\max}$ from the root is at least $w_{\min} c_{\max}$, this is a lower bound on OPT. Thus, if we visit every node that is within distance $\frac{c_{\max} w_{\min}}{8n^2 W}$ of the root first, this contributes at most $\frac{OPT}{4nW}$ to the latency of each node and hence at most $\frac{OPT}{4n}$

to the total weighted latency of the tour. Thus, we can amend our instance to one in which $\frac{c_{\max}}{c_{\min}} \leq \frac{8n^2 W}{w_{\min}}$ while giving up only an extra $\frac{1}{4n}$ in the approximation factor. Therefore, we need only to worry about making $\frac{W}{w_{\min}}$ small. We will accomplish this by visiting the nodes of very small weight at the end.

With this motivation behind us, here is the description of our algorithm. Sort the nonroot nodes by lowest to highest weight. Scan the sorted list from the beginning, and insert a dividing point between any two consecutive nodes for which the ratio of their weights is more than $8n^3$. Let B_i denote the set of nodes between the $(i-1)$th and ith dividing points. We will call this set a *block*, and we will construct subtours on each block independently. If the blocks are $B_1, B_2, \dots, B_\ell$, we will traverse their subtours in the order $B_\ell, B_{\ell-1}, \dots, B_1$. Within each block, we will apply the randomized algorithm of section 8.2, where the amended instance for each block is derived by deleting all nodes within distance $\frac{c'_{\max} w'_{\min}}{8n^2 W'}$ of the root, where $w'_{\min}$ and W' are, respectively, the minimum weight of any node and the total weight of all nodes in the block.

THEOREM 8.3. *The randomized algorithm described above for the weighted MLP yields an approximation factor of* $(2 - \frac{1}{4n})\gamma$ *in expectation and runs in time dominated by* $O(n^2 \log^2 n)$ *PCST calls.*

Proof. Because the blocks partition the nodes and the running time for each block is superlinear in the number of nodes in the block, the worst case is when there is only one block. Thus, we analyze this case. Rename the nodes so that $w_1 \leq w_2 \leq \cdots \leq w_n$. By construction of the block, $w_{i+1} \leq 8n^3 w_i$ for each i. Thus, $W \leq n w_n \leq (8n^3)^n w_{\min}$, so $O(\log \frac{W}{w_{\min}}) = O(n \log n)$. Because $\frac{c_{\max}}{c_{\min}} \leq \frac{8n^2 W}{w_{\min}}$, we have $O(\log \frac{c_{\max}}{c_{\min}}) = O(n \log n)$ as well. Thus, by Corollary 9.5, each critical search takes $O(\log \frac{nW c_{\max}}{w_{\min} c_{\min}}) = O(n \log n)$ PCST calls, and there are $O(\log \frac{n c_{\max}}{c_{\min}}) = O(n \log n)$ relevant threshold values that require critical searches, for a total of $O(n^2 \log^2 n)$ PCST calls.

In analyzing the approximation ratio, we must account for the possibility of multiple blocks. First, let $OPT(B_i)$ denote the total latency of the optimal tour that starts at r and visits only the nodes in B_i. If $w'_{\min}$ is the minimum weight of any node in block B_i and $c'_{\max}$ is the distance from the root to the farthest node in B_i, then $w'_{\min} c'_{\max} \leq OPT(B_i)$. Moreover, $OPT(B_i)$ is a lower bound on the total latency of the nodes in B_i in the optimal tour that visits all nodes in the original instance. Thus, $OPT(B_1) + \cdots + OPT(B_\ell) \leq OPT$. Moreover, we will charge the latency of our overall tour block by block but in two pieces. First, we will account for the latency of block B_i's subtour, as if it were traversed on its own. Second, we will account for the amount that traversing B_i's subtour adds to the latencies of the subtours $B_{i-1}, \dots, B_1$ that come afterward in our final tour. Both costs will be charged against $OPT(B_i)$.

The expected latency of the subtour we generated for block B_i's amended instance is at most $(2 - \frac{1}{2n})\gamma OPT(B_i)$. Visiting the close deleted nodes in arbitrary order adds at most $\frac{c'_{\max} w'_{\min}}{4n} \leq \frac{OPT(B_i)}{4n}$ to the total weighted latency of all nodes in block B_i. Thus, the total latency of B_i's subtour, if it were done in isolation, would be at most $[(2 - \frac{1}{2n})\gamma + \frac{1}{4n}]OPT(B_i)$.

No matter what subtour we generated for block B_i, its total length is at most $2nc'_{\max}$. The total weight of all blocks $B_{i-1}, \dots, B_1$ is at most $\frac{w'_{\min}}{8n^2}$, so putting block B_i before them all adds at most $\frac{c'_{\max} w'_{\min}}{4n} \leq \frac{OPT(B_i)}{4n}$ to their total latency. Thus, the total amount that our subtour for block B_i adds to the weighted latency of the overall tour is at most $(2 - \frac{1}{4n})\gamma OPT(B_i)$. Summing this over the blocks gives the desired result. □

Derandomizing this algorithm using the technique of section 7 with $p = O(n)$ yields the following result.

Corollary 8.4. *There is a deterministic algorithm for the weighted MLP that achieves an approximation factor of* 2γ *in time dominated by* $O(n^3 \log^2 n)$ *PCST calls or a factor of* $2\gamma(1+\epsilon)$ *using only* $O(\frac{1}{\epsilon} n^2 \log^2 n)$ *PCST calls.*

We note that one can obtain a deterministic $2\gamma(1+\epsilon)$-approximation for the weighted MLP using only $O(\frac{1}{\epsilon} n^2 \log^2 \frac{1}{\epsilon})$ PCST calls, by dividing up the input nodes into blocks more cleverly and then into subblocks based on their distances from the root. However, since the description and analysis of this algorithm is significantly more complex and the payoff is just to shave off a $\log^2 n$ factor in the running time, we will omit the details.

9. The critical search primitive.

In this section, we show how to implement the critical search primitive that forms the basis for most of the algorithms and analyses of the previous sections. Recall our goal: Given a size target w^* or a bound target b^*, we wish to find an interval $[\lambda_{\text{lo}}, \lambda_{\text{hi}}]$ of λ values such that

1. $b_w \le OPT_w$ for $w_{\text{lo}} \le w \le w_{\text{hi}}$,
2. $c(T(\lambda)) \le \beta b_{|T(\lambda)|}$ for $\lambda \in \{\lambda_{\text{lo}}, \lambda_{\text{hi}}\}$,
3. $b_w = b_{w_{\text{lo}}} + \frac{b_{w_{\text{hi}}} - b_{w_{\text{lo}}}}{w_{\text{hi}} - w_{\text{lo}}}(w - w_{\text{lo}})$,

where $w_{\text{lo}} \le w^* \le w_{\text{hi}}$ in the case of a size target or $b_{w_{\text{lo}}} \le b^* \le b_{w_{\text{hi}}}$ in the case of a bound target.

For $w = w_{\text{lo}}, w_{\text{hi}}$, let $\tilde{b}_w$ be the lower bound $LB(\lambda)$ on OPT_w delivered by the PCST subroutine when run with $\lambda = \lambda_{\text{lo}}, \lambda_{\text{hi}}$, respectively. Set $\delta < 1$. We aim to show that, if $\lambda_{\text{hi}} - \lambda_{\text{lo}}$ is small enough, then setting $b_w = \delta \tilde{b}_w$ (for $w = w_{\text{lo}}, w_{\text{hi}}$) satisfies properties 1–3 above. In other words, scaling down the lower bounds by a factor of δ makes the interpolated lower bounds valid. Since $c(T(\lambda)) \le (2 - \frac{1}{n-1}) b_{|T(\lambda)|}$ (for $\lambda = \lambda_{\text{lo}}, \lambda_{\text{hi}}$), property 2 will hold with $\beta = \frac{1}{\delta}(2 - \frac{1}{n-1})$. For instance, $\delta = 1 - \frac{1}{4n-1}$ is sufficient to yield $\beta = 2 - \frac{1}{2n}$.

We begin by explaining where the lower bound comes from in the PCST algorithm and why it extends to give lower bounds on OPT_w for all values of w. We model the w-MST problem as the following integer program:

$$\text{Min} \quad \sum_{e \in E} c_e x_e$$

subject to:

$$\begin{aligned}
\sum_{e \in \delta(S)} x_e + \sum_{T: T \supseteq S} z_T &\ge 1 \quad && \forall S \subseteq V \setminus \{r\},\\
\sum_{S: S \subseteq V \setminus \{r\}} w(S) z_S &\le W - w, &&\\
x_e &\in \{0,1\} && \forall e \in E,\\
z_S &\in \{0,1\} && \forall S \subseteq V \setminus \{r\},
\end{aligned}$$

where $\delta(S)$ is the set of edges with exactly one end point in S and $w(S) = \sum_{v \in S} w_v$. The variable $x_e = 1$ indicates that the edge e is in the tree, while $z_S = 1$ indicates that the set of nodes S is not spanned. The first set of constraints says that, for any set S of nodes not containing the root, either they are contained in the unspanned set or some edge in $\delta(S)$ is selected. The second constraint says that the total weight of unspanned nodes is at most $W - w$.

Following [13], we can convert this to something close to a PCST problem by applying Lagrangian relaxation to the second constraint:

$$\text{Min} \quad \sum_{e\in E} c_e x_e + \lambda\left(\sum_{S:S\subseteq V\setminus\{r\}} w(S) z_S - (W-w)\right)$$

subject to:

$$\begin{aligned} \sum_{e\in\delta(S)} x_e + \sum_{T:T\supseteq S} z_T &\ge 1 && \forall S\subseteq V\setminus\{r\},\\ x_e &\in \{0,1\} && \forall e\in E,\\ z_S &\in\{0,1\} && \forall S\subseteq V\setminus\{r\}.\end{aligned}$$

Note that any solution feasible for the previous integer program will be feasible for this one, and if $\lambda \ge 0$, it will have no greater cost in the new integer program. Recall the definition of the PCST problem: We are given an undirected graph (V,E), a root node $r\in V$, nonnegative costs on the edges $c_e \ge 0$ for all $e\in E$, and nonnegative penalties p_i for $i\in V, i\ne r$. The goal is to find a tree spanning the root node so as to minimize the cost of the edges in the tree plus the penalties of the nodes not in the tree. Here we set all penalties $p_i = \lambda w_i$. Observe that the integer program above exactly models this problem for $p_i = \lambda w_i$, except that the objective function has an additional constant term of $-(W-w)\lambda$.

Goemans and Williamson [21] give a primal-dual 2-approximation algorithm for the PCST problem. Their algorithm returns a tree spanning the root node and a solution to the dual of a linear programming relaxation of the PCST problem. The dual solution is feasible for the dual of the linear programming relaxation of the integer program above; in particular, this dual is

$$\text{Max} \quad \sum_{S\subseteq V\setminus\{r\}} y_S - \lambda(W-w)$$

subject to:

$$\text{(D)} \qquad \begin{aligned} \sum_{S:e\in\delta(S)} y_S &\le c_e && \forall e\in E,\\ \sum_{T:T\subseteq S} y_T &\le w(S)\lambda && \forall S\subseteq V\setminus\{r\},\\ y_S &\ge 0 && \forall S\subseteq V\setminus\{r\}.\end{aligned}$$

We will abbreviate their algorithm as PCST. In particular, they show the following.

Theorem 9.1 (see [21]). *PCST returns a tree T and a dual solution y feasible for* (D) *such that if X is the set of nodes not spanned by T, then*

$$\sum_{e\in T} c_e + \left(2-\frac{1}{n-1}\right)\lambda w(X) \le \left(2-\frac{1}{n-1}\right)\sum_{S\subseteq V\setminus\{r\}} y_S.$$

From this theorem we obtain the following lemma.

Lemma 9.2. *Letting T and y be the tree and dual solution returned by PCST, respectively, define*

$$\tilde{b}_w := \sum_{S\subseteq V\setminus\{r\}} y_S - \lambda(W-w).$$

Then $\tilde{b}_w \leq OPT_w$ for each $w = 1, \dots, W$, and the cost of T is no more than $(2 - \frac{1}{n-1})\tilde{b}_{w(T)}$, where $w(T)$ is the total weight of the nodes spanned by T.

Proof. Note that if y is a feasible dual solution to (D), then since $\sum_{S\subseteq V\setminus\{r\}} y_S - \lambda(W-w)$ is the dual objective function of (D), it is a lower bound on the cost of an optimal w-MST. By Theorem 9.1, if PCST returns tree T and X is the set of nodes not spanned by T, then $w(X) = W - w(T)$. Thus

$$\sum_{e\in T} c_e + \left(2 - \frac{1}{n-1}\right)\lambda(W - w(T)) \leq \left(2 - \frac{1}{n-1}\right)\sum_{S\subseteq V\setminus\{r\}} y_S,$$

which implies that

$$\sum_{e\in T} c_e \leq \left(2 - \frac{1}{n-1}\right)\left(\sum_{S\subseteq V\setminus\{r\}} y_S - \lambda(W - w(T))\right) = \left(2 - \frac{1}{n-1}\right)\tilde{b}_{w(T)}. \qquad \square$$

Notice that running the PCST algorithm for a single value of λ yields lower bounds on OPT_w for *every* value of w simultaneously.

We further need the following observation, which relies on the workings of the PCST algorithm.

Observation 9.3. If we call PCST with $\lambda = 0$, it will return a tree containing only the root node. If we call PCST with $\lambda = \frac{c_{\max}}{w_{\min}}$, where $c_{\max}$ is the maximum edge cost and $w_{\min}$ is the minimum node weight, it will return a tree spanning all nodes.

We are now ready to prove our crucial lemma.

LEMMA 9.4. *Let T_{lo}, T_{hi} be the trees and $\tilde{b}_{\text{lo}}, \tilde{b}_{\text{hi}}$ the lower bounds returned by PCST when the penalty parameter λ is set to λ_{lo} and λ_{hi}, respectively, with $\lambda_{\text{lo}} \leq \lambda_{\text{hi}}$ and $\lambda_{\text{hi}} - \lambda_{\text{lo}} \leq \frac{1-\delta}{\delta}\frac{c_{\min}}{W}$. Let $w \in (w(T_{\text{lo}}), w(T_{\text{hi}}))$, and express w as a convex combination $w = \alpha_{\text{lo}} w(T_{\text{lo}}) + \alpha_{\text{hi}} w(T_{\text{hi}})$, where $\alpha_{\text{lo}} + \alpha_{\text{hi}} = 1$. If we set $b_{\text{lo}} = \delta\tilde{b}_{\text{lo}}, b_{\text{hi}} = \delta\tilde{b}_{\text{hi}}$, and $b_w = \alpha_{\text{lo}} b_{\text{lo}} + \alpha_{\text{hi}} b_{\text{hi}}$, then $b_w \leq OPT_w$.*

Proof. Let $w_{\text{lo}} = w(T_{\text{lo}})$ and $w_{\text{hi}} = w(T_{\text{hi}})$. Let y^{lo} and y^{hi} be the dual solutions returned by PCST for penalty value λ_{lo} and λ_{hi}, respectively. Letting $y = \alpha_{\text{lo}} y^{\text{lo}} + \alpha_{\text{hi}} y^{\text{hi}}$, observe that (y, λ_{hi}) is feasible for (D) by the convexity of the feasible region, and thus both $\sum_{S\subseteq V\setminus\{r\}} y_S - (W-w)\lambda_{\text{hi}}$ and $c_{\min}$ are lower bounds on the cost of an optimal w-MST. Then

$$\begin{aligned}
b_w &= \alpha_{\text{lo}} b_{\text{lo}} + \alpha_{\text{hi}} b_{\text{hi}} \\
&= \delta\left(\alpha_{\text{lo}}\tilde{b}_{\text{lo}} + \alpha_{\text{hi}}\tilde{b}_{\text{hi}}\right) \\
&= \delta\left(\alpha_{\text{lo}}\left(\sum_{S\subseteq V\setminus\{r\}} y_S^{\text{lo}} - (W - w_{\text{lo}})\lambda_{\text{lo}}\right) + \alpha_{\text{hi}}\left(\sum_{S\subseteq V\setminus\{r\}} y_S^{\text{hi}} - (W - w_{\text{hi}})\lambda_{\text{hi}}\right)\right) \\
&= \delta\left(\sum_{S\subseteq V\setminus\{r\}} y_S - \alpha_{\text{lo}}(W - w_{\text{lo}})(\lambda_{\text{hi}} + \lambda_{\text{lo}} - \lambda_{\text{hi}}) - \alpha_{\text{hi}}(W - w_{\text{hi}})\lambda_{\text{hi}}\right) \\
&\leq \delta\left(\sum_{S\subseteq V\setminus\{r\}} y_S - (W - w)\lambda_{\text{hi}} + \alpha_{\text{lo}}(W - w_{\text{lo}})\frac{1-\delta}{\delta}\frac{c_{\min}}{W}\right) \\
&\leq \delta\left(OPT_w + \frac{1-\delta}{\delta}OPT_w\right) \\
&\leq OPT_w. \qquad \square
\end{aligned}$$

Corollary 9.5. *Given $\delta < 1$ and $\beta \geq \frac{1}{\delta}(2 - \frac{1}{n-1})$, a single critical search can be performed using $O(\log \frac{\delta c_{\max} W}{(1-\delta) c_{\min} w_{\min}})$ PCST calls.*

Proof. Start the binary search with the interval $[0, \frac{c_{\max}}{w_{\min}}]$. Each PCST call halves the interval, and by Lemma 9.4, we can stop once we shrink the interval to width $\frac{(1-\delta)c_{\min}}{\delta W}$. □

In particular, if we set $\delta = 1 - \frac{1}{4n-1}$ in order to achieve $\beta = 2 - \frac{1}{2n}$, then each critical search takes $O(\log \frac{n c_{\max} W}{c_{\min} w_{\min}})$ PCST calls.

10. Experimental results. The approximation ratio of our algorithm for real instances is much better then one could expect by the worst-case analysis. We tested our deterministic algorithm of section 6 on some 2-dimensional Euclidean instances available from the TSPLIB [33]. In Table 2 we show the experimental factor between the obtained tour latency and the lower bound. The average ratio on the tested instances is 3.01, and its maximum is 3.66. We note that these numbers are much less than $2\gamma \approx 7.18$, which is the theoretical worst case.

11. Concluding remarks. We showed how to use the tree concatenation technique of Blum et al. as refined by Goemans and Kleinberg to construct a $2\gamma \approx 7.18$-approximation algorithm for the MLP, while having access to 2-approximate k-MSTs for only a few values of k that we cannot specify in advance. The 2γ guarantee comes from two sources. The 2 comes from our k-MSTs being 2-approximate, while the γ comes from the tree concatenation procedure. Both of these pieces represent significant barriers to further improvement.

The factor of $\gamma \approx 3.59$ from the tree concatenation is inherent in any analysis that blindly concatenates trees and upper bounds the latency by the sum of modified latencies. This is because Goemans and Kleinberg prove that Theorem 4.3 is tight; that is, the costs $d_2, \ldots, d_n$ can be selected such that the ratio of shortest path length in the graph G to $d_2 + \cdots + d_n$ is arbitrarily close to γ. Thus, in order to attain a provably better latency, one would need to either have some knowledge of the costs d_i or pay attention to the actual structure of the trees being concatenated.

All known constant factor approximation algorithms for the k-MST problem rely explicitly or implicitly on the LP relaxation we used for the PCST problem. Since this relaxation has an integrality gap of essentially 2, it seems that achieving a β-approximation algorithm for k-MST for a constant $\beta < 2$ will require a significantly different approach.

As we mentioned in section 1, Chaudhuri et al. [12] get around this difficulty by using the optimal k-stroll rather than the optimal k-MST as their lower bound on the latency of the kth node visited in the tour. Our work paved the way for theirs, since they use our technique of bluffing the GK algorithm with phantom trees, in order to make up for the fact that the Lagrangian relaxation will typically fail to yield trees of some sizes. Their algorithm requires them to guess the end point of the k-stroll, which incurs an extra factor of n in their running time. Their improvement comes because this allows them to raise the coefficient of nearly all of their dual variables from 1 to 2 in the objective function of the dual LP, which is a slight variant of (D). They use the analogous variant of the PCST algorithm to find their tree and dual solution. This is essentially the same trick used by Goemans and Williamson in [21] to transform their PCST algorithm into a 2-approximation for the prize-collecting TSP.

One direction for future work would be to consider LP relaxations that address the MLP objective function directly rather than using k-MSTs or k-strolls for our lower bounds. Perhaps the most attractive special case to look at is where the underlying

Table 2

Experimental results of our deterministic algorithm from section 6*. Running times are in CPU seconds on an IBM RS*/6000 43*P Model* 140*.*

Instance name	Tour latency	Lower bound	Factor	Running time
berlin52	197137	58644	3.36	0.983
bier127	5929120	1886700	3.14	4.033
ch130	455849	148344	3.07	11.033
ch150	571369	209537	2.72	13.833
d198	1380470	556278	2.48	21.983
d493	10441397	3305791	3.15	144.150
d657	20831492	6487270	3.21	288.000
eil101	38582	12157	3.17	2.900
eil51	14683	4390	3.34	0.383
eil76	26128	8046	3.24	1.017
fl417	2531146	825513	3.06	258.867
gil262	393641	126697	3.10	34.183
kroA100	1307340	432542	3.02	3.683
kroA150	2494782	811515	3.07	8.967
kroA200	3387616	1173404	2.88	40.667
kroB100	1274207	442308	2.88	2.083
kroB150	2376125	820770	2.89	6.883
kroB200	3731218	1174833	3.17	44.083
kroC100	1207746	432224	2.79	4.700
kroD100	1297932	412501	3.14	4.450
kroE100	1345314	446334	3.01	1.800
lin105	780662	274250	2.84	4.767
lin318	7475822	2532401	2.95	128.750
p654	10251922	3545177	2.89	408.000
pcb442	14683399	4844532	3.03	79.133
pr1002	164844296	50583204	3.25	1479.983
pr107	2205490	915582	2.40	124.350
pr124	4778217	1454570	3.28	7.817
pr136	8720053	2891809	3.01	30.967
pr144	4844537	1674418	2.89	30.700
pr152	6075505	2334659	2.60	49.333
pr226	10421449	3283953	3.17	15.750
pr264	7674241	2628452	2.91	26.517
pr299	8553790	2938150	2.91	109.250
pr439	24126010	7900826	3.05	143.083
pr76	4359810	1467212	2.97	1.350
rat195	280900	102741	2.73	25.850
rat575	2511713	847350	2.96	418.833
rat783	4410164	1527124	2.88	449.983
rat99	75048	25964	2.89	5.033
rd100	458419	153887	2.97	5.250
rd400	3930767	1230238	3.19	93.517
st70	26384	9033	2.92	0.800
ts225	17953213	6271875	2.86	2815.500
tsp225	537080	181263	2.96	88.733
u1060	146511585	46213643	3.17	753.283
u159	3837650	1301475	2.94	9.433
u574	12906940	4159616	3.10	195.233
u724	19821239	6222958	3.18	517.433
vm1084	153128900	41816544	3.66	1041.333

metric is given by a tree. Since the k-MST problem can be solved optimally on trees, Goemans and Kleinberg [22] used their algorithm to obtain a 3.59-approximation for this special case, without resorting to the k-stroll approach of [12]. This is still the best result known for tree metrics.

Acknowledgments. A preliminary version of this paper by the first and third authors [4] had a performance guarantee of 9.28. The second author contributed the core idea for the improvement of the performance guarantee to 7.18, which first appeared in [3]. We thank Tim Roughgarden for many enlightening discussions and Martin Pál, David Shmoys, and Éva Tardos for helpful comments on our presentation.

REFERENCES

[1] F. Afrati, S. Cosmadakis, C. H. Papadimitriou, G. Papageorgiou, and N. Papakostantinou, *The complexity of the traveling repairman problem*, Informatique Theorique et Applications, 20 (1986), pp. 79–87.

[2] S. R. Agnihothri, *A mean value analysis of the travelling repairman problem*, IIE Transactions, 20 (1988), pp. 223–229.

[3] A. Archer, A. Levin, and D. P. Williamson, *A faster, better approximation algorithm for the minimum latency problem*, Technical report 1362, Cornell University ORIE, 2003.

[4] A. Archer and D. P. Williamson, *Faster approximation algorithms for the minimum latency problem*, in Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, 2003, pp. 88–96.

[5] S. Arora and G. Karakostas, *Approximation schemes for minimum latency problems*, SIAM J. Comput., 32 (2003), pp. 1317–1337.

[6] S. Arora and G. Karakostas, *A* $2+\epsilon$ *approximation algorithm for the k-MST problem*, Math. Program., 107 (2006), pp. 491–504.

[7] S. Arya and H. Ramesh, *A* 2.5 *factor approximation algorithm for the k-MST problem*, Inform. Process. Lett., 65 (1998), pp. 117–118.

[8] G. Ausiello, S. Leonardi, and A. Marchetti-Spaccamela, *On salesmen, repairmen, spiders and other traveling agents*, in Proceedings of the Italian Conference on Algorithms and Complexity, Lecture Notes in Comput. Sci., 1767, Springer, Berlin, 2000, pp. 1–16.

[9] I. Averbakh and O. Berman, *Sales-delivery man problems on treelike networks*, Networks, 25 (1995), pp. 45–58.

[10] L. Bianco, A. Mingozzi, and S. Ricciardelli, *The traveling salesman problem with cumulative costs*, Networks, 23 (1993), pp. 81–91.

[11] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan, *The minimum latency problem*, in Proceedings of the 26th Annual ACM Symposium on Theory of Computing, 1994, pp. 163–171.

[12] K. Chaudhuri, B. Godfrey, S. Rao, and K. Talwar, *Paths, trees, and minimum latency tours*, in Proceedings of the 44th Annual IEEE Symposium on the Foundations of Computer Science, 2003, pp. 36–45.

[13] F. A. Chudak, T. Roughgarden, and D. P. Williamson, *Approximate k-MSTs and k-Steiner trees via the primal-dual method and Lagrangean relaxation*, Math. Program., 100 (2004), pp. 411–421.

[14] J. Fakcharoenphol, C. Harrelson, and S. Rao, *The k-traveling repairman problem*, in Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, 2003, pp. 655–664.

[15] E. Feuerstein and L. Stougie, *On-line single-server dial-a-ride problems*, Theoret. Comput. Sci., 268 (2001), pp. 91–105.

[16] M. Fischetti, G. Laporte, and S. Martello, *The delivery man problem and cumulative matroids*, Oper. Res., 41 (1993), pp. 1065–1064.

[17] F. V. Fomin and A. Lingas, *Approximation algorithms for time-dependent orienteering*, Inform. Process. Lett., 83 (2002), pp. 57–62.

[18] H. N. Gabow and S. Pettie, *The dynamic vertex minimum problem and its application to clustering-type approximation algorithms*, in Proceedings of the 8th Scandinavian Workshop on Algorithm Theory, Lecture Notes in Comput. Sci. 2368, Springer, Berlin, 2002, pp. 190–199.

[19] N. Garg, *A* 3*-approximation for the minimum tree spanning k vertices*, in Proceedings of the 37th Annual Symposium on Foundations of Computer Science, IEEE, Piscataway, NJ, 1996, pp. 302–309.

[20] N. Garg, *Saving an epsilon: A* 2*-approximation for the k-MST problem in graphs*, in Proceedings of the 37th Annual ACM Symposium on Theory of Computing, 2005, pp. 396–402.

[21] M. X. Goemans and D. P. Williamson, *A general approximation technique for constrained forest problems*, SIAM J. Comput., 24 (1995), pp. 296–317.

[22] M. Goemans and J. Kleinberg, *An improved approximation ratio for the minimum latency problem*, Math. Program., 82 (1998), pp. 111–124.

[23] R. Hassin and A. Levin, *An approximation algorithm for the minimum latency set cover problem*, in Proceedings of the 13th Annual European Symposium on Algorithms, Lecture Notes in Comput. Sci. 3669, Springer, Berlin, 2005, pp. 726–733.

[24] K. Jain and V. V. Vazirani, *Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation*, J. ACM, 48 (2001), pp. 274–296.

[25] E. Koutsoupias, C. H. Papadimitriou, and M. Yannakakis, *Searching a fixed graph*, in Proceedings of the 23rd International Colloquium on Automata, Languages, and Programming, Lecture Notes in Comput. Sci. 1099, Springer, Berlin, 1996, pp. 280–289.

[26] S. O. Krumke, W. de Paepe, D. Poensgen, and L. Stougie, *News from the online traveling repairman*, Theoret. Comput. Sci., 295 (2003), pp. 279–294.

[27] G. Lin, C. Nagarajan, R. Rajaraman, and D. P. Williamson, *A general approach for incremental approximation and hierarchical clustering*, in Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms, 2006, pp. 1147–1156.

[28] A. Lucena, *Time-dependent traveling salesman problem - the deliveryman case*, Networks, 20 (1990), pp. 753–763.

[29] N. Megiddo, *Combinatorial optimization with rational objective functions*, Math. Oper. Res., 4 (1979), pp. 414–424.

[30] E. Minieka, *The delivery man problem on a tree network*, Ann. Oper. Res., 18 (1989), pp. 261–266.

[31] C. H. Papadimitriou and M. Yannakakis, *The traveling salesman problem with distances one and two*, Math. Oper. Res., 18 (1993), pp. 1–11.

[32] J.-C. Picard and M. Queyranne, *The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling*, Oper. Res., 26 (1978), pp. 86–110.

[33] G. Reinelt, *TSPLIB*, Technical report, Universität Heidelberg Institut für Informatik, Im Neuenheimer Feld 368, D-69120 Heidelberg, Germany.

[34] S. Sahni and T. Gonzalez, *P-complete approximation problems*, J. ACM, 23 (1976), pp. 555–565.

[35] D. Simchi-Levi and O. Berman, *Minimizing the total flow time of n jobs on a network*, IIE Transactions, 23 (1991), pp. 236–244.

[36] R. Sitters, *The minimum latency problem is NP-hard for weighted trees*, in Proceedings of the 9th Conference on Integer Programming and Combinatorial Optimization, Lecture Notes in Comput. Sci. 2337, Springer, Berlin, 2002, pp. 230–239.

[37] J. N. Tsitsiklis, *Special cases of traveling salesman and repairman problems with time windows*, Networks, 22 (1992), pp. 263–282.

[38] R. J. Vander Wiel and N. V. Sahinidis, *Heuristic bounds and test problem generation for the time-dependent traveling salesman problem*, Transportation Science, 29 (1995), pp. 167–183.

[39] R. J. Vander Wiel and N. V. Sahinidis, *An exact solution approach for the time-dependent traveling-salesman problem*, Naval Res. Logist., 43 (1996), pp. 797–820.

[40] I. R. Webb, *Depth-first solutions for the deliveryman problem on tree-like networks: An evaluation using a permutation model*, Transportation Science, 30 (1996), pp. 134–147.

[41] B. Wu, *Polynomial time algorithms for some minimum latency problems*, Inform. Process. Lett., 75 (2000), pp. 225–229.

[42] C. Yang, *A dynamic programming algorithm for the travelling repairman problem*, Asia-Pac. J. Oper. Res., 6 (1989), pp. 192–206.

SIAM J. COMPUT.
Vol. 37, No. 5, pp. 1499–1516

OPTIMAL POWER-DOWN STRATEGIES[*]

JOHN AUGUSTINE[†], SANDY IRANI[‡], AND CHAITANYA SWAMY[§]

Abstract. We consider the problem of selecting threshold times to transition a device to low-power sleep states during an idle period. The two-state case, in which there is a single active and a single sleep state, is a continuous version of the ski-rental problem. We consider a generalized version in which there is more than one sleep state, each with its own power-consumption rate and transition costs. We give an algorithm that, given a system, produces a deterministic strategy whose competitive ratio is arbitrarily close to optimal. We also give an algorithm to produce the optimal online strategy given a system and a probability distribution that generates the length of the idle period. We also give a simple algorithm that achieves a competitive ratio of $3 + 2\sqrt{2} \approx 5.828$ for any system.

Key words. online algorithms, power-aware computation, dynamic power management

AMS subject classifications. 68M07, 49N30, 68M20, 68W40, 68W25

DOI. 10.1137/05063787X

1. Introduction.

Suppose you are about to go skiing for the first time in your life. Naturally, you ask yourself whether to rent skis or to buy them. Renting skis costs, say, $30, whereas buying skis costs $300. If you knew how many times you would go skiing in the future (ignoring complicating factors such as inflation and changing models of skis), then your choice would be clear. If you knew you would go at least 10 times, you would be financially better off by buying skis right from the beginning, whereas if you knew you would go less than 10 times, you would be better off renting skis every time. Alas, the future is unclear, and you must make a decision nonetheless.

Although the *ski-rental* problem is a very simple abstraction, this basic paradigm arises in many applications in computer systems. In these situations, there is a system that can reside in either a low-cost or a high-cost state. Occasionally, it is forced to be in the high-cost state (usually to perform some task). A period between any two such points in time is called an *idle period*.

The system pays a per time unit cost to reside in the high-cost state. Alternatively, it can transition to the low-cost state at a fixed one-time cost. If the idle period is long, it is advantageous to transition to the low-cost state immediately; if the idle period is short, it is better to stay in the high-cost state. An online algorithm which does not know the length of the idle period must balance these two possibilities.

[*] Received by the editors August 10, 2005; accepted for publication (in revised form) October 10, 2007; published electronically January 30, 2008.
http://www.siam.org/journals/sicomp/37-5/63787.html

[†] School of Information and Computer Science, University of California at Irvine, Irvine, CA 92697. Current address: Tata Research Development and Design Center, Pune 411 013, India (john.augustine@tcs.com). This author's research was supported partially by NSF grants CCR-0105498 and CCF-0514082 and by ONR Award N00014-00-1-0617.

[‡] School of Information and Computer Science, University of California at Irvine, Irvine, CA 92697 (irani@ics.uci.edu). This author's research was supported partially by NSF grants CCR-0105498 and CCF-0514082 and by ONR Award N00014-00-1-0617.

[§] Center for the Mathematics of Information, Caltech, Pasadena, CA 91125. Current address: Department of Combinatorics and Optimization, Faculty of Mathematics, University of Waterloo, Waterloo, ON N2L 3G1, Canada (cswamy@math.uwaterloo.ca). This work was done while this author was a student at the Department of Computer Science, Cornell University, Ithaca, NY 14853. This author's research was supported partially by NSF grant CCR-9912422.

This problem has been studied in the context of shared memory multiprocessors in which a thread is waiting for a locked piece of data and must decide whether to spin or block [8, 10]. Researchers investigating the interface between IP networks and connection-oriented networks have discovered this same underlying problem in deciding whether to keep a connection open between bursts of packets that must be sent along the connection [11]. Karlin, Kenyon, and Randall study the transmission control protocol (TCP) acknowledgment problem and the related Bahncard problem, both of which are at heart ski-rental problems [9]. The problem also arises in cache coherency in deciding whether to update or invalidate data that has been changed in a processor's local cache [5, 1].

An important application of the ski-rental problem is in minimizing the power consumed by devices that can transition to a low-power *sleep* state when idle. The sleep state consumes less power; however, one incurs a fixed start-up cost in making the transition to the high-power *active* state in order to begin work when a new job arrives. At the architectural level, the technique of eliminating power to a functional component is called clock/power gating. At a higher level, the powered-down component might be a disk drive or even the whole system (e.g., a laptop that hibernates). The embedded systems community has invested a great deal of effort in devising policies governing the selection of power states during idle periods (termed *dynamic power management* in their literature); see, for example, [3] for a survey. These techniques have been critical to maximizing battery use in mobile systems. While power is already a first-class parameter in system design, it will become increasingly important in the future since battery capacities are increasing at a much slower rate than power requirements.

Most of the previous work on this problem has been concerned with two-state systems, which have an active state and a *single* sleep state. This paper focuses on finding power-down thresholds for systems that have more than one low-power state.

2. Previous work and new results.

For the two-state problem, an online algorithm consists of a single threshold T after which time the algorithm will transition from the active to the sleep state. The input to the problem is the length of the idle period, and the cost of an algorithm is the total amount of energy it consumes over a single idle period. Typically, an online algorithm is evaluated in terms of its competitive ratio—the ratio of the cost of the online algorithm to the cost of the optimal offline algorithm, maximized over all inputs. When randomized algorithms are considered where the threshold T is chosen at random, we look at the ratio of the expected cost of the online algorithm to the cost of the offline algorithm. Previous work has also addressed the two-state problem when the idle period is generated by a known probability distribution. In this case, the online algorithm will choose a threshold which minimizes its expected cost, where the expectation here is taken over the random choice of the idle period. We call such algorithms *probability-based* algorithms.

The best deterministic online algorithm will stay in the high-power state until the total energy spent is equal to the cost to power up from the low-power state. It is known that this algorithm achieves the optimal (deterministic) competitive ratio of 2 [8]. When one considers randomized online algorithms, the best competitive ratio achievable improves to $e/(e-1)$ [8]. If the idle period is generated by a known probability distribution, then the algorithm that chooses T so as to minimize the expected cost is always within a factor of $e/(e-1)$ of optimal. Furthermore, this bound is tight since there is a distribution over the idle period lengths which will force any online algorithm to incur an expected cost that is a factor $e/(e-1)$ times larger than that incurred by the optimal offline algorithm [8].

Note that in the context of power-down systems, it may not be the case that the power usage in the sleep state is zero or even that the start-up cost in the active state is zero. In these cases, both the online and the offline algorithms will incur an identical additional cost. Thus, the ratio of the online to the offline costs will decrease, and the optimal competitive ratio will be strictly less than two. However, these additional costs do not change the optimal online or offline strategy in either the deterministic or the probability-based case, and the optimal competitive ratio that can be achieved for such systems can easily be determined as a function of all the parameters of the system.

We denote the problem that involves powering down through k sleep states $PD(k)$. A formal description of the problem is as follows: we are given a sequence of $k+1$ states $S = \langle s_0, \ldots, s_k \rangle$. There is also a vector of power-consumption rates $\mathcal{K} = \langle \kappa_0, \ldots, \kappa_k \rangle$, where κ_i is the power-consumption rate of the system in state s_i. We assume as a convention that the states are ordered so that $\kappa_i > \kappa_j$ for $0 \leq i < j \leq k$. So s_0 is the *active state*, and the system must transition to s_0 (i.e., power up) at the end of the idle period. There is an associated transition cost $d_{i,j}$ to move from state s_i to s_j. A *system* is described by a pair $(\mathcal{K}, d)$. Note that there can be costs to move from high-power states to low-power states and vice versa. However, the only power-up costs that are of interest are the costs to transition from a particular state s_i to the active state s_0 since the only reason to transition to a higher-power state is when a new task arrives. A *schedule* or *strategy* $\mathcal{A} = (\mathcal{S}_\mathcal{A}, \mathcal{T}_\mathcal{A})$ consists of a sequence of $n_\mathcal{A}+1$ states $\mathcal{S}_\mathcal{A}$, which is a subsequence of S, and a sequence of transition times $\mathcal{T}_\mathcal{A}$. Where obvious, we will omit the subscript $\mathcal{A}$. We require that $\mathcal{S}(0) = s_0$ and $T(0) = 0$. We use $\mathcal{A}(t)$ to denote the cost of the schedule produced by strategy $\mathcal{A}$ for an idle period of length t. We also consider a generalization of $PD(k)$ that we call $PD(k, m)$ wherein we require that $n_\mathcal{A} \leq m$, where $0 < m \leq k$ is some limiting integer constant. This generalization would be especially useful for engineers who have a large number of sleep state options available in the design phase but are required to implement at most a fixed number of states in the product that rolls out into the market.

The only previous work that examines the multiple-state problem $PD(k)$ (from the perspective of worst-case guarantees) is [6], which considers the special case where the cost to power down is zero and the algorithm pays only to move from low-power states to higher-power states. Note that this also includes the case where the transition costs are additive ($d_{i,j} + d_{j,k} = d_{i,k}$ for $i < j < k$) since the costs to power down can then be folded into the costs to power up. [6] gives natural generalizations of the algorithms for the two-state case, both for the case when the idle period length is unknown and the case when it is generated by a known probability distribution. It is shown that when the transition costs are additive, the generalized deterministic algorithm is 2-competitive and the probability-based algorithm is $e/(e-1)$-competitive, thus matching the guarantees in the two-state case.

There are two important directions left open by this work. The first is based on the observation that systems, in general, do not have additive transition costs. In many scenarios, additional energy is spent in transitioning to lower-power states. Furthermore, there could be overhead in stopping at intermediate states, resulting in nonadditive transition costs (see [3] for an example). The second point is that the known upper bounds are typically not optimal *for the system under consideration.* That is, while it is true that there *exist* systems for which the optimal competitive ratio that can be achieved by any deterministic algorithm is 2 (and $e/(e-1)$ by any randomized algorithm), it is possible to achieve a better competitive ratio for many systems. For multistate systems, the optimal competitive ratio that can be achieved will, in general, be a complicated function of all the parameters of the system (the power-consumption rates as well as transition costs). For probability-based algorithms, the optimal competitive ratio will also depend on the probability distribution generating the length of the idle period. While it may not be feasible to express the optimal competitive ratio as a function of all these parameters, a system designer would, in general, like to design a power-down strategy that obtains the best possible competitive ratio given the constraints of his or her particular system.

This paper establishes the following results.

- We give an algorithm that takes as input an instance of $PD(k)$ that is described by $(\mathcal{K}, d)$, and an error parameter ϵ, and produces a power-down strategy $\mathcal{A} = (\mathcal{S}_{\mathcal{A}}, \mathcal{T}_{\mathcal{A}})$ whose competitive ratio is within an additive ϵ of the best competitive ratio that can be achieved for that system. The algorithm runs in time $O(k^2(\log k)\log(1/\epsilon))$, where $k+1$ is the number of states in the system, and also outputs the competitive ratio of $\mathcal{A}$. The algorithm works via a decision procedure which determines for a system and a constant ρ if there is a ρ-competitive strategy for that system. This decision procedure also allows us to obtain lower bounds on the competitive ratio achievable by deterministic algorithms for specific systems, which in turn provides a lower bound on the competitive ratio achievable by deterministic algorithms in general. In particular, we obtain a lower bound of 2.45 on the competitive ratio for deterministic algorithms. This is the first lower bound known that is greater than 2. Independently, Damaschke has given a lower bound of 3.618 [4].
- The above approach can be modified to solve the more general version where a bound of m is specified on the number of states allowed in final strategy. We show how to extend the decision procedure to answer if there is a ρ-competitive strategy for the system that uses at most m power states.
- Experimental results show that there are significant performance gains to be made by estimating the distribution governing the length of an idle period based on recent history and using this estimate to drive a probability-based strategy [7]. We give an algorithm that takes as input a description of a system and a probability distribution generating the idle period length and produces the optimal power-down strategy. Naturally, the running time of the algorithm will depend on the representation of the distribution. In practice, this is most likely to be a histogram. Our algorithm runs in time $O(k^2(\log k + B))$, where B is the number of bins in the histogram and $k+1$ is the number of states. One outcome of the proof is that it also establishes the optimality of the strategy given in [6] for *additive* systems. We then generalize this to find the best online algorithm subject to the restriction that at most m states are used, at the expense of an extra factor of m in the running time.
- We give a simple deterministic strategy that achieves a competitive ratio of $3+2\sqrt{2} \approx 5.8284$ for all systems. This result gives a bound on the competitive ratio achieved by the optimal strategies generated by our algorithms. Note that $3+2\sqrt{2}$ also serves as a bound on the ratio of the expected costs of the online and offline algorithms when the input is probabilistically generated.

In the remainder of this paper, we use the terms *schedule* or *strategy* interchangeably to refer to the choices of states and threshold times for powering down. The term *algorithm* will refer to a procedure that produces a schedule or strategy based on a particular system.

Azar et al. in [2] consider a related problem which they refer to as *capital investment.* This problem is a different generalization of the ski-rental problem than the power-down problem considered here. However, a special case of their problem coincides with a special case of our problem. Specifically, they give a $(4 + 2\sqrt{2})$-competitive deterministic algorithm for the special case of the power-down problem in which the cost to transition to each state is the same, regardless of the state from which one is transitioning. Later, Damaschke in [4] improves the upper bound on the competitive ratio for this special case (also in the context of capital investment) to 4 for deterministic algorithms and 2.88 for ranomized algorithms. In addition, Damaschke gives a 3.618 lower bound for any deterministic algorithm which subsumes the lower bound of 2.45 given here.

3. Preliminaries. First we will establish that we can assume without loss of generality that the power-up transition costs are zero. If this is not the case for some system $(\mathcal{K}, d)$, we can define a new system such that for any $i < j$, the cost to transition from s_i to s_j is $d_{i,j} + d_{j,0} - d_{i,0}$ and the cost to go from s_j to s_i is 0. Since there is never any reason to transition to a higher-power state unless the system is transitioning to the active state at the arrival of a new task, any set of actions in the original system will incur the same cost in the new system. Thus, in what follows we assume that $d_{i,0} = 0$ for all i.

We also need to establish that we can assume that for all $i < j$, $d_{i,j} < d_{0,j}$. Recall that we are really using $d_{i,j}$ to denote $d_{i,j} + d_{j,0} - d_{i,0}$ and $d_{0,j}$ to denote $d_{0,j} + d_{j,0}$. Thus, the assumption that $d_{i,j} < d_{0,j}$ really amounts to assuming that $d_{i,j} < d_{i,0} + d_{0,j}$. If this were not the case, we could just transition from state s_i to state s_j by first going to s_0 and then going down to s_j.

Let $D(i)$ denote $d_{0,i}$. Then $OPT(t) = \min_i(D(i) + \kappa_i t)$. Let $S(t)$ denote the state which attains the minimum—the optimal state. The optimal strategy is to transition to state $S(t)$ at time 0 and stay there through time t. We assume that, for every state, there is some idle period length for which the optimal strategy will use that state, i.e., range$(S(t)) = \{s_0, \dots, s_k\}$. None of the online strategies we present will make use of a state that is never used by the optimal offline strategy for any time t.

Note that $OPT(t)$ is piecewise-linear and $S(t)$ is nondecreasing with t—as the idle period length gets longer, it becomes more worthwhile to pay the extra cost to transition to a lower-power state. Let b_i denote the first time instant at which state s_i becomes the optimal state, so $b(0) = 0$ and $D(i-1) + \kappa_{i-1} b_i = D(i) + \kappa_i b_i \Rightarrow b_i = \frac{D(i)-D(i-1)}{\kappa_{i-1}-\kappa_i}$. We have $b(0) < b(1) < \dots b(k)$. Figure 1 shows the total energy consumed by OPT as a function of the length of the idle period. There is a line for each state. The y-intercept is the transition cost to move to that state from the active state and the slope is the power-consumption rate. The energy consumed by the optimal strategy is the lower envelope of these lines since it will pick the single state which minimizes the cost for a given idle period length. Thus for $t \in [b_i, b_{i+1}]$,

$$OPT(t) = D(i) + \kappa_i t = \sum_{j=0}^{i-1} \kappa_j (b_{j+1} - b_j) + \kappa_i (t - b_i). \tag{1}$$

We compare our online strategy with $OPT(t)$ and want to get a strategy $\mathcal{A}$ which minimizes the competitive ratio, $c_{\mathcal{A}} = \sup_t \frac{\mathcal{A}(t)}{OPT(t)}$, where $\mathcal{A}(t)$ denotes the total power consumption of $\mathcal{A}$ by time t.

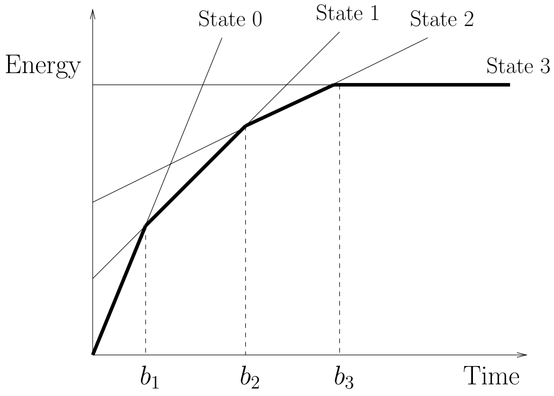


FIG. 1. *Energy consumed by the optimal strategy as a function of idle period length.*

4. A simple $(3+2\sqrt{2})$-competitive strategy.

Let us for the moment assume that for some $\gamma > 1$, $D(i) \geq \gamma D(i-1)$ for all $i = 1, \ldots, k$. This is a nontrivial assumption that we will have to handle later. Consider the strategy, $\mathcal{A}$, which always stays in state $S(t)$, the same state as OPT, at every time t. The optimal strategy which knows the length of the idle period in advance will just transition to the optimal state. Strategy $\mathcal{A}$, however, must "follow" the optimal strategy, making each transition to a new state as the idle period gets longer. This is the strategy proposed in [6] and shown to be 2-competitive for additive systems. Note that this strategy is the same as the 2-competitive balance strategy for the two-state case.

For $t \in [b_i, b_{i+1}]$ the online cost is $\mathcal{A}(t) = \sum_{j=0}^{i-1} \big(\kappa_j(b_{j+1} - b_j) + d_{j,j+1}\big) + \kappa_i(t - b_i)$. In comparing this cost to the optimal cost in (1), observe that both terms have an additive $\kappa_i(t - b_i)$, which means that the ratio $\frac{\mathcal{A}(t)}{OPT(t)}$ will be maximized at $t = b_i$. To bound the cost of $\mathcal{A}$ in terms of OPT, we use the fact that $OPT(b_i) \geq D(i)$ and $OPT(b_i) = \sum_{j=0}^{i-1} \kappa_j(b_{j+1} - b_j)$, both of which come from (1). This last equation is used in line three of the equations below as is the fact that $D(i) \geq \gamma D(i-1)$ for all $i = 1, \ldots, k$:

$$\begin{aligned}
\mathcal{A}(b_i) &= \sum_{j=0}^{i-1} \Big(\kappa_j(b_{j+1} - b_j) + d_{j,j+1}\Big) \\
&\leq \sum_{j=0}^{i-1} \kappa_j(b_{j+1} - b_j) + \sum_{j=1}^{i} D(j) \\
&\leq OPT(b_i) + D(i) \sum_{j=1}^{i} \gamma^{-(i-j)} \\
&\leq \left(1 + \frac{\gamma}{\gamma - 1}\right) OPT(b_i) \quad = \quad \frac{2\gamma - 1}{\gamma - 1} \cdot OPT(b_i). \qquad (2)
\end{aligned}$$

This holds for any t, implying a competitive ratio of $\frac{2\gamma-1}{\gamma-1}$.

Now suppose the assumption $D(i) \geq \gamma D(i-1)$ does not hold. We consider a new *offline* strategy OPT' that uses only a subset of states S' for which the property does hold and is a γ-approximation of OPT; i.e., $OPT'(t) \leq \gamma \cdot OPT(t)$. We now view our problem as specified by just the states in S', and we execute strategy $\mathcal{A}$ as specified above, emulating OPT' instead of OPT. We get that $\mathcal{A}'(t) \leq \frac{2\gamma-1}{\gamma-1} OPT'(t) \leq \frac{\gamma(2\gamma-1)}{\gamma-1} OPT(t)$. Setting $\gamma = 1 + \frac{1}{\sqrt{2}}$, we get a competitive ratio of $3 + 2\sqrt{2} \approx 5.8284$.

We determine OPT' as follows. Let $S' = \{s_k\}$ initially. Consider the states in S in reverse order. Let s_i be the last state added to S'. We find the largest j, $0 \leq j < i$, such that $D(j) \leq D(i)/\gamma$. We add s_j to S' and continue until no such j exists. Note that $s_0 \in S'$ since $D(0) = 0$. OPT' will execute the optimal offline strategy assuming that only the states in S' are available. Consider i, j such that $s_i, s_j \in S'$ and no s_ℓ is in S' for $i < \ell < j$. We have $OPT'(t) = OPT(t)$ for $t \in [b_i, b_{i+1})$ and $t \in [b_j, b_{j+1})$. For ℓ such that $i < \ell < j$ and time $t \in [b_\ell, b_{\ell+1})$, $OPT'(t) = \min(D(i)+\kappa_i t, D(j)+\kappa_j t)$ and $OPT(t) = D(\ell) + \kappa_\ell t$. j was chosen to be the largest value less than i such that $D(j) \leq D(i)/\gamma$, which means that $D(\ell) > D(i)/\gamma$. Furthermore, since $\kappa_i \leq \kappa_\ell$, we have that

$$OPT'(t) \leq D(i) + \kappa_i t \leq \gamma\big(D(\ell) + \kappa_\ell t\big) = \gamma OPT(t),$$

and OPT' is a γ-approximation to OPT.

THEOREM 1. *There is a $(3 + 2\sqrt{2})$-competitive strategy for any system.*

5. A near-optimal deterministic algorithm.

In this section, we turn our attention to obtaining a near-optimal schedule for a particular system. More precisely, given a system $(\mathcal{K}, d)$ with state sequence S for which the optimal online schedule has competitive ratio ρ^*, we give an algorithm that returns a $(\rho^* + \epsilon)$-competitive online schedule in time $O(k^2 \log k \log(1/\epsilon))$. The algorithm is based on a decision procedure which determines whether a ρ-competitive schedule exists for a given value of ρ. Theorem 1 establishes an upper bound of $3 + 2\sqrt{2}$ on the optimal competitive ratio, so we perform a bisection search in the range $[1, 3 + 2\sqrt{2}]$ to find the smallest ρ such that there exists a ρ-competitive schedule. We also output the resulting schedule.

The following lemma shows that the online strategy must eventually get to a sufficiently low-power state. Lemma 3 allows us to limit our concern to just the transition points in any online schedule.

LEMMA 2. *If $\mathcal{A} = (\mathcal{S}, \mathcal{T})$ is a ρ-competitive strategy and s_ℓ is the last state in $\mathcal{S}$, then $\kappa_\ell \leq \rho \cdot \kappa_k$.*

Proof. For the sake of contradiction, assume that $\kappa_\ell > \rho \cdot \kappa_k$. For $\mathcal{A}$ to be ρ-competitive, the function $\mathcal{A}(t)$ must lie entirely below $\rho \cdot OPT(t)$. However, the last line of $\rho \cdot OPT(t)$ has slope $\rho \cdot \kappa_k$ and will therefore intersect the last line of $\mathcal{A}(t)$, which has a larger slope κ_ℓ, after which time $\mathcal{A}(t)$ will exceed $\rho OPT(t)$. This is a contradiction. □

LEMMA 3. *If a schedule $\mathcal{A}$ has finite competitive ratio, then the earliest time $\bar{t} > 0$ at which $\frac{\mathcal{A}(t)}{OPT(t)}$ is maximized is a transition point in the strategy $\mathcal{A}$.*

Proof. Let $\rho = \max_{t>0} \frac{\mathcal{A}(t)}{OPT(t)}$. Consider the functions $\mathcal{A}(t)$ and $\rho OPT(t)$. The function $\mathcal{A}(t)$ never exceeds $\rho OPT(t)$, and $\bar{t}$ is the earliest point at which these two functions have the same value, not considering the origin. For the sake of contradiction, assume that $\bar{t}$ is not a transition point in $\mathcal{A}$. So we can find some small $\epsilon > 0$ such that $\mathcal{A}(t)$ is linear in $(\bar{t} - \epsilon, \bar{t} + \epsilon)$. Since $\mathcal{A}(t)$ is strictly less than $\rho OPT(t)$ in the interval $(\bar{t} - \epsilon, \bar{t})$ and $\mathcal{A}(\bar{t}) = \rho OPT(\bar{t})$, it must be the case that the slope of $\mathcal{A}(t)$ is larger than the slope of $\rho OPT(t)$ in this interval. This results in a contradiction, because $\mathcal{A}(t)$ has constant slope over $(\bar{t} - \epsilon, \bar{t} + \epsilon)$, and $\rho OPT(t)$ is a continuous function with decreasing slope, which means that $\mathcal{A}(t) > \rho OPT(t)$ for $t > \bar{t}$. □

We now explore ways to restrict the space of schedules we need to consider in searching for a ρ-competitive schedule. For a strategy $\mathcal{A} = (\mathcal{S}, \mathcal{T})$, we say that a

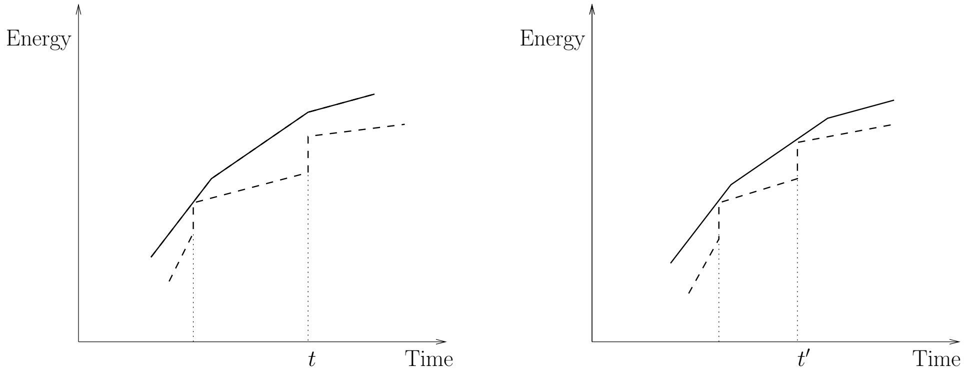


FIG. 2. *Energy consumed by the online and optimal strategies as a function of idle period length. The solid line is $\rho \cdot OPT(t)$. The dashed line is the online cost. t is the first transition time that is not eager. t' shows the transformed strategy which now has an eager transition.*

transition at time $t \in \mathcal{T}$ is *ρ-eager* (or just eager if ρ is clear from the context) if $\mathcal{A}(t) = \rho OPT(t)$. We say that $\mathcal{A}$ is a ρ-eager strategy if $\mathcal{A}(t) = \rho OPT(t)$ for every $t \in \mathcal{T}$. Note that by Lemmas 2 and 3, a ρ-eager strategy that ends at state s such that $\kappa_s \leq \rho \cdot \kappa_k$ is ρ-competitive.

LEMMA 4. *If $\mathcal{A} = (\mathcal{S}, \mathcal{T})$ is a ρ-competitive strategy, then there exists an eager strategy $\mathcal{A}' = (\mathcal{S}, \mathcal{T}')$ that is also ρ-competitive.*

Proof. Figure 2 shows a schematic of the proof. The jumps in the online cost (the dashed line) are transition costs. The solid line is $\rho OPT(t)$. The figure shows a transition time t at which the online cost is less than $\rho OPT(t)$. The idea is that we can slide such a transition time earlier until it hits the function $\rho OPT(t)$.

Consider the earliest transition time T which is not eager. Suppose that $\mathcal{A}$ transitions from state s_i to state s_j at time T. Let $T' < T$ be the time of the immediately preceding transition; if there is no such transition time, then set $T' = 0$. The function $\rho OPT(t) - \mathcal{A}(t)$ is continuous in the interval (T', T) since $\mathcal{A}$ does not have any transitions in this open interval, and $\rho OPT(t) - \mathcal{A}(t)$ is 0 at time T' and is strictly greater than $d_{i,j}$ at time $T - \epsilon$ for a small enough ϵ. Let $\overline{T}$ be the earliest time after T' such that $\rho OPT(t) - \mathcal{A}(t) = d_{i,j}$, so $\overline{T} < T$.

Consider the strategy $\mathcal{A}'$ that is identical to $\mathcal{A}$ except that the transition from s_i to s_j is moved earlier from T to $\overline{T}$. We need to argue that $\mathcal{A}'$ is ρ-competitive. Clearly $\mathcal{A}'(t) = A(t)$ for $t \in [T, \overline{T})$ and $\mathcal{A}(\overline{T}) = \rho OPT(\overline{T})$. Also $\mathcal{A}'(T) < \mathcal{A}(T)$ since $\mathcal{A}'$ transitions earlier to the low-power state s_j and hence uses less total energy, and since the strategies behave the same after time T, $\mathcal{A}'$ will continue to have a lower cost at all times $t > T$. To see that $\mathcal{A}'(t) \leq \rho OPT(t)$ over the interval $(\overline{T}, T)$, note that $\mathcal{A}'(t)$ is linear over this interval since $\mathcal{A}'$ remains in state s_j. Also $\rho OPT(t)$ is a piecewise-linear concave function since its slope is nonincreasing over time. Thus, since the points $(\overline{T}, \mathcal{A}'(\overline{T}))$ and $(T, \mathcal{A}'(T))$ both lie on or below this curve, the straight line connecting them lies under the curve $\rho OPT(t)$.

The procedure above can be repeated until all the transitions are eager. ∎

LEMMA 5. *Suppose a strategy makes a ρ-eager transition to state s_i at time t_i and next makes a transition to state s_j. Using the function $\rho OPT(t)$, one can compute the earliest ρ-eager transition time $\bar{t}$ to state s_j in time $O(\log k)$.*

Proof. Define the line $l(t) = \kappa_i t + \rho OPT(t_i) - \kappa_i t_i + d_{i,j}$. $\bar{t}$ is the smallest $t > t_i$ such that $\rho OPT(t) = l(t)$. If there is no such t, then a ρ-eager transition from s_i

to s_j does not exist. Since $\rho OPT(t)$ is concave, we have that if $l(t) < \rho OPT(t)$, or if $l(t) \geq \rho OPT(t)$ and the slope of $\rho OPT(t)$ is less than or equal to κ_i, then $\bar{t} \leq t$; otherwise $\bar{t} \geq t$. These inequalities allow one to do a binary search using the line segments of $\rho OPT(t)$ to determine $\bar{t}$ if it exists. Let s_ℓ be the optimal state (i.e., state of $OPT(t)$) at time t_i. Consider the line segments of $\rho OPT(t)$ corresponding to states s_ℓ and s_k. Recall that b_ℓ and b_k are, respectively, the left endpoints of these segments—these are the first time instants at which s_ℓ and s_k become the optimal states, respectively. Using the above inequalities, if we determine that $\bar{t} \geq b_k$, then $\bar{t}$ is simply the point of intersection (if it exists) of $l(t)$ with the segment (of $\rho OPT(t)$) corresponding to s_k. Otherwise we have a "low" segment with endpoint b_ℓ and a "high" segment with endpoint b_k. Now we repeatedly consider the left endpoint of the segment that is in the middle of the low and high segments, and we use the above inequalities to update the low or high segment and the corresponding endpoint accordingly, until the endpoints of the low and high segments correspond, respectively, to the left and right endpoints of a segment of $\rho OPT(t)$. When this happens we can compute $\bar{t}$ by finding the intersection point (if it exists) of $l(t)$ and this segment. The binary search can be implemented in time $\log k$, where k is the number of segments (i.e., number of states). □

Lemma 4 immediately gives an algorithm that is exponential in k, the number of states, and determines whether a ρ-competitive strategy exists for the system. This algorithm enumerates all subsequences of states and determines the ρ-eager strategy for that subsequence by finding the eager transition to each state based on the eager transitions to the previous states as described in the proof of Lemma 5. A ρ-competitive strategy for the system exists if and only if one of these ρ-eager strategies is ρ-competitive (i.e., ends at a state s with $\kappa_s \leq \rho \cdot \kappa_k$). The remainder of this section presents a way to remove the exponential dependence on k.

Let $S = \langle s_0, s_1, \ldots, s_k \rangle$ be a sequence of states that form a system. Define $S_{s_i \to s_j}$ to be the contiguous subsequence $\langle s_i, \ldots, s_j \rangle$, where s_i and s_j are elements of S such that $i < j$. Let Ψ_s be the set of subsequences of $S_{s_0 \to s}$ that include s_0 and s such that for each $\psi \in \Psi_s$, one can find transition times for the state sequence ψ so that in the resulting schedule, each transition up to and including the transition to state s is a ρ-eager transition. For a state $q \in \psi$, we will use $t_{\psi,q}$ to denote this ρ-eager transition time to q for the sequence ψ. (Note that ψ uniquely determines the transition times $t_{\psi,q}$.)

We define the *earliest* transition time $E(s,\rho)$ of state s for the given system as $E(s,\rho) = \min_{\psi \in \Psi_s} t_{\psi,s}$; that is, $E(s,\rho)$ is the earliest time at which any online strategy can transition to state s while remaining ρ-eager over all its transitions up to (and including) the transition to state s. Observe that if there is ρ-competitive strategy that uses state s, then by Lemma 4, there is such a ρ-eager strategy, so $\Psi_s \neq \phi$ and $E(s,\rho)$ is well defined. We call a transition to state s *ρ-early* (or simply early) if it happens at time $E(s,\rho)$. A strategy that consists entirely of early transitions is called a *ρ-early* strategy.

LEMMA 6. *If there is a ρ-competitive strategy $\mathcal{A} = (\mathcal{S}, \mathcal{T})$, then there is an eager and early ρ-competitive strategy.*

Proof. Let s be the last state in $\mathcal{S}$. Consider the sequence $\psi \in \Psi_s$ such that $t_{\psi,s} = E(s,\rho)$ and the strategy π uses only the states in ψ, transitioning to state $q \in \psi$ at time $t_{\psi,q}$, i.e., $\pi = \big(\psi, \{t_{\psi,q}\}_{q \in \psi}\big)$. Since $\mathcal{A}$ is ρ-competitive, it must be that $\kappa_s \leq \rho\kappa_k$ and since π by definition has all ρ-eager transitions and ends in state s, it is also ρ-competitive. We now argue that π is an early strategy. Note that π was chosen so that the transition to state s is ρ-early. We have to show that the remaining transitions of π are also ρ-early.

Suppose not. Consider the latest transition that is not ρ-early. Suppose this happens for state r ($\neq s$), so $T = t_{\psi,r} > E(r,\rho)$. Let r' be the state just after r in sequence ψ. Let $\psi' \in \Psi_r$ be the sequence for which $t_{\psi',r} = E(r,\rho) = T'$. T' is the earliest time that a ρ-eager schedule can transition to state r and the sequence of states in this schedule is given by ψ'. Consider the hybrid strategy π' that uses the states in ψ' followed by the states in ψ that appear after r, with the transition times being $t_{\psi',q}$ for $q \in \psi'$ and $t_{\psi,q}$ for $q \in \psi_{r' \to s}$. Strategy π transitions to state r at time T and strategy π' transitions to state r at time $T' < T$. Both of these transitions are eager transitions. Both strategies are in state r at time T and make the same state transitions thereafter. Thus, for any $t \geq T$, $\pi(t) - \pi(T) = \pi'(t) - \pi'(T)$. In particular, both strategies transition to r' (the state after r) at time $t_{\psi,r'} = E(r',\rho) = T''$. Using the equation above we have that $\pi'(T'') = \pi(T'') - \big(\pi(T) - \pi'(T)\big)$. We will show that $\pi'(T) < \pi(T)$, which implies, in particular, that $\pi'(T'') < \pi(T'')$. So in π' the transition to r' is no longer ρ-eager. Arguing as in Lemma 4 this means that we can shift the transition to r' to get an eager transition at an *earlier* time. But this contradicts the assumption that the transition to state r' at time T'' was an early transition.

We now prove that $\pi'(T) < \pi(T)$. This proof is illustrated in Figure 3. The transitions to state r in schedules π and π' are eager transitions, so both the points $(T', \pi'(T'))$ and $(T, \pi(T))$ lie on the $\rho OPT(t)$ curve. Since $\pi(t) < \rho OPT(t)$ for all t, the the slope of $\rho OPT(t)$ at time T is at least κ_r, the slope of $\pi(t)$ at time T, and strictly greater since the gap between $\rho OPT(t)$ and $\pi(t)$ must accommodate the transition cost from state r to r' at time T''. The concavity of $\rho OPT(t)$ implies that its slope is greater than κ_r over the interval $[T', T]$. So $\pi(T) = \rho OPT(T) > \rho OPT(T') + \kappa_r(T - T') = \pi'(T)$, where the last inequality follows since π' stays in state r in the interval $[T', T]$. □

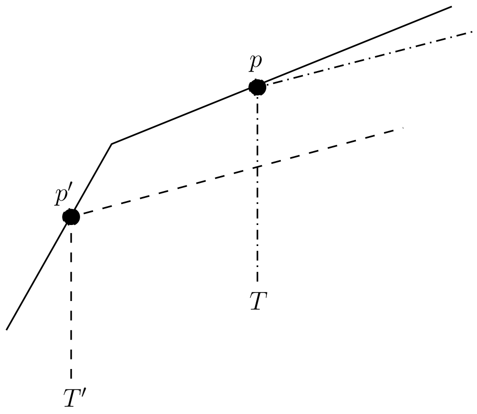


FIG. 3. *The solid line is $\rho \cdot OPT$. The dashed line is the schedule π' from Lemma 6, and the dashed/dotted line is π. The point labeled p is $(T, \pi(T))$, and p' is $(T', \pi'(T'))$. The idea is to show that at time T, π' has a lower cost than π.*

From Lemma 6 we can deduce that we need only to consider a specific early and eager schedule, the one that is determined by the $E(.,\rho)$ values, to determine if a ρ-competitive strategy exists. We can now define a decision procedure EXISTS that takes a system and a constant ρ and outputs YES if a ρ-competitive strategy exists for the system, and NO otherwise. The procedure can be modified to also output a ρ-competitive strategy (if it exists). We employ a dynamic programming approach to calculate $E(s_i, \rho)$ for $0 < i \leq k$. We always start with the high-power state, and hence $E(s_0, \rho) = 0$. Suppose we have computed $E(s_j, \rho)$ for all $j = 0, \dots, i-1$. Let t_j be the earliest time at which the system ρ-eagerly transitions from s_j to s_i given that the transition to s_j is ρ-eager and occurs at time $E(s_j, \rho)$. If such a transition is not possible, then we assign $t_j = \infty$. We can compute t_j in $O(\log k)$ time as described in Lemma 5. Then, $E(s_i, \rho) = \min_{j<i} t_j$. Determining each $E(s_i, \rho)$ requires examining j different possibilities, so finding all the early transition times for all states takes time $O(k^2 \log k)$. By Lemma 2, we know that if $E(s_i, \rho)$ is finite for some state s_i, where $\kappa_i \leq \rho \cdot \kappa_k$, we know that a ρ-competitive strategy exists. One can quickly elicit the schedule by starting from state k and retracing the states that minimized the earliest transition time. We use the procedure EXISTS to do a bisection search in the interval $[1, 3 + 2\sqrt{2}]$ and find a ρ-competitive strategy where $\rho \leq \rho^* + \epsilon$. The total time taken is $O(k^2 \log k \log(1/\epsilon))$.

We now turn our attention to adapting this dynamic programming technique to solve $PD(k, m)$, where a bound of m is specified on the number of states that can be used by the online algorithm. We introduce a new parameter b, modifying our function to $E(s_i, \rho, b)$, where $0 \leq b \leq \min(i, m)$. The intuition is that function E is now required to return the earliest time when the system can transition to state s_i while staying entirely below $\rho OPT(t)$ and using at most $b+1$ states from $\langle s_0, \dots, s_i \rangle$. The base case is $E(s_0, \rho, b) = 0$ for all $b \geq 0$. Intuitively, $E(s_i, \rho, b)$ is determined by the "best" state s_j prior to s_i such that at most $b-1$ states were used to reach s_j. Notice that for any given state s_i and fixed ρ, $E(s_j, \rho, b)$ is nonincreasing as b increases. Therefore, as above we can write $E(s_i, \rho, b) = \min_{j<i} t'_j$, where t'_j is the earliest time when the system ρ-eagerly transitions from s_j to s_i given that the transition to s_j was ρ-eager and occurred at $E(s_j, \rho, b-1)$. The running time increases by a factor of m now and is $O(k^2 m (\log k) \log(1/\epsilon))$.

6. A probability-based algorithm.

Karlin et al. study the two-state case when the length of the idle period is generated by a known probability distribution p [8]. (Although they examined the problem in the context of the spin-block problem, their problem is identical to our two-state case.) They observed that the expected cost of the online strategy that makes the transition to the sleep state at time T is

$$\int_0^T p(t)(\kappa_0 t)dt + \int_T^\infty p(t)\Big(\kappa_0 T + \kappa_1(t-T) + \beta\Big)dt, \tag{3}$$

where κ_0 is the power-consumption rate in the active state, κ_1 is the power-consumption rate in the sleep state, and β is the transition cost between the two states. The online strategy then should select the transition time T that minimizes this cost.

The multistate case presents two distinct challenges. The first is to determine the optimal sequence of states through which an online strategy should transition throughout the course of the idle period. Then, once this sequence has been determined, the optimal transition times need to be determined. Our proof proceeds by establishing that the only transition times that need to be considered are the optimal transition times for two-state systems. Suppose, for example, that we are considering a sequence of state transitions in which state s_i is followed by state s_j. Let $T_{i,j}$ denote the optimal transition time from state s_i to s_j if these were the only two states in the system (that is, if s_i were the active state and s_j were the only sleep state). Note that $T_{i,j}$ can be determined by the expression above. We establish that regardless of the rest of the sequence, the optimal transition point from state s_i to s_j is $T_{i,j}$. We call the $T_{i,j}$'s the *pairwise-optimal* transition times.

Lemmas 7 and 8 establish that the pairwise-optimal transition times happen in the right order. That is, for $i < k < j$, $T_{i,k} \leq T_{k,j}$. If this is not the case, then any subsequence that has s_i followed by s_k followed by s_j cannot possibly be the best sequence of states. Note that the $T_{i,j}$'s may not necessarily be unique. In general, we will select the earliest transition time that minimizes the cost for the two-state system.

Lemma 9 then shows that as long as the pairwise-optimal transition times are in the right order, they give the globally optimal set of transition times for that subsequence. Our algorithm then uses this fact to find the optimal sequence of states by dynamic programming. Note that it is not necessary to exhaustively consider all possible subsequences.

6.1. Optimal transition times. Consider a particular subsequence of $l+1$ states $s_{a_0}, \ldots s_{a_l}$. In order to avoid the double subscripts, throughout this subsection we will rename our subsequence $q_0, q_1, \ldots, q_l$. Since the strategy must start in state s_0, we can assume that $q_0 = s_0$. For $i < j$, define $\beta_{i,j}$ to be the cost to transition from state q_i to state q_j, that is, $\beta_{i,j} = d_{a_i,a_j}$. Furthermore, we will refer to the power-consumption rate of state q_i as α_i, that is, $\alpha_i = \kappa_{a_i}$.

We will consider the strategy that transitions through the states in the subsequence $q_0, q_1, \ldots, q_l$. Suppose that we use transition time T_i to transition from state q_{i-1} to state q_i. It will be convenient for notation to define $T_{l+1} = \infty$ and $T_0 = 0$. The cost of the strategy that uses these transition times is

$$\begin{aligned} cost(T_1, \ldots, T_l) = &\sum_{j=1}^{l+1} \int_{T_{j-1}}^{T_j} p(t)\alpha_{j-1}(t - T_{j-1})dt \\ &+ \sum_{j=1}^{l} \int_{T_j}^{\infty} p(t)\Big(\alpha_{j-1}(T_j - T_{j-1}) + \beta_{j-1,j}\Big)dt. \end{aligned} \tag{4}$$

The goal is to pick the $T_1, \ldots, T_l$ so as to minimize the above cost. This is the optimal cost for the subsequence $q_0, \ldots, q_l$.

For each $i \in \{1, \ldots, l\}$, let $\gamma_i = \frac{\alpha_{i-1} - \alpha_i}{\beta_{i-1,i}}$.

LEMMA 7. *Suppose that there is an $i < j$ such that $\gamma_i < \gamma_j$; then there is a strict subsequence of $q_0, \ldots, q_l$ whose optimal cost is no greater than the optimal cost for $q_0, \ldots, q_l$.*

Proof. Consider the first j such that $\gamma_{j-1} < \gamma_j$. Let $(\bar{t}_1, \ldots \bar{t}_{j-1}, \bar{t}_j, \ldots, \bar{t}_l)$ be the sequence of thresholds that minimizes the cost of this sequence of states. Define the following quantities:

$$\begin{aligned} F_{j-1,j} &= cost(\bar{t}_1, \ldots \bar{t}_{j-2}, \bar{t}_{j-1}, \bar{t}_j, \bar{t}_{j+1} \ldots, \bar{t}_l), \\ F_{j-1,j-1} &= cost(\bar{t}_1, \ldots \bar{t}_{j-2}, \bar{t}_{j-1}, \bar{t}_{j-1}, \bar{t}_{j+1} \ldots, \bar{t}_l), \\ F_{j,j} &= cost(\bar{t}_1, \ldots \bar{t}_{j-2}, \bar{t}_j, \bar{t}_j, \bar{t}_{j+1} \ldots, \bar{t}_l). \end{aligned}$$

We will show that $F_{j-1,j}$ is greater than or equal to a weighted average of $F_{j-1,j-1}$ and $F_{j,j}$, which means that it must be greater than or equal to at least one of these values. This means that the strategy that transitions from state q_{j-2} to state q_{j-1} and then immediately transitions to state q_j at time either $\bar{t}_{j-1}$ or $\bar{t}_j$ is at least as good as the original strategy. Since $\beta_{j-2,j} \leq \beta_{j-2,j-1} + \beta_{j-1,j}$, skipping state $j-1$ altogether can only improve the strategy.

Below we have an expression for $F_{j,j} - F_{j-1,j}$ which can be derived from the definition for the cost in (4). Under $F_{j,j}$ the transition from state q_{j-2} to state q_{j-1} is moved forward from time $\bar{t}_{j-1}$ to time $\bar{t}_j$. Any time spent in the interval $[\bar{t}_{j-1}, \bar{t}_j]$ happens at the higher-power rate of α_{j-2} instead of α_{j-1}. This is accounted for in the first two terms of the sum. However, idle times ending in the interval $[\bar{t}_{j-1}, \bar{t}_j]$ save on the transition cost, which is accounted for in the last term below:

$$F_{j,j} - F_{j-1,j} = \int_{\bar{t}_{j-1}}^{\bar{t}_j} p(t)(t-\bar{t}_{j-1})(\alpha_{j-2}-\alpha_{j-1})dt + \int_{\bar{t}_j}^{\infty} p(t)(\bar{t}_j-\bar{t}_{j-1})(\alpha_{j-2}-\alpha_{j-1})dt - \int_{\bar{t}_{j-1}}^{\bar{t}_j} \beta_{j-2,j-1} p(t)dt.$$

Dividing by $(\alpha_{j-2} - \alpha_{j-1})$, this becomes

$$(5)\quad \frac{F_{j,j} - F_{j-1,j}}{\alpha_{j-2} - \alpha_{j-1}} = \int_{\bar{t}_{j-1}}^{\bar{t}_j} p(t)(t - \bar{t}_{j-1})dt + \int_{\bar{t}_j}^{\infty} p(t)(\bar{t}_j - \bar{t}_{j-1})dt - \int_{\bar{t}_{j-1}}^{\bar{t}_j} \frac{1}{\gamma_{j-1}} p(t)dt.$$

Below, we use the definition of cost in (4) to get an expression for $F_{j-1,j} - F_{j-1,j-1}$. Note that in $F_{j-1,j-1}$, the transition from state q_{j-1} to state q is moved back from time $\bar{t}_j$ to time $\bar{t}_{j-1}$. Thus, $F_{j-1,j}$ will spend $\alpha_{j-1} - \alpha_j$ more power than $F_{j-1,j-1}$ for any time spent in the interval $[\bar{t}_{j-1}, \bar{t}_j]$. Furthermore, $F_{j-1,j-1}$ will have an additional transition cost of $\beta_{j-1,j}$ for those intervals that end in the period $[\bar{t}_{j-1}, \bar{t}_j]$:

$$F_{j-1,j} - F_{j-1,j-1} = \int_{\bar{t}_{j-1}}^{\bar{t}_j} p(t)(t-\bar{t}_{j-1})(\alpha_{j-1}-\alpha_j)dt + \int_{\bar{t}_j}^{\infty} p(t)(\bar{t}_j-\bar{t}_{j-1})(\alpha_{j-1}-\alpha_j)dt - \int_{\bar{t}_{j-1}}^{\bar{t}_j} \beta_{j-1,j} p(t)dt.$$

Dividing by $(\alpha_{j-1} - \alpha_j)$, this becomes

$$(6)\quad \frac{F_{j-1,j} - F_{j-1,j-1}}{\alpha_{j-1} - \alpha_j} = \int_{\bar{t}_{j-1}}^{\bar{t}_j} p(t)(t-\bar{t}_{j-1})dt + \int_{\bar{t}_j}^{\infty} p(t)(\bar{t}_j-\bar{t}_{j-1})dt - \int_{\bar{t}_{j-1}}^{\bar{t}_j} \frac{1}{\gamma_j} p(t)dt.$$

Comparing, (5) and (6), the expressions are almost identical except for the γ in the last term. Since $\gamma_{j-1} < \gamma_j$ and $\int_{\bar{t}_{j-1}}^{\bar{t}_j} p(t)dt \geq 0$, we have that

$$\frac{F_{j,j} - F_{j-1,j}}{\alpha_{j-2} - \alpha_{j-1}} \leq \frac{F_{j-1,j} - F_{j-1,j-1}}{\alpha_{j-1} - \alpha_j}.$$

Let $\omega_1 = 1/(\alpha_{j-2} - \alpha_{j-1})$ and $\omega_2 = 1/(\alpha_{j-1} - \alpha_j)$. Note that both ω_1 and ω_2 are at least 0. Rearranging, we get that

$$\left(\frac{\omega_1}{\omega_1 + \omega_2}\right) F_{j,j} + \left(\frac{\omega_2}{\omega_1 + \omega_2}\right) F_{j-1,j-1} \leq F_{j-1,j}. \qquad \square$$

Now suppose that we consider only the two-state system consisting of state q_{i-1} and state q_i. We will let τ_i denote the optimal threshold time if these are the only two states in the system. We have that τ_i is the time T that minimizes

$$\int_0^T p(t)\alpha_{i-1} t dt + \int_T^{\infty} p(t)\Big(\alpha_{i-1}T + \alpha_i(t-T) + \beta_{i-1,i}\Big)dt.$$

Note that the value of T that results in the minimum above may not be unique. In this case, we take τ to be the smallest value which achieves the minimum. Also note that, by subtracting the term $\int_0^\infty p(t)\alpha_i tdt$ (which is independent of T) and dividing by $\beta_{i-1,i}$ in the above definition, it can be seen that $\tau_i = \arg\min_T f(\gamma_i, T)$, where

$$f(\gamma, T) = \int_0^T p(t)\gamma tdt + \int_T^\infty p(t)(\gamma T + 1)dt.$$

Note that for a two-state system whose active state and sleep states have power-consumption rates of γ and 0, respectively, and whose transition cost is 1, $f(\gamma, T)$ denotes the expected power consumed by an online strategy that transitions to the sleep state at time T. We will show that for a particular subsequence of states, if we minimize the cost over all choices for the thresholds, the resulting thresholds are those obtained by the pairwise optimization above. First, however, we must establish that the τ_i values have the correct ordering.

LEMMA 8. *If $\gamma_i > \gamma_{i+1}$, then $\tau_i \leq \tau_{i+1}$.*

Proof. Intuitively, γ_i is the ratio of the additional power cost of being in state q_i instead of state q_{i-1} over the transition costs between the two states. It stands to reason that the larger this cost is, the sooner one would want to transition from state q_{i-1} to state q_i.

We will formalize this argument using a proof by contradiction. Suppose that we have $\tau_i > \tau_{i+1}$ and $\gamma_i > \gamma_{i+1}$. The proof will make use of the definition of $f(\gamma, T)$ given above. τ_i is the smallest value for T which attains the minimum of $f(\gamma_i, T)$. Since $\tau_{i+1} < \tau_i$, we know that $f(\gamma_i, \tau_{i+1}) > f(\gamma_i, \tau_i)$. By the definition of τ_{i+1}, we have that $f(\gamma_{i+1}, \tau_i) \geq f(\gamma_{i+1}, \tau_{i+1})$. Thus, it should be the case that

$$f(\gamma_{i+1}, \tau_i) - f(\gamma_{i+1}, \tau_{i+1}) \geq 0 > f(\gamma_i, \tau_i) - f(\gamma_i, \tau_{i+1}). \tag{7}$$

Using the definition of $f(\gamma, T)$ above, for any $T_1 < T_2$,

$$f(\gamma, T_2) - f(\gamma, T_1) \;=\; \gamma\left[\int_{T_1}^{T_2} p(t)(t - T_1)dt + \int_{T_2}^\infty p(t)(T_2 - T_1)dt\right] - \int_{T_1}^{T_2} p(t)dt.$$

The quantity inside the square braces above is nonnegative. This implies that the quantity $f(\gamma, T_2) - f(\gamma, T_1)$ is nondecreasing in γ. This, however, contradicts inequality (7) and the fact that $\gamma_i > \gamma_{i+1}$. □

Finally, we prove the main lemma which states that the transition times are simultaneously optimized at the pairwise-optimal transition points.

LEMMA 9. *For a given subsequence of states $q_0, \ldots, q_l$, if $\tau_{i-1} < \tau_i$ for all $i \in \{1, \ldots, l\}$, then the minimum total cost is achieved for cost$(\tau_1, \ldots, \tau_l)$.*

Proof. The basic idea is that we can interpret $cost(T_1, \ldots, T_l) - \int_0^\infty p(t)\alpha_l tdt$ as the sum of the power consumed in l two-state systems, where the ith system (for $i = 1, \ldots, l$) has states whose power-consumption rates are $(\alpha_{i-1} - \alpha_i)$ and 0, and the cost to transition between the two is $\beta_{i-1,i}$. Note that $\int_0^\infty p(t)\alpha_l tdt$ is a constant, independent of the choice of T_i's. After rescaling, one can write this expression as a linear combination of the $f(\gamma_i, T_i)$ terms. Since τ_i minimizes $f(\gamma_i, T)$, and the τ_i values have the right ordering, this implies that $cost(T_1, \ldots, T_l)$ is minimized by setting $T_i = \tau_i$ for $i = 1, \ldots, l$.

We will establish below that we can rewrite (4) as follows:

$$\begin{aligned}(8)\quad cost(T_1,\dots,T_l) &= \int_0^\infty p(t)\alpha_l t dt \\ &+ \sum_{i=1}^{l}\left[\int_0^{T_i} p(t)(\alpha_{i-1}-\alpha_i)tdt + \int_{T_i}^\infty p(t)\Big((\alpha_{i-1}-\alpha_i)T_i + \beta_{i-1,i}\Big)dt\right].\end{aligned}$$

So, by rescaling, we get that

$$cost(T_1,\dots,T_l) - \int_0^\infty p(t)\alpha_l tdt = \sum_{i=1}^{l}\beta_{i-1,i}f(\gamma_i,T_i).$$

We want to choose $T_1 \le \cdots \le T_l$ to minimize this expression. Since $\tau_1 \le \cdots \le \tau_l$ and each $\tau_i = \arg\min_T f(\gamma_i,T)$ it follows that the minimum is attained by setting $T_i = \tau_i$ for each i.

To complete the proof we show the equivalence of (4) and (8). It suffices to show that (4) and (8) integrate the same expression over each interval $[T_{i-1},T_i)$ for $i = 1,\dots,l+1$. The integrand in (4) over the interval $[T_{i-1},T_i)$ is

$$p(t)\left[\alpha_{i-1}(t-T_{i-1}) + \sum_{j=1}^{i-1}\Big(\alpha_{j-1}(T_j - T_{j-1}) + \beta_{j-1,j}\Big)\right],$$

and the integrand in (8) is

$$(9)\qquad p(t)\left[\sum_{j=1}^{i-1}\Big((\alpha_{j-1}-\alpha_j)T_j + \beta_{j-1,j}\Big) + \Big(\sum_{j=i}^{l}(\alpha_{j-1}-\alpha_j) + \alpha_l\Big)t\right].$$

The summations over the j indices in (9) telescope to show that the two expressions are identical. □

6.2. The optimal state sequence. We now present a simple polynomial time algorithm to obtain the optimal state sequence for a given system. First, for each pair (i,j), $0 \le i < j \le k$, let $T_{i,j}$ denote the optimal transition point if s_i and s_j were the only two states in the system. The time complexity of determining a single $T_{i,j}$ depends on the representation of the probability distribution. In practice, this is most likely to be estimated by a finite histogram with B bins starting at time 0 and sampled at a uniform discrete interval of δ. It follows that bin i corresponds to time δi. It is not difficult to generalize this for variable-sized bins. We will also assume that all transition times occur at some δi. The height of bin i is $H(i)$, and this implies that the probability that the idle time t equals δi is given by $\frac{H(i)}{\sum_i H(i)}$. In Algorithm 1, we calculate $ACC[i]$ and $ACCT[i]$ values, which are $\int_0^{i\delta} p(t)dt$, and $\int_0^{i\delta} tp(t)dt$ and we then use them to evaluate $T_{i,j}$ values. We can rewrite the expression for the cost of a two-state system in (3) as

$$\kappa_i\int_0^T p(t)tdt + \kappa_j\int_T^\infty p(t)tdt + \Big((\kappa_i-\kappa_j)T + \beta_{i,j}\Big)\int_T^\infty p(t)dt.$$

We also denote $\int_0^{B\delta} p(t)dt$ and $\int_0^{B\delta} tp(t)dt$ as $TOTAL$ and $TOTALT$, respectively. Using the precalculated values above, the cost of transitioning from state s_i to state s_j at time δl is

$$\kappa_i \cdot ACCT[l] + (\kappa_i l\delta - \kappa_j l\delta + \beta_{i,j})(TOTAL - ACC[l]) + \kappa_j(TOTALT - ACCT[l]).$$

Algorithm 1. Evaluating $T_{i,j}$ values.

```
$ACC[0] \leftarrow H[0]$
$ACCT[0] \leftarrow 0$
for $k = 1$ to $B$ do
  $ACC[k] \leftarrow ACC[k-1] + H[k]$
  $ACCT[k] \leftarrow ACC[k-1] + H[k] \times k \cdot \delta$
end for
$TOTAL \leftarrow ACC[B]$
$TOTALT \leftarrow ACCT[B]$
for all $(i, j)$ pairs such that $0 \leq i < j \leq k$ do
  min$\leftarrow \infty$, argmin $\leftarrow -1$
  for $l = 0$ to $B - 1$ do
    $val = \kappa_i \cdot ACCT[l] + (\kappa_i l\delta - \kappa_j l\delta + \beta_{i,j})(TOTAL - ACC[l])$
          $+\kappa_j(TOTALT - ACCT[l])$
    if $val <$ min then
      min $\leftarrow val$
      argmin $\leftarrow l$
    end if
  end for
  $T_{i,j} \leftarrow$ argmin $\cdot \delta$
end for
```

Once the $T_{i,j}$'s are found, we sweep through them in nondecreasing order, keeping a running tab of the best subschedules that we can achieve ending in each state s_i at each point in time. When we encounter a $T_{i,j}$, we check to see if transitioning from s_i to s_j can improve the current best subschedule ending in s_j, and if it does, we update our data structure to reflect it.

A given strategy divides time into intervals, where each interval is the period of time spent in a particular state. The expected cost for a strategy given in (4) is obtained by summing over the expected cost incurred in each interval. The cost for each interval is divided into two parts, which results in two separate summations in (4). We define the function Q for the first term, which is

$$Q(t_s, j, t_f) = \int_{t_s}^{t_f} p(t)\kappa_i(t - t_s)dt.$$

This is the expected cost of staying in state s_i in the interval $[t_s, t_f)$ for those idle periods whose length is also in the interval $[t_s, t_f)$. Define

$$R(i, t_s, j, t_f) = \int_{t_f}^{\infty} p(t)\Big(\alpha_i(t_f - t_s) + \beta_{i,j}\Big)dt.$$

This is the expected cost for those intervals longer than t_f of staying in state s_i over the time period $[t_s, t_f)$ and then transitioning to state s_j. Note that $Q(\delta l_i, j, \delta l_j)$ and $R(i, \delta l_i, j, \delta l_j)$ can both be evaluated in constant time given $ACC[l_i]$, $ACC[l_j]$, $ACCT[l_i]$, and $ACCT[l_j]$ defined above.

At each transition $T_{i,j}$, we check to see if the current best schedule that ends in state s_j can be improved by transitioning to j from the current best schedule that ends in state s_i. For this purpose, we maintain two arrays of size $k+1$: $t[i]$ is the time at which the current best schedule that ends at state s_i transitions to s_i, and $h[i]$ is the cost at $t[i]$ of that schedule. Initially, $h[0] \leftarrow 0$ and all other $h[i] \leftarrow \infty$. $t[i]$ for all i can be initialized to 0. In Procedure 2, we provide the pseudocode for processing at each transition point $T_{i,j}$.

PROCEDURE 2. Processing $T_{i,j}$ in the line sweep algorithm.

```
**Current Status:** $T_{i,j}$ is the transition point that is being processed
{The cost up to time $T_{i,j}$ if transitioning from $i$ to $j$ at $T_{i,j}$}
$h1 \leftarrow h[i] + Q(t[i], j, T_{i,j}) + R(i, t[i], j, T_{i,j})$
{The cost up to time $T_{i,j}$ if transitioning to $j$ at the current best time of $t[j]$}
$h2 \leftarrow h[j] + Q(t[j], j, T_{i,j}) + R(j, t[j], j, T_{i,j})$
if $h1 < h2$ then
  $h[j] \leftarrow h1$
  $t[j] \leftarrow T_{i,j}$
end if
```

It is easy to see that each transition point takes a constant amount of processing. The sorting takes an overhead of $O(k^2 \log k)$. The initial preprocessing to calculate the transition points takes $O(k^2 B)$. Hence, the total running time is $O(k^2(\log k + B))$.

The algorithm can be easily extended to find the algorithm that minimizes the expected cost subject to the constraint that only m states are ever reached. We maintain $t[i, b]$ and $h[i, b]$ for all states s_i and $b < \min\{m, i\}$. These are the best time and energy required to reach state i subject to at most b states being reached. The algorithm is given below in Procedure 3.

PROCEDURE 3. Processing $T_{i,j}$ in the line sweep algorithm with the number of states constrained.

```
**Current Status:** $T_{i,j}$ is the transition point that is being processed
for $b = 1 \ldots j-1$ do
  $h1 \leftarrow h[i, b-1] + Q(t[i, b-1], j, T_{i,j}) + R(i, t[i, b-1], j, T_{i,j})$
  $h2 \leftarrow h[j, b] + Q(t[j, b], j, T_{i,j}) + R(j, t[j, b], j, T_{i,j})$
  if $h1 < h2$ then
    $h[j, b] \leftarrow h1$
    $t[j, b] \leftarrow T_{i,j}$
  end if
end for
```

REFERENCES

[1] C. ANDERSON AND A. KARLIN, *Two adaptive hybrid cache coherency protocols*, in Proceedings of the Second International Symposium on High-Performance Computer Architecture, San Jose, CA, 1996, pp. 303–313.

[2] Y. AZAR, Y. BARTAL, E. FEUERSTEIN, A. FIAT, S. LEONARDI, AND A. ROSEN, *On capital investment*, Algorithmica, 25 (1999), pp. 22–36.

[3] L. BENINI, A. BOGLIOLO, AND G. DE MICHELI, *A survey of design techniques for system-level dynamic power management*, IEEE Trans. Very Large Scale Integration (TVLSI) Systems, 8 (2000), pp. 299–316.

[4] P. DAMASCHKE, *Nearly optimal strategies for special cases of on-line capital investment*, Theoret. Comput. Sci., 302 (2003), pp. 35–44.

[5] S. J. Eggers and R. H. Katz, *Evaluating the performance of four snooping cache coherency protocols*, in Proceedings of the 16th Annual International Symposium on Computer Architecture, ACM, New York, 1989, pp. 2–15.

[6] S. Irani, R. Gupta, and S. Shukla, *Competitive analysis of dynamic power management strategies for systems with multiple power savings states*, in Proceedings of the IEEE Conference on Design, Automation and Test in Europe, IEEE Computer Society, Washington, DC, 2002, p. 117.

[7] S. Irani, S. Shukla, and R. Gupta, *Online strategies for dynamic power management in systems with multiple power saving states*, Trans. on Embedded Computing Sys., Special Issue on Power Aware Embedded Computing, 8 (2003), pp. 325–346.

[8] A. Karlin, M. Manasse, L. McGeoch, and S. Owicki, *Randomized competitive algorithms for non-uniform problems*, in Proceedings of the ACM–SIAM Symposium on Discrete Algorithms, ACM, New York, SIAM, Philadelphia, 1990, pp. 301–309.

[9] A. R. Karlin, C. Kenyon, and D. Randall, *Dynamic tcp acknowledgement and other stories about $e/(e-1)$*, in Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, ACM, New York, 2001, pp. 502–509.

[10] A. R. Karlin, K. Li, M. S. Manasse, and S. Owicki, *Empirical studies of competitve spinning for a shared-memory multiprocessor*, in Proceedings of the 13th ACM Symposium on Operating Systems Principles, ACM, New York, 1991, pp. 41–55.

[11] S. Keshav, C. Lund, S. Phillips, N. Reingold, and H. Saran, *An empirical evaluation of virtual circuit holding time policies in ip-over-atm networks*, IEEE J. Selected Areas in Communications, 13 (1995), pp. 1371–1382.

SIAM J. COMPUT.
Vol. 37, No. 5, pp. 1517–1535

SPLITTING NP-COMPLETE SETS*

CHRISTIAN GLAßER†, A. PAVAN‡, ALAN L. SELMAN§, AND LIYU ZHANG¶

Abstract. We show that a set is m-autoreducible if and only if it is m-mitotic. This solves a long-standing open question in a surprising way. As a consequence of this unconditional result and recent work by Glaßer et al., complete sets for all of the following complexity classes are m-mitotic: NP, coNP, ⊕P, PSPACE, and NEXP, as well as all levels of PH, MODPH, and the Boolean hierarchy over NP. In the cases of NP, PSPACE, NEXP, and PH, this at once answers several well-studied open questions. These results tell us that complete sets share a redundancy that was not known before. In particular, every NP-complete set A splits into two NP-complete sets A_1 and A_2. We disprove the equivalence between autoreducibility and mitoticity for all polynomial-time-bounded reducibilities between 3-tt-reducibility and Turing-reducibility: There exists a sparse set in EXP that is polynomial-time 3-tt-autoreducible, but not weakly polynomial-time T-mitotic. In particular, polynomial-time T-autoreducibility does not imply polynomial-time weak T-mitoticity, which solves an open question by Buhrman and Torenvliet.

Key words. computational and structural complexity, NP-complete sets, autoreducibility, mitoticity

AMS subject classifications. 68Q17, 68Q15

DOI. 10.1137/060673886

1. Introduction.

We show in this paper that NP-complete sets split into two equivalent parts. Let L be an NP-complete set containing an infinite number of strings. Then there is a set $S \in \mathrm{P}$ such that the sets $L_1 = S \cap L$ and $L_2 = \overline{S} \cap L$ are both NP-complete, and $L = L_1 \cup L_2$. Since L_1 and L_2 are both many-one-equivalent to L, we may say that they contain the same information as L. For this reason, sets L with this property are called *mitotic*.[1] Briefly, we prove that all NP-complete sets are mitotic.

Our story begins with the notion of autoreducibility. Trakhtenbrot [19] defined a set A to be *autoreducible* if there is an oracle Turing machine M such that $A = L(M^A)$ and M on input x never queries x. For complexity classes like NP and PSPACE, refined measures are needed. In this spirit, Ambos-Spies [1] defined the notion of polynomial-time autoreducibility and the more restricted form m-autoreducibility. A set A is *polynomial-time autoreducible* if it is autoreducible via an oracle Turing machine that runs in polynomial time. A is *m-autoreducible* if A is polynomial-time many-one reducible to A via a function f such that $f(x) \neq x$ for every x. Autore-

*Received by the editors November 1, 2006; accepted for publication (in revised form) October 19, 2007; published electronically February 1, 2008. A preliminary version of this paper appeared at the 23rd International Symposium on Theoretical Aspects of Computer Science (STACS 2006), Marseille, France, published in Lecture Notes in Comput. Sci. 3884, Springer, Berlin, pp. 444–454.

http://www.siam.org/journals/sicomp/37-5/67388.html

†Lehrstuhl für Informatik IV, Universität Würzburg, 97074 Würzburg, Germany (glasser@informatik.uni-wuerzburg.de).

‡Department of Computer Science, Iowa State University, Ames, IA 50010 (pavan@cs.iastate.edu). This author's research was supported in part by NSF grants CCR-0344817 and CCF-0430807.

§Department of Computer Science and Engineering, State University of New York at Buffalo, Buffalo, NY 14260 (selman@cse.buffalo.edu). This author's research was supported in part by NSF grant CCR-0307077 and by the Alexander von Humboldt-Stiftung.

¶Department of Computer Science and Information Systems, University of Texas at Brownsville, Brownsville, TX 78520 (Liyu.Zhang@utb.edu).

[1] Mitosis in biology is the process by which a cell separates its duplicated genome into two identical halves.

ducible sets contain redundant information just as do mitotic sets. For example, if a set A is m-autoreducible, then x and $f(x)$ contain the same information about membership in A.

A recent paper of Glaßer et al. [11] showed that the complete sets for many interesting classes such as NP, PSPACE, NEXP, and levels of PH are m-autoreducible. The main technical result of the present paper is that m-autoreducible implies m-mitotic. As a consequence, complete sets for interesting complexity classes such as NP, PSPACE, NEXP, and levels of PH are m-mitotic. This result resolves several long-standing open questions raised by Ambos-Spies [1], Buhrman, Hoene, and Torenvliet [5], and Buhrman and Torenvliet [6].

The notion of *mitoticity* was originally introduced by Lachlan [14] for recursively enumerable sets. Mitoticity was studied comprehensively by Ladner [16, 15]. Ambos-Spies [1] formulated two related notions in the polynomial-time setting, mitoticity and weak mitoticity. A set A is *m-mitotic* if there is a set $S \in \mathrm{P}$ such that A, $A \cap S$, and $A \cap \overline{S}$ are polynomial-time many-one equivalent. If the latter holds without the assumption $S \in \mathrm{P}$, then A is *weakly m-mitotic.*

Ambos-Spies [1] showed that if a set is m-mitotic, then it is m-autoreducible, and he raised the question of whether the converse holds. As stated above, we resolve this question and show that every m-autoreducible set is m-mitotic. Since its proof is technically involved, we illustrate parts of the main combinatorial idea with the help of a simplified graph problem that will be described in section 3. We remark that this example is a strong simplification of the general case that we have to solve. Our main result is all the more surprising, because it is known [1] that polynomial-time T-autoreducibility does not imply polynomial-time T-mitoticity. We improve this and disprove the equivalence between autoreducibility and mitoticity for all polynomial-time-bounded reducibilities between 3-tt-reducibility and Turing-reducibility: There exists a sparse set in EXP that is polynomial-time 3-tt-autoreducible but not weakly polynomial-time T-mitotic. In particular, polynomial-time T-autoreducible does not imply polynomial-time weakly T-mitotic. This result settles another open question raised by Buhrman and Torenvliet [6].

Our main result relates local redundancy of information to global redundancy of information in the following sense. If a set A is m-autoreducible, then x and $f(x)$ contain the same information about A. This can be viewed as local redundancy. Whereas if A is m-mitotic, then A can be split into two sets B and C such that A, B, and C are polynomial-time many-one equivalent. Thus the sets B and C have exactly the same information as the original set A. This can be viewed as global redundancy in A. Our main result states that local redundancy is the same as global redundancy.

Our result can also be viewed as a step towards understanding the isomorphism conjecture [3]. This conjecture states that all NP-complete sets are isomorphic to each other. In spite of several years of research, we do not have any concrete evidence either in support or against the isomorphism conjecture.[2] It is easy to see that if the isomorphism conjecture holds for classes such as NP, PSPACE, and EXP, then complete sets for these classes are m-autoreducible as well as m-mitotic. Given our current inability to make progress about the isomorphism conjecture, the next best thing we can hope for is to make progress on statements that the isomorphism conjecture implies. We note that this is not an entirely new approach. For example,

[2] It is currently believed that if one-way functions exist, then the isomorphism conjecture is false. However, we do not have a proof of this.

if the isomorphism conjecture is true, then NP-complete sets cannot be sparse. This motivated researchers to consider the question of whether complete sets for NP can be sparse. This line of research led to beautiful results: Mahaney [17] showed that many-one-hard sets for NP are not sparse unless P = NP. Karp and Lipton [13] proved that if sparse Turing-hard sets for NP exist, then PH collapses to the second level. Ogiwara and Watanabe [18] showed that bounded-truth-table-hard sets for NP cannot be sparse unless P = NP. Our results show that another consequence of isomorphism, namely "NP-complete sets are m-mitotic," holds. Note that this is an unconditional result.

Buhrman et al. [4] and Buhrman and Torenvliet [7, 8] argue that it is critical to study the notions of autoreducibility and mitoticity. They showed that resolving questions regarding autoreducibility of complete sets leads to unconditional separation results. For example, consider the question of whether truth-table complete sets for PSPACE are nonadaptive autoreducible. An affirmative answer separates NP from NL, while a negative answer separates the polynomial-time hierarchy from PSPACE. They argue that this approach does not have the curse of *relativization* and is worth pursuing. We refer the reader to the recent survey by Buhrman and Torenvliet [8] for more details.

1.1. Previous work. The question of whether complete sets for various classes are autoreducible has been studied extensively [20, 2, 4]. Beigel and Feigenbaum [2] showed that Turing complete sets for the classes that form the polynomial hierarchy, Σ_i^{P}, Π_i^{P}, and Δ_i^{P}, are Turing autoreducible. Thus, all Turing complete sets for NP are Turing autoreducible. Buhrman et al. [4] showed that Turing complete sets for EXP and Δ_i^{EXP} are autoreducible, whereas there exists a Turing complete set for EESPACE that is not Turing autoreducible. Regarding NP, Buhrman et al. [4] showed that truth-table complete sets for NP are probabilistic truth-table autoreducible. Recently, Glaßer et al. [11] showed that complete sets for classes such as NP, PSPACE, and Σ_i^{P} are m-autoreducible.

Buhrman, Hoene, and Torenvliet [5] showed that EXP complete sets are weakly many-one mitotic. This result was recently improved independently by Kurtz [8] and Glaßer et al. [12, 11]. Buhrman and Torenvliet [8] observed that Kurtz's proof can be extended to show that 2-tt complete sets for EXP are 2-tt mitotic. This cannot be extended to 3-tt reductions: There exist 3-tt complete sets for EXP that are not btt-autoreducible and hence not btt-mitotic [4]. Glaßer et al. also showed that NEXP complete sets are weakly m-mitotic and that PSPACE-complete sets are weakly Turing-mitotic.

2. Preliminaries. We use standard notation and assume familiarity with standard resource-bounded reductions. We consider words in lexicographic order. All used reductions are polynomial-time computable. PF denotes the class of polynomial-time computable functions.

Definition 2.1 (see [1]). *A set A is* polynomially T-autoreducible *(T-autoreducible, for short) if there exists a polynomial-time-bounded oracle Turing machine M such that $A = L(M^A)$ and for all x, M on input x never queries x. A set A is* polynomially m-autoreducible *(m-autoreducible, for short) if $A \leq_m^p A$ via a reduction function f such that for all x, $f(x) \neq x$.*

Definition 2.2 (see [1]). *A recursive set A is* polynomial-time T-mitotic *(T-mitotic, for short) if there exists a set $B \in \mathrm{P}$ such that $A \equiv_T^p A \cap B \equiv_T^p A \cap \overline{B}$. A is* polynomial-time m-mitotic *(m-mitotic, for short) if there exists a set $B \in \mathrm{P}$ such that $A \equiv_m^p A \cap B \equiv_m^p A \cap \overline{B}$.*

Definition 2.3 (see [1]). *A recursive set A is* polynomial-time weakly T-mitotic *(weakly T-mitotic, for short) if there exist disjoint sets A_0 and A_1 such that $A_0 \cup A_1 = A$ and $A \equiv_T^p A_0 \equiv_T^p A_1$. A is* polynomial-time weakly m-mitotic *(weakly m-mitotic, for short) if there exist disjoint sets A_0 and A_1 such that $A_0 \cup A_1 = A$ and $A \equiv_m^p A_0 \equiv_m^p A_1$.*

3. m-autoreducibility equals m-mitoticity.

It is easy to see that if a nontrivial language L is m-mitotic, then it is m-autoreducible. If L is m-mitotic, then there is a set $S \in \mathrm{P}$ such that $L \cap S \leq_m^p L \cap \overline{S}$ via some f and $L \cap \overline{S} \leq_m^p L \cap S$ via some g. On input x, the m-autoreduction for L works as follows: If $x \in S$ and $f(x) \notin S$, then output $f(x)$. If $x \notin S$ and $g(x) \in S$, then output $g(x)$. Otherwise, output a fixed element from $\overline{L} - \{x\}$.

So m-mitoticity implies m-autoreducibility. The main result of this paper shows that surprisingly the converse holds true as well; i.e., m-mitoticity and m-autoreducibility are equivalent notions.

Theorem 3.1. *Let L be any set such that $|\overline{L}| \geq 2$. L is m-autoreducible if and only if L is m-mitotic.*

Before proceeding to the proof we first discuss parts of the main ideas and the intuition behind the proof. To give an example of the difficulties in the proof, we strongly simplify the general case so that we end at a much easier problem for finite graphs. Later, we have to solve this problem for *infinite* graphs.

Assume that L is m-autoreducible via reduction function f. Given x, the repeated application of f yields a sequence of words $x, f(x), f(f(x)), \ldots,$ which we call the trajectory of x. These trajectories either are infinite or end in a cycle of length at least 2. Note that as f is an autoreduction, $x \neq f(x)$.

At first glance it seems that m-mitoticity can be easily established by the following idea: In every trajectory, label the words at even positions with $+$ and all other words with $-$, i.e., $f(x), f(f(f(x))), \ldots$ obtain $+$ and $x, f(f(x)), \ldots$ obtain $-$. Define S to be the set of strings whose label is $+$. With this "definition" of S it seems that f reduces $L \cap S$ to $L \cap \overline{S}$ and $L \cap \overline{S}$ to $L \cap S$.

However, this labeling strategy has at least two problems. First, it is not clear that $S \in \mathrm{P}$, because, given a string y, we have to compute the parity of the position of y in a trajectory. As trajectories can be of exponential length, this might take exponential time. The second and more fundamental problem is the following: The labeling generated above is *inconsistent* and not well defined. For example, let $f(x) = y$. To label y which trajectory should we use? The trajectory of x or the trajectory of y? If we use trajectory of x, y gets a label of $+$, whereas if we use the trajectory of y, then it gets a label of $-$. Thus S is not well defined, and so this idea does not work. It fails because the labeling strategy is a global strategy. To label a string we have to consider all the trajectories in which x occurs. Every single x gives rise to a labeling of possibly infinitely many words, and these labelings may overlap in an inconsistent way.

We resolve this by using a *local labeling strategy.* More precisely, we compute a label for a given x just by looking at the neighboring values x, $f(x)$, and $f(f(x))$. The strategy is well defined and therefore defines a consistent labeling. We also should guarantee that this local strategy strictly alternates labels; i.e., x gets $+$ if and only if $f(x)$ gets $-$. Such an alternation of labels would help us to establish the m-mitoticity of L.

Thus our goal will be to find a local labeling strategy that has a nice alternation behavior. However, we settle for something less. Instead of requiring that the labels strictly alternate, we require only that, given x, at least one of $f(x), f(f(x)), \dots,$ $f^m(x)$ gets a label that is different from the label of x, where m is polynomially bounded in the length of x. Speaking in terms of graphs, we will solve the following problem.

Infinite graph labeling problem. Let G be an infinite, loop-free, directed graph whose set of nodes is $\mathbb{N}$ such that all nodes have outdegree 1. Moreover, let s be a polynomial-time computable function that on input of a node u computes its successor; i.e., $s(u)$ is the uniquely determined node such that $(u, s(u))$ is an edge. Find a strategy that labels each node of G with either $+$ or $-$ such that

(i) the label of a given node can be computed in polynomial time;

(ii) there is a polynomial p such that for a given node u we can compute in polynomial time a node $v \in \{s(u), s(s(u)), \dots, s^{p(n)}(u)\}$ that has a different label than u. In particular, at least one of the nodes $s(u), s(s(u)), \dots, s^{p(n)}(u)$ has a different label than u.

If we can solve this problem, then this shows that m-autoreducibility implies m-mitoticity. This is seen as follows. Let L be m-autoreducible via autoreduction s, and let G be the graph whose set of nodes is $\mathbb{N}$ and whose set of edges is $\{(u, s(u)) \mid u \in \mathbb{N}\}$. Observe that G and s have the properties mentioned in the infinite graph labeling problem. By solving this problem we obtain a labeling strategy $S = \{u \in \mathbb{N} \mid$ u has label $+\}$ that satisfies (i) and (ii). By (i), $S \in \mathrm{P}$. By (ii), there exists a polynomial-time computable function g that for a given node u computes the node v described in (ii). In particular,

$$u \in S \Leftrightarrow g(u) \notin S. \tag{3.1}$$

Moreover, from $g(u) \in \{s(u), s(s(u)), \dots, s^{p(n)}(u)\}$ and from the fact that s is an autoreduction for L it follows that

$$u \in L \Leftrightarrow g(u) \in L. \tag{3.2}$$

The equivalences (3.1) and (3.2) yield the m-mitoticity of L (formally proved in Proposition 3.2).

To keep this proof sketch simpler, we now restrict our analysis to finite graphs, make several assumptions, and ignore several technical but important details. If we assume (for simplicity) that on strings $x \notin 1^*$ the autoreduction is length preserving such that $f(x) > x$, then we arrive at the following labeling problem for finite graphs.

Simplified problem. Let G_n be a directed graph with 2^n vertices such that every string of length n is a vertex of G_n. Assume that 1^n is a sink, that nodes $u \neq 1^n$ have outdegree 1, and that $u < v$ for edges (u, v). For $u \neq 1^n$ let $s(u)$ denote u's unique successor, i.e., $s(u) = v$ if (u, v) is an edge. Find a strategy that labels each node with either $+$ or $-$ such that

(i) given a node u, its label can be computed in polynomial time in n;

(ii) there exists a polynomial p such that for every node u at least one of the nodes $s(u), s(s(u)), \dots, s^{p(n)}(u)$ has a different label than u.

Cole and Vishkin [9] solve the r-ruling set problem which corresponds to the following restriction of the above simplified problem for finite graphs: First, the graph must be connected and it must have indegree 1. This means that the graph has to be a simple ring. Second, the predecessor of a node must be computable efficiently (i.e., in polynomial time in n). In contrast, the general problem we have to solve here comprises graphs that are infinite, that are not necessarily connected, and that have an unbounded indegree.

We now exhibit a labeling strategy for the simplified problem. To define this labeling, we use the following *distance function*: $d(x,y) \overset{df}{=} \lfloor \log |y-x| \rfloor$. (Our formal proof uses a variant of this function.)

```
0   // Strategy for labeling node x
1   let $y = s(x)$ and $z = s(y)$.
2   if $d(x,y) > d(y,z)$, then output $-$
3   if $d(x,y) < d(y,z)$, then output $+$
4   $r := d(x,y)$
5   if $\lfloor x/2^{r+1} \rfloor$ is even, then output $+$; else output $-$
```

Clearly, this labeling strategy satisfies condition (i). We give a sketch of the proof that also satisfies condition (ii). Define $m = 5n$ and let $u_1, u_2, \ldots, u_m$ be a path in the graph. It suffices to show that not all the nodes $u_1, u_2, \ldots, u_m$ obtain the same label. Assume that this does not hold; say all these nodes get label $+$. So no output is made in line 2, and therefore the distances $d(u_i, u_{i+1})$ do not decrease. Note that the distance function maps to natural numbers. If the distance increases more than n times, then $d(u_{m-1}, u_m) > n$. Therefore, $u_m - u_{m-1} > 2^{n+1}$, which is impossible for words of length n. Hence we have seen that the distances do not decrease and that they increase at most n times. So along the path $u_1, u_2, \ldots, u_m$ there exist at least $m - n = 4n$ positions where the distance stays the same. By a pigeon hole argument, there exist four consecutive such positions, i.e., nodes $v = u_i$, $w = u_{i+1}$, $x = u_{i+2}$, $y = u_{i+3}$, $z = u_{i+4}$ such that $d(v,w) = d(w,x) = d(x,y) = d(y,z)$. So for the inputs v, w, and x, we reach line 4 where the algorithm will assign $r = d(v,w)$. Observe that for all words w_1 and w_2, the value $d(w_1, w_2)$ allows an approximation of $w_2 - w_1$ up to a factor of 2. More precisely, $w - v$, $x - w$, and $y - x$ belong to the interval $[2^r, 2^{r+1})$. It is an easy observation that this implies that not all of the following values can have the same parity: $\lfloor v/2^{r+1} \rfloor$, $\lfloor w/2^{r+1} \rfloor$, and $\lfloor x/2^{r+1} \rfloor$. According to line 5, not all words v, w, and x obtain the same label. This is a contradiction which shows that not all the nodes $u_1, u_2, \ldots, u_m$ obtain the same label. This proves (ii) and solves the simplified problem.

A generalization of this strategy allows us to solve the infinite graph labeling problem, which in turn establishes m-mitoticity for the m-autoreducible set L.

Now we give a formal proof of Theorem 3.1.

The dyadic representation of natural numbers provides a one-one correspondence between words over $\Sigma = \{0,1\}$ and natural numbers. This correspondence translates operations and relations over natural numbers to operations and relations over words. We denote the absolute value of an integer by $\mathrm{abs}(x)$. This avoids a conflict between the notation of the length of a word w and the notation of the absolute value of the integer represented by w. Moreover, $\log(x)$ denotes x's logarithm to base 2. We use the following proposition.

Proposition 3.2. *Let L be any set such that $|\overline{L}| \geq 2$. L is m-mitotic if and only if there exist a total $g \in \mathrm{PF}$ and a set $S \in \mathrm{P}$ such that, for all x,*

1. *$x \in L \Leftrightarrow g(x) \in L$, and*
2. *$x \in S \Leftrightarrow g(x) \notin S$.*

Proof. Choose distinct words $w_1, w_2 \in \overline{L}$. If L is m-mitotic, then there exists $S \in \mathrm{P}$ such that $L \cap S \leq_m^p L \cap \overline{S}$ via some $g_1 \in \mathrm{PF}$ and $L \cap \overline{S} \leq_m^p L \cap S$ via some $g_2 \in \mathrm{PF}$. We may assume that $w_1 \in S$ and $w_2 \in \overline{S}$; otherwise the set $S \cup \{w_1\} - \{w_2\}$ can be used instead of S. A simple proof shows that the following function g satisfies the statements 1 and 2 from the proposition:

$$g(x) \overset{df}{=} \begin{cases} g_1(x) & \text{if } x \in S \text{ and } g_1(x) \in \overline{S}, \\ w_2 & \text{if } x \in S \text{ and } g_1(x) \in S, \\ g_2(x) & \text{if } x \in \overline{S} \text{ and } g_2(x) \in S, \\ w_1 & \text{if } x \in \overline{S} \text{ and } g_2(x) \in \overline{S}. \end{cases}$$

Now assume there exist a total $g \in \mathrm{PF}$ and an $S \in \mathrm{P}$ that satisfy statements 1 and 2. It follows that $L \cap S \leq_m^p L \cap \overline{S}$ and $L \cap \overline{S} \leq_m^p L \cap S$, both via g. The following function reduces L to $L \cap S$:

$$g'(x) \overset{df}{=} \begin{cases} x & \text{if } x \in S, \\ g(x) & \text{if } x \in \overline{S}. \end{cases}$$

The following function reduces $L \cap S$ to L:

$$g''(x) \overset{df}{=} \begin{cases} x & \text{if } x \in S, \\ w_1 & \text{if } x \in \overline{S}. \end{cases}$$

This shows $L \equiv_m^p L \cap S \equiv_m^p L \cap \overline{S}$, and hence L is m-mitotic. □

Proof of Theorem 3.1. If L is m-mitotic, then there exist $S \in \mathrm{P}$ and $f_1, f_2 \in \mathrm{PF}$ such that $L \cap S \leq_m^p L \cap \overline{S}$ via f_1 and $L \cap \overline{S} \leq_m^p L \cap S$ via f_2. By assumption, there exist different words $v, w \in \overline{L}$. The following function is an m-autoreduction for L:

$$f'(x) \overset{df}{=} \begin{cases} f_1(x) & \text{if } x \in S \text{ and } f_1(x) \notin S, \\ f_2(x) & \text{if } x \notin S \text{ and } f_2(x) \in S, \\ \min(\{v, w\} - \{x\}) & \text{otherwise.} \end{cases}$$

For the other direction, let us assume that L is m-autoreducible, and let $f \in \mathrm{PF}$ be an m-autoreduction for L. Choose $k \geq 1$ such that f is computable in time $n^k + k$. Using Proposition 3.2, we show L's m-mitoticity as follows: We construct a total $g \in \mathrm{PF}$ and an $S \in \mathrm{P}$ such that $(x \in L \Leftrightarrow g(x) \in L)$ and $(x \in S \Leftrightarrow g(x) \notin S)$.

Let t be a tower function defined by $t(0) = 0$ and $t(i+1) = t(i)^k + k$ for $i \geq 0$. Define the inverse tower function as $t^{-1}(n) = \min\{i \mid t(i) \geq n\}$. Note that $t^{-1} \in \mathrm{PF}$. We partition the set of all words according to the parity of the inverse tower function of their lengths:

$$S_0 \overset{df}{=} \{x \mid t^{-1}(|x|) \equiv 0 (\mathrm{mod}\ 2)\},$$
$$S_1 \overset{df}{=} \{x \mid t^{-1}(|x|) \equiv 1 (\mathrm{mod}\ 2)\}.$$

Note that $S_0, S_1 \in \mathrm{P}$.

The following *distance* function for natural numbers x and y plays a crucial role in our proof:

$$d(x, y) \overset{df}{=} \mathrm{sgn}(y - x) \cdot \lfloor \log(\mathrm{abs}(y - x)) \rfloor.$$

This function is computable in polynomial time. We define a set S (which will be used as separator for L) by the following algorithm which works on input x.

```
// Algorithm for set S
y := f(x), z := f(f(x))
if |y| > |x|, then (if x ∈ S_0 then accept else reject)
```

```
3   if $|z| > |y|$, then (if $y \in S_1$ then accept else reject)
4   if $x = z$, then (if $x > f(x)$ then accept else reject)
5   // here x, y, and z are pairwise different
6   if $d(x, y) > d(y, z)$, then reject
7   if $d(x, y) < d(y, z)$, then accept
8   $r := d(x, y)$
9   if $\lfloor y/2^{abs(r)+1} \rfloor$ is even, then accept else reject
```

Observe that $S \in$ P. We will show $L \equiv_m^p L \cap S \equiv_m^p L \cap \overline{S}$, which implies that L is m-mitotic.

CLAIM 3.3. *Let y be any word, and let $m = |y|$. If for all $i \in [0, 6m + 3]$, $|f^i(y)| \geq |f^{i+1}(y)|$, then there exists $j \in [0, 6m + 3]$ such that*

$$f^j(y) \in S \Leftrightarrow f^{j+1}(y) \notin S.$$

Proof. Assume that the claim does not hold. Moreover, assume that for all $j \in [0, 6m + 4]$, $f^j(y) \in S$. For the other case (i.e., for all $j \in [0, 6m + 4]$, $f^j(y) \notin S$) one can argue analogously. Consider the algorithm for S.

Fact 1. For $j \in [0, 6m + 2]$, the algorithm on input $f^j(y)$ stops either in line 7 or in line 9.

This fact is proved as follows. Assume there exists $j \in [0, 6m + 2]$ such that the algorithm on input $f^j(y)$ stops in lines 2 or 3. In this case, $|f^j(y)| < |f^{j+1}(y)|$ or $|f^{j+1}(y)| < |f^{j+2}(y)|$, which contradicts our assumption.

Assume there exists $j \in [0, 6m + 2]$ such that the algorithm on input $f^j(y)$ stops in line 4. By assumption of the claim, $|f^j(y)| \geq |f^{j+1}(y)| \geq |f^{j+2}(y)|$. Moreover, $f^j(y) = f^{j+2}(y)$, since we stop in line 4. So $|f^j(y)| = |f^{j+1}(y)|$. Therefore, on both inputs, $f^j(y)$ and $f^{j+1}(y)$, the algorithm stops in line 4. Note that $f^j(y) \neq f^{j+1}(y)$, since f is an m-autoreduction. Hence by line 4, $f^j(y) \in S \Leftrightarrow f^{j+1}(y) \notin S$, which contradicts our assumption.

Assume there exists $j \in [0, 6m + 2]$ such that the algorithm on input $f^j(y)$ stops in line 6. So $f^j(y) \notin S$, which contradicts the assumption. This proves Fact 1:

$$J \overset{df}{=} \{j \in [0, 6m + 2] \mid \text{on input } f^j(y) \text{ the algorithm for } S \text{ stops in line 7}\},$$
$$K \overset{df}{=} \{j \in [0, 6m + 2] \mid \text{on input } f^j(y) \text{ the algorithm for } S \text{ stops in line 9}\}.$$

By Fact 1, $J \cup K = \{0, \ldots, 6m + 2\}$. From the algorithm we see the following:

$$\forall j \in J, \quad d(f^j(y), f^{j+1}(y)) < d(f^{j+1}(y), f^{j+2}(y)), \tag{3.3}$$

$$\forall j \in K, \quad d(f^j(y), f^{j+1}(y)) = d(f^{j+1}(y), f^{j+2}(y)). \tag{3.4}$$

Case 1: $\|J\| > 2m$. Together with (3.3) and (3.4) this shows

$$d(f^{6m+3}(y), f^{6m+4}(y)) - d(f^0(y), f^1(y)) > 2m. \tag{3.5}$$

It follows that

$$d(f^{6m+3}(y), f^{6m+4}(y)) > m \tag{3.6}$$

or

$$d(f^0(y), f^1(y)) < -m. \tag{3.7}$$

Assume that (3.6) holds. By the assumption of the claim, $f^{6m+3}(y)$ and $f^{6m+4}(y)$ are words of length $\leq m$. So the length of $\mathrm{abs}(f^{6m+4}(y) - f^{6m+3}(y))$ is $\leq m$. From the dyadic representation of numbers it follows that $\log(\mathrm{abs}(f^{6m+4}(y) - f^{6m+3}(y))) < m+1$ and therefore $d(f^{6m+3}(y), f^{6m+4}(y)) \leq m$. This is a contradiction, since we assumed that (3.6) holds.

Assume now that (3.7) holds. Again, $f^0(y)$ and $f^1(y)$ are words of length $\leq m$. So $1 \leq \mathrm{abs}(f^0(y) - f^1(y)) \leq 2^{m+1} - 2$. It follows that $0 \leq \log(\mathrm{abs}(f^1(y) - f^0(y))) < m+1$, and therefore $d(f^0(y), f^1(y)) \geq -m$. This is a contradiction, since we assumed that (3.7) holds.

Case 2: $\|J\| \leq 2m$. Note that $[0, 6m+2]$ contains $6m+3$ elements, while J contains at most $2m$ elements. So there exists $j \in [0, 6m]$ such that $j, j+1, j+2 \in K$. A look at the algorithm tells us the following:

(3.8)
$$d(f^j(y), f^{j+1}(y)) = d(f^{j+1}(y), f^{j+2}(y)) = d(f^{j+2}(y), f^{j+3}(y)) = d(f^{j+3}(y), f^{j+4}(y)).$$

Define r as the number shown in (3.8), and let $z_1 \overset{df}{=} f^j(y)$, $z_2 \overset{df}{=} f^{j+1}(y)$, $z_3 \overset{df}{=} f^{j+2}(y)$, and $z_4 \overset{df}{=} f^{j+3}(y)$. Recall that $z_1, z_2, z_3 \in S$ and that, on input of these words, the algorithm stops in line 9. Therefore, the following must hold:

$$a_1 \overset{df}{=} \lfloor z_2 / 2^{\mathrm{abs}(r)+1} \rfloor \text{ is even,} \tag{3.9}$$

$$a_2 \overset{df}{=} \lfloor z_3 / 2^{\mathrm{abs}(r)+1} \rfloor \text{ is even,} \tag{3.10}$$

$$a_3 \overset{df}{=} \lfloor z_4 / 2^{\mathrm{abs}(r)+1} \rfloor \text{ is even.} \tag{3.11}$$

Case 2a: $r = 0$. Here $z_2 \neq z_4$, since otherwise on input z_2 the algorithm stops in line 4, which contradicts Fact 1. Also, $z_2 \neq z_3$ and $z_3 \neq z_4$, since f is an m-autoreduction. From (3.8) and from the definition of the distance function d we obtain either $z_2 = z_3 - 1 = z_4 - 2$ or $z_4 = z_3 - 1 = z_2 - 2$. So $z_4 - z_2$ equals 2 or -2, and hence $a_3 - a_1$ equals 1 or -1. This contradicts the observations (3.9) and (3.11).

Case 2b: $r > 0$. Here we have $z_1 < z_2 < z_3 < z_4$ and therefore $a_1 \leq a_2 \leq a_3$.

Assume $a_1 = a_3$. Since $d(z_2, z_3) = r$, it holds that $\log(\mathrm{abs}(z_3 - z_2)) \geq r$ and hence $z_3 - z_2 \geq 2^r$. The same argument shows $z_4 - z_3 \geq 2^r$. So $z_4 \geq z_2 + 2^{r+1} = z_2 + 2^{\mathrm{abs}(r)+1}$ and hence $a_3 \geq a_1 + 1$. The latter contradicts the assumption $a_1 = a_3$.

So assume $a_1 < a_3$ which implies $a_3 - a_1 \geq 2$, since both values are even. Since a_2 is even as well, we obtain $a_2 - a_1 \geq 2$ or $a_3 - a_2 \geq 2$. If $a_2 - a_1 \geq 2$, then $z_3 - z_2 > 2^{r+1}$ and so $d(z_2, z_3) > r$. If $a_3 - a_2 \geq 2$, then $z_4 - z_3 > 2^{r+1}$ and so $d(z_3, z_4) > r$. Both conclusions contradict (3.8).

Case 2c: $r < 0$. Here we have $z_1 > z_2 > z_3 > z_4$ and therefore, $a_1 \geq a_2 \geq a_3$.

Assume $a_1 = a_3$. Since $d(z_2, z_3) = r$, it holds that $\log(\mathrm{abs}(z_3 - z_2)) \geq \mathrm{abs}(r)$ and hence $z_2 - z_3 \geq 2^{\mathrm{abs}(r)}$. The same argument shows $z_3 - z_4 \geq 2^{\mathrm{abs}(r)}$. So $z_2 \geq z_4 + 2^{\mathrm{abs}(r)+1}$ and hence $a_1 \geq a_3 + 1$. The latter contradicts the assumption $a_1 = a_3$.

So assume $a_1 > a_3$ which implies $a_1 - a_3 \geq 2$, since both values are even. Since a_2 is even as well, we obtain $a_1 - a_2 \geq 2$ or $a_2 - a_3 \geq 2$. If $a_1 - a_2 \geq 2$, then $z_2 - z_3 > 2^{\mathrm{abs}(r)+1}$ and so $d(z_2, z_3) < -\mathrm{abs}(r) = r$. If $a_2 - a_3 \geq 2$, then $z_3 - z_4 > 2^{\mathrm{abs}(r)+1}$ and so $d(z_3, z_4) < -\mathrm{abs}(r) = r$. Both conclusions contradict (3.8). This completes the proof of Claim 3.3. □

CLAIM 3.4. *There exists a total $r \in \mathrm{PF}$ such that $L \leq_m^p L$ via r and, for every x,*

1. $|f(r(x))| \leq |r(x)|$ *or*
2. $x \in S \Leftrightarrow r(x) \notin S$.

Proof. For every x, let

$$r(x) \overset{df}{=} f^i(x),$$

where i is the smallest number such that $|f^{i+1}(x)| \leq |f^i(x)|$ or $(x \in S \Leftrightarrow f^i(x) \notin S)$. We will prove that such an i exists. Consider the following algorithm which works on input x.

```
0  // Algorithm for function r
1  $z := x$
2  while ($|f(z)| > |z|$ and ($x \in S \Leftrightarrow z \in S$))
3      // here $|z| < |x|^k + k$
4      $z := f(z)$
5  end
6  output z
```

Observe that this algorithm computes the function r.

We prove the invariant in line 3, which will guarantee that the loop in the algorithm halts within polynomial steps in $|x|$. Assume that at some point this invariant does not hold. We consider the first time when this happens. In this case, we must have reached line 3 before, since otherwise $|x| \geq |x|^k + k$, which is not possible. Let z' denote the value of variable z when line 3 was reached last time. So $z = f(z')$. Note that the following inequalities hold, since otherwise the algorithm stops earlier:

$$|x| < |f(x)|, \tag{3.12}$$

$$|z'| < |f(z')|, \tag{3.13}$$

$$|z| < |f(z)|, \tag{3.14}$$

$$|x| < |z'|. \tag{3.15}$$

Moreover,

$$|z'| < |x|^k + k, \tag{3.16}$$

since otherwise already z' violates the invariant, which contradicts the fact that with z we chose the earliest violation of the invariant. From (3.15) and (3.16) we obtain

$$t^{-1}(|x|) \leq t^{-1}(|z'|) \leq t^{-1}(|x|^k + k) = t^{-1}(|x|) + 1. \tag{3.17}$$

From (3.12) it follows that, on input x, the algorithm for S stops in line 2. We see the same for z' and z using (3.13) and (3.14). This implies the following:

$$x \in S \Leftrightarrow x \in S_0, \tag{3.18}$$

$$z' \in S \Leftrightarrow z' \in S_0, \tag{3.19}$$

$$z \in S \Leftrightarrow z \in S_0. \tag{3.20}$$

Note that

$$x \in S \Leftrightarrow z' \in S \Leftrightarrow z \in S, \tag{3.21}$$

since otherwise the algorithm for r stops earlier. Together with (3.18), (3.19), and (3.20) this shows

$$x \in S_0 \Leftrightarrow z' \in S_0 \Leftrightarrow z \in S_0 \tag{3.22}$$

and therefore,

$$t^{-1}(|x|) \equiv t^{-1}(|z'|) \equiv t^{-1}(|z|) \pmod 2. \tag{3.23}$$

Now (3.17) implies $t^{-1}(|x|) = t^{-1}(|z'|)$, and we obtain

$$t^{-1}(|z'|) = t^{-1}(|x|) < t^{-1}(|x|^k + k) \leq t^{-1}(|z|). \tag{3.24}$$

From (3.23) and (3.24) it follows that $t^{-1}(|z|) - t^{-1}(|z'|) \geq 2$. Therefore, $|f(z')| > |z'|^k + k$. This contradicts f's computation time and proves the invariant in line 3.

From the invariant we immediately obtain that every single step of the algorithm can be carried out in time polynomial in $|x|$. Each execution of line 4 increases the length of z. By our invariant, the algorithm must terminate within $|x|^k + k$ iterations of the loop. This shows that r is total and polynomial-time computable. Since r is defined by repeated applications of f, and since f is an autoreduction of L, we obtain $L \leq_m^p L$ via r. The statements 1 and 2 of the claim follow immediately from line 2 of the algorithm, and the proof is complete. □

We continue the proof of Theorem 3.1. Choose a function r according to Claim 3.4. Define a function g by the following algorithm, which works on input x. Below we will show that g satisfies the conditions in Proposition 3.2.

```
0   // Algorithm for function g
1   $y := r(x)$, $m := |y|$
2   if $|y| < |f(y)|$, then return y
3   // here $|y| \geq |f(y)|$
4   $z := y$
5   for $i := 0$ to $6m + 3$
6   // here $z = f^i(y)$, $|z| \leq m$, and for all $0 \leq j \leq i$, $|f^j(y)| \geq |f^{j+1}(y)|$
7       if $|f(z)| < |f(f(z))|$, then
8           if $(f(z) \in S \Leftrightarrow x \in S)$ then return z else return f(z)
9       endif
10      $z := f(z)$
11  next i
12  // here for all $0 \leq j \leq 6m + 3$, $|f^j(y)| \geq |f^{j+1}(y)|$
13  $z := y$
14  for $i := 0$ to $6m + 3$
15      // here $z = f^i(y)$ and $|z| \leq m$
16      if $z \in S \Leftrightarrow f(z) \notin S$, then
17          if $(z \in S \Leftrightarrow x \in S)$ then return f(z) else return z
18      endif
19      $z := f(z)$
20  next i
21  // this line is never reached
```

Claim 3.5. *The statements claimed in the comments of the algorithm for g hold true.*

Proof. Clearly, the condition in line 3 holds. Observe that whenever we reach line 6, then $z = f^i(y)$ and $|z| \geq |f(z)|$. Therefore, the condition in line 6 holds. It follows that if we reach line 12, then we must have passed line 6 for $i = 6m + 3$. This shows the condition in line 12. Whenever we reach line 15 it holds that $z = f^i(y)$. From the condition in line 12 it follows that $|z| \leq m$ in line 15.

Finally we argue that we do not reach line 21. Assume that we reach line 12. By the condition in line 12, we satisfy the assumption of Claim 3.3. Therefore, there exists $j \in [0, 6m+3]$ such that $f^j(y) \in S \Leftrightarrow f^{j+1}(y) \notin S$. So for $i = j$ the condition in line 16 is true, and therefore the algorithm stops before reaching line 21. This completes the proof of Claim 3.5. □

CLAIM 3.6. *g is a total function in* PF *and $L \leq_m^p L$ via g.*

Proof. We immediately see that g is total, since line 21 is never reached.

We argue that $g \in$ PF. Recall that f and r are total functions in PF, and recall that $S \in$ P. So steps 1–4 are computable in polynomial time in $|x|$. Note that m is polynomially bounded in $|x|$. By the remark in line 6, the loop 5–11 needs only polynomial time in $|x|$. The remark in line 15 implies the same for the loop 14–20. This shows $g \in$ PF.

We show $L \leq_m^p L$ via g. Observe that in any case the algorithm returns $f^j(y)$ for a suitable $j \geq 0$. By Claim 3.4, $x \in L \Leftrightarrow y = r(x) \in L$. Since f is an autoreduction of L, we obtain $x \in L \Leftrightarrow g(x) = f^j(y) \in L$, and the proof is complete. □

CLAIM 3.7. *For every x, $x \in S \Leftrightarrow g(x) \notin S$.*

Proof. Consider the computation of the algorithm for g on input x.

Case 1: The output is made in line 2. So we have $|f(r(x))| > |r(x)|$. From Claim 3.4 it follows that $x \in S \Leftrightarrow g(x) = r(x) \notin S$.

Case 2: The output is made in line 8. By lines 6 and 7,

$$|f^i(y)| \geq |f^{i+1}(y)| \quad \text{and} \quad |f^{i+1}(y)| < |f^{i+2}(y)|.$$

(Here i refers to the value of the variable i in the algorithm for g at the time when the algorithm stops in line 8.) Therefore, if we look at the algorithm for S (earlier in this section), then we see that on input $f^i(y)$ the algorithm stops in step 3, while on input $f^{i+1}(y)$ the algorithm stops in step 2. It follows that

$$f^i(y) \in S \Leftrightarrow f^{i+1}(y) \in S_1 \quad \text{and}$$
$$f^{i+1}(y) \in S \Leftrightarrow f^{i+1}(y) \in S_0.$$

So $z = f^i(y) \in S \Leftrightarrow f(z) \notin S$, and therefore, by line 8 of the algorithm for g,

$$x \in S \Leftrightarrow g(x) \notin S.$$

Case 3: The output is made in line 17. From line 16 it follows that $x \in S \Leftrightarrow g(x) \notin S$. This completes the proof of Claim 3.7. □

Claims 3.6 and 3.7 allow the application of Proposition 3.2. Hence L is m-mitotic. This finishes the proof of Theorem 3.1. □

Call a set L *nontrivial* if $\|L\| \geq 2$ and $\|\overline{L}\| \geq 2$.

COROLLARY 3.8. *Every nontrivial set that is many-one complete for one of the following complexity classes is m-mitotic:*

- NP*,* coNP*,* ⊕P*,* PSPACE*,* EXP*,* NEXP*,*
- *any level of* PH*,* MODPH*, or the Boolean hierarchy over* NP*.*

Proof. Glaßer et al. [11] showed that all many-one complete sets of the above classes are m-autoreducible. By Theorem 3.1, these sets are m-mitotic. □

COROLLARY 3.9. *A nontrivial set L is* NP*-complete if and only if L is the union of two disjoint* P*-separable* NP*-complete sets.*

So unions of disjoint P-separable NP-complete sets form exactly the class of NP-complete sets. What class is obtained when we drop P-separability? Does this class contain a set that is not NP-complete? In other words, is the union of disjoint NP-complete sets always NP-complete? We leave this as an open question.

Ambos-Spies [1] defined a set A to be *ω-m-mitotic* if for every $n \geq 2$ there exists a partition $(Q_1, \ldots, Q_n)$ of Σ^* such that each Q_i is polynomial-time decidable and the following sets are polynomial-time many-one equivalent: $A, A \cap Q_1, \ldots, A \cap Q_n$.

Corollary 3.10. *Every nontrivial infinite set that is many-one complete for a class mentioned in Corollary* 3.8 *is ω-m-mitotic.*

Proof. Fix a class mentioned in Corollary 3.8, and let A be a nontrivial, infinite, many-one complete set. We need to prove that for every $n \geq 2$ there exists a partition $(Q_1, \ldots, Q_n)$ of Σ^* such that each Q_i belongs to P and the sets $A, A \cap Q_1, \ldots, A \cap Q_n$ are all polynomial-time many-one equivalent. We prove this by induction on n. The base $n = 2$ is an immediate consequence of Corollary 3.8. Now assume $n \geq 3$. By the induction hypothesis, there exists a partition $(Q_1, \ldots, Q_{n-1})$ of Σ^* such that each Q_i belongs to P and the sets $A, A \cap Q_1, \ldots, A \cap Q_{n-1}$ are all polynomial-time many-one equivalent. Since A is infinite and $(Q_1, \ldots, Q_{n-1})$ is a partition of Σ^*, there exists some j such that $A \cap Q_j$ is infinite as well. Moreover, $A \cap Q_j$ is nontrivial and polynomial-time many-one complete. By Corollary 3.8, $A \cap Q_j$ is m-mitotic. So there exists $S \in \mathrm{P}$ such that $A \cap Q_j$, $A \cap Q_j \cap S$, and $A \cap Q_j \cap \overline{S}$ are polynomial-time many-one equivalent. Define $Q'_j = Q_j \cap S$ and $Q''_j = Q_j \cap \overline{S}$. The following polynomial-time decidable partition of Σ^* completes the proof: $(Q_1, \ldots, Q_{j-1}, Q'_j, Q''_j, Q_{j+1}, \ldots, Q_{n-1})$. □

Next, we note that the proof the main theorem also yields the following result.

Theorem 3.11. *Every* 1*-tt-autoreducible set is* 1*-tt-mitotic.*

The proof of Theorem 3.1 provides a strategy that solves the infinite graph labeling problem defined at the beginning of this section. In particular, for every graph that satisfies certain prerequisites there exists a polynomial-time algorithm that labels given nodes with either + or − such that each node has a polynomial-bounded path that leads to a node with a different label. This is made precise as follows.

Corollary 3.12. *Let $G = (\mathbb{N}, E)$ be an infinite, loop-free, directed graph with outdegree* 1 *such that there exists an $f \in \mathrm{PF}$ that computes the successor of a given node u, i.e., $(u, f(u)) \in E$. Then there exist a polynomial p, $S \in \mathrm{P}$, and $g \in \mathrm{PF}$ such that for all x,*

$$(\exists i \in [1, p(|x|)],\, g(x) = f^i(x)) \quad \textit{and} \quad (x \in S \Leftrightarrow g(x) \notin S).$$

Proof. Note that f is an m-autoreduction for $L \overset{df}{=} \Sigma^*$. Choose $k \geq 1$ such that f is computable in time $n^k + k$. Now consider the implication from left to right in the proof of Theorem 3.1. There the assumption $|\overline{L}| \geq 2$ is needed only at the end of the proof when Proposition 3.2 is applied. So for $L = \Sigma^*$ the proof goes through until Proposition 3.2 is applied. In particular, we obtain a total $g \in \mathrm{PF}$ and an $S \in \mathrm{P}$ such that, for all x,

$$x \in S \Leftrightarrow g(x) \notin S. \tag{3.25}$$

Consider the function r which is defined in the proof of Claim 3.4. From the claim it follows that r is a total, polynomial-time computable function such that $r(x) = f^i(x)$ for some $i \geq 0$. Moreover, in the proof of the claim we show that the algorithm for r on input x terminates within $|x|^k + k$ iterations of the loop. From the algorithm it follows that, for all x,

$$\exists i \in [0, |x|^k + k] \quad \text{such that} \quad r(x) = f^i(x). \tag{3.26}$$

Let q be a polynomial bounding the computation time for r. Now consider the algorithm for g in the proof of Theorem 3.1. By Claim 3.5, the statements in the comments of this algorithm hold true. In particular, at any time it holds that $z = f^i(y)$ for some $i \in [0, 6m+4]$, where $y = r(x)$ and $m = |y| \leq q(|x|)$. If the algorithm for g stops, then it returns either z or $f(z)$. So $g(x) = f^i(y)$ for some $i \in [0, 6m+5] \subseteq [0, 6q(|x|)+5]$. Together with (3.26) this shows

$$\exists i \in [0, p(|x|)] \quad \text{such that} \quad g(x) = f^i(x),$$

where $p(n) \overset{df}{=} 6q(n) + 5 + n^k + k$. By (3.25), $g(x) \neq x$, which implies

$$\exists i \in [1, p(|x|)] \quad \text{such that} \quad g(x) = f^i(x). \tag{3.27}$$

The corollary follows from (3.25) and (3.27). □

4. 3-tt-autoreducibility does not imply weak T-mitoticity.

In this section we prove a theorem that shows in a strong way that T-autoreducible does not imply weakly T-mitotic. Hence, our main theorem cannot be generalized.

Lemma 4.1. *Let $l, m \geq 0$, and let $k \geq (l+2)^{2^m}$. If $Q_1, \ldots, Q_k$ are sets of cardinality $\leq l$ and if $n_1, \ldots, n_k$ are pairwise different numbers, then there exist pairwise different indices $i_1, \ldots, i_m$ such that for all $s, t \in [1, m]$,*

$$s \neq t \;\Rightarrow\; n_{i_s} \notin Q_{i_t}.$$

Proof. The proof is by induction on $n = l + m$ such that the induction base covers all cases where $l = 0$ or $m = 0$. For these cases the lemma holds trivially. In particular, this covers the case $n = 1$.

Assume there exists $n \geq 1$ such that the lemma holds for all l and m such that $l = 0$ or $m = 0$ or $l + m \leq n$. Now we prove it for l and m such that $l \geq 1$, $m \geq 1$, and $l + m = n + 1$.

Case 1: There exist at least $k - \sqrt{k} - l - 1$ indices $j > 1$ such that $n_1 \in Q_j$. Let $k' = \lceil k - \sqrt{k} - l - 1 \rceil$ and choose pairwise different indices $j_1, \ldots, j_{k'}$ such that for all i, $j_i \neq 1$ and $n_1 \in Q_{j_i}$. Let $l' = l - 1$ and let $R_i = Q_{j_i} - \{n_1\}$ and $r_i = n_{j_i}$ for $1 \leq i \leq k'$. Observe $l' \geq 0$ and $m \geq 1$. We estimate k' as follows:

(4.1)

$$\begin{aligned} k' &\geq k - \sqrt{k} - l - 1 \\ &\geq (l+2)^{2^m} - \sqrt{(l+2)^{2^m}} - l - 1 && \text{(since } (a \geq b \Rightarrow a - \sqrt{a} \geq b - \sqrt{b}) \text{ for } a, b \geq 1) \\ &\geq (l'+2)^{2^m} && \text{(follows from (4.2) in the estimation below).} \end{aligned}$$

For (4.1) the following estimation is needed:

(4.2)

$$\begin{aligned} l + 1 &\geq l + 1, \\ (l+2)^{2^{m-1}-1} \cdot (l + 2 - 1) &\geq (l+1)^{2^{m-1}-1} \cdot (l+1), \\ (l+2)^{2^{m-1}} - 1 &\geq (l+1)^{2^{m-1}} && \text{(since } (l+2)^{2^{m-1}-1} \geq 1), \\ (l+2)^{2^{m-1}} \cdot \left[(l+2)^{2^{m-1}} - 1\right] &\geq (l+2)^{2^{m-1}} \cdot \left[(l+1)^{2^{m-1}}\right], \\ (l+2)^{2^m} - (l+2)^{2^{m-1}} &\geq (l+1+1) \cdot (l+1)^{2^{m-1}-1} \cdot \left[(l+1)^{2^{m-1}}\right], \\ (l+2)^{2^m} - \sqrt{(l+2)^{2^m}} &\geq (l+1)^{2^m} + (l+1)^{2^m-1}, \\ (l+2)^{2^m} - \sqrt{(l+2)^{2^m}} - l - 1 &\geq (l'+2)^{2^m} && \text{(since } (l+1)^{2^m-1} \geq l+1). \end{aligned}$$

Note that $l' + m = n$. Also, $R_1, \dots, R_{k'}$ are sets of cardinality $\le l'$ and $r_1, \dots, r_{k'}$ are pairwise different numbers. By the induction hypothesis there exist pairwise different indices $i_1, \dots, i_m$ such that for all $s, t \in [1, m]$, $(s \ne t \Rightarrow r_{i_s} \notin R_{i_t})$. For all $s \in [1, m]$, $r_{i_s} \ne n_1$. Therefore, for all $s, t \in [1, m]$,

$$(s \ne t \Rightarrow r_{i_s} \notin R_{i_t} \cup \{n_1\})$$

and hence

$$(s \ne t \Rightarrow n_{j_{i_s}} \notin Q_{j_{i_t}}).$$

Thus the lemma is satisfied by the indices $j_{i_1}, j_{i_2}, \dots, j_{i_m}$.

Case 2: There exist less than $k - \sqrt{k} - l - 1$ indices $j > 1$ such that $n_1 \in Q_j$. So there exist more than $\sqrt{k} + l$ indices $j > 1$ such that $n_1 \notin Q_j$. Since $\|Q_1\| \le l$, there exist more than $\sqrt{k}$ indices $j > 1$ such that $n_1 \notin Q_j$ and $n_j \notin Q_1$. Hence there exist at least $k' \overset{df}{=} \lceil \sqrt{k} \rceil$ such indices. So we can choose pairwise different indices $j_1, \dots, j_{k'}$ such that, for all i,

$$j_i \ne 1 \wedge n_1 \notin Q_{j_i} \wedge n_{j_i} \notin Q_1. \tag{4.3}$$

Let $m' = m - 1$ and let $R_i = Q_{j_i}$ and $r_i = n_{j_i}$ for $1 \le i \le k'$. Note that $l \ge 1$ and $m' \ge 0$. Observe that

$$k' \ge \sqrt{k} \ge \sqrt{(l+2)^{2^m}} = (l+2)^{2^{m'}}$$

and $l + m' = n$. Also, $R_1, \dots, R_{k'}$ are sets of cardinality $\le l$, and $r_1, \dots, r_{k'}$ are pairwise different numbers. So by the induction hypothesis there exist pairwise different indices $i_1, \dots, i_{m'}$ such that for all $s, t \in [1, m']$,

$$s \ne t \Rightarrow r_{i_s} \notin R_{i_t}$$

and hence

$$s \ne t \Rightarrow n_{j_{i_s}} \notin Q_{j_{i_t}}.$$

From (4.3) it follows that the lemma is satisfied by the indices $1, j_{i_1}, j_{i_2}, \dots, j_{i_{m'}}$. □

THEOREM 4.2. *There exists $L \in \text{SPARSE} \cap \text{EXP}$ such that*

- *L is 3-tt-autoreducible, but*
- *L is not weakly T-mitotic.*

Proof. Define a tower function by $t(0) = 4$ and

$$t(n+1) = 2^{2^{2^{2^{t(n)}}}}.$$

For any word s, let $W(s) = \{s00, s01, s10, s11\}$. We will define L such that it satisfies the following:

(i) If $w \in L$, then there exists n such that $|w| = t(n)$.

(ii) For all n and all $s \in \Sigma^{t(n)-2}$, the set $W(s) \cap L$ either is empty or contains exactly two elements.

It is easy to see that such an L is 3-tt-autoreducible: On input w, determine n such that $|w| = t(n)$. If such n does not exist, then reject. Otherwise, let s be w's prefix of length $|w| - 2$. Accept if and only if the set $L \cap (W(s) - \{w\})$ contains an odd number of elements. This is a 3-tt-autoreduction.

We turn to the construction of L. Let $M_1, M_2, \ldots$ be an enumeration of deterministic, polynomial-time-bounded Turing machines such that the running time of M_i is $n^i + i$. Let $\langle \cdot, \cdot \rangle$ be a pairing function such that $\langle x, y \rangle > x + y$. We construct L stagewise such that in stage n we determine which of the words of length $t(n)$ belong to L. In other words, at stage n we define a set $W_n \subseteq \Sigma^{t(n)}$, and finally we define L to be the union of all W_n.

We start by defining $W_0 = \emptyset$. Suppose we are at stage $n > 0$. Let $m = t(n)$, and determine i and j such that $n = \langle i, j \rangle$. If such i and j do not exist, then let $W_n = \emptyset$ and go to stage $n + 1$. Otherwise, i and j exist. In particular, $i + j < \log \log m$. Let $O \overset{df}{=} W_0 \cup \cdots \cup W_{n-1}$ be the part of L that has been constructed so far. Let $O_1, O_2, \ldots, O_l$ be the list of all subsets of O (lexicographically ordered according to their characteristic sequences). Since $O \subseteq \Sigma^{\leq t(n-1)}$ we obtain $\|O\| \leq 2^{t(n-1)+1}$. Therefore,

$$l \leq 2^{2^{t(n-1)+1}} \leq 2^{2^{2^{t(n-1)}}} = \log \log t(n) = \log \log m. \tag{4.4}$$

We give some intuition for the claim below. If L is weakly T-mitotic, then in particular, there exists a partition $L = L_1 \cup L_2$ such that $L_2 \leq_T^p L_1$ via some machine M_i. Hence $O \cap L_1$ must appear (say, as O_k) in our list of subsets of O. The following claim makes sure that we can find a list of words $s_1, \ldots, s_l$ of length $m - 2$ such that for all $k \in [1, l]$ it holds that if the partition of L is such that $O \cap L_1 = O_k$, then M_i on input of a string from $\{s_k00, s_k01, s_k10, s_k11\}$ does not query the oracle for words from $W(s_r)$ if $r \neq k$. Hence, if M_i queries a word of length m that does not belong to $\{s_k00, s_k01, s_k10, s_k11\}$, then it always gets a no answer. So the following is the only information about the partition of L that can be exploited by M_i:

- the partition of $O = \Sigma^{<t(n)} \cap L$,
- the partition of $W(s_k) \cap L$.

In particular, M_i cannot exploit information about the partition of $W(s_r) \cap L$ for $r \neq k$. This independence of M_i makes our diagonalization possible.

CLAIM 4.3. *There exist pairwise different words $s_1, \ldots, s_l \in \Sigma^{m-2}$ such that for all $k, r \in [1, l]$, $k \neq r$, and all $y \in W(s_k)$, neither $M_i^{O-O_k}(y)$ nor $M_j^{O_k}(y)$ query the oracle for words in $W(s_r)$.*

Proof. For $s \in \Sigma^{m-2}$, let

$$Q_s \overset{df}{=} \{s' \in \Sigma^{m-2} \mid \exists k \in [1, l], \exists y \in W(s), \exists q \in W(s') \text{ such that } q \text{ is queried by } M_i^{O-O_k}(y) \text{ or } M_j^{O_k}(y)\}.$$

Observe that, for every $s \in \Sigma^{m-2}$,

$$\begin{aligned} \|Q_s\| &\leq 4l[(m^i + i) + (m^j + j)] \\ &\leq 4(\log \log m)[m^{\log \log m} + \log \log m] \\ &\leq 8(\log \log m) m^{\log \log m} \\ &\leq m^{2 \log \log m} \\ &\leq 2^{\log^2 m} - 2. \end{aligned} \tag{4.5}$$

We identify numbers in $[1, 2^{m-2}]$ with strings in Σ^{m-2}. Considered in this way, each Q_s is a subset of $[1, 2^{m-2}]$. By (4.5), $Q_1, Q_2, \ldots, Q_{2^{m-2}}$ are sets of cardinality $\leq 2^{\log^2 m} - 2$. Clearly, $1, 2, \ldots, 2^{m-2}$ are pairwise different numbers. By (4.4),

$$2^{m-2} \geq (2^{\log^2 m})^{\log m} \geq (2^{\log^2 m})^{2^l}.$$

Therefore, we can apply Lemma 4.1. We obtain indices $s_1, \dots, s_l$ such that for all $k, r \in [1, l]$,

$$r \neq k \;\Rightarrow\; s_r \notin Q_{s_k}. \tag{4.6}$$

Assume there exist $k, r \in [1, l]$, $k \neq r$, and $y \in W(s_k)$ such that some $q \in W(s_r)$ is queried by $M_i^{O-O_k}(y)$ or $M_j^{O_k}(y)$. Hence $s_r \in Q_{s_k}$. This contradicts (4.6) and finishes the proof of Claim 4.3. □

Let $s_1, \dots, s_l \in \Sigma^{m-2}$ be the words assured by Claim 4.3. We define W_n such that for every $k \in [1, l]$ we define a set $V_k \subseteq W(s_k)$, and finally we define W_n to be the union of all V_k. The cardinality of each V_k is either 0 or 2.

Fix some $k \in [1, l]$ and let $Q_k \overset{df}{=} O - O_k$.

Case 1: $M_i^{Q_k}(s_k00)$ accepts or $M_j^{O_k}(s_k00)$ accepts. Define $V_k \overset{df}{=} \emptyset$.

Case 2: $M_i^{Q_k}(s_k00)$ and $M_j^{O_k}(s_k00)$ reject.

Case 2a: For all $y \in \{s_k01, s_k10, s_k11\}$, $M_i^{Q_k \cup \{s_k00\}}(y)$ rejects. Define V_k as a subset of $W(s_k)$ such that $|V_k| = 2$, $s_k00 \in V_k$, and

$$s_k01 \in V_k \Leftrightarrow M_j^{O_k \cup \{s_k00\}}(s_k01) \text{ rejects.}$$

Case 2b: For all $y \in \{s_k01, s_k10, s_k11\}$, $M_j^{O_k \cup \{s_k00\}}(y)$ rejects. Define V_k as a subset of $W(s_k)$ such that $|V_k| = 2$, $s_k00 \in V_k$, and

$$s_k01 \in V_k \Leftrightarrow M_i^{Q_k \cup \{s_k00\}}(s_k01) \text{ rejects.}$$

Case 2c: There exist $y \in \{s_k01, s_k10, s_k11\}$ and $z \in \{s_k01, s_k10, s_k11\}$ such that both $M_i^{Q_k \cup \{s_k00\}}(y)$ and $M_j^{O_k \cup \{s_k00\}}(z)$ accept. Choose $v \in W(s_k) - \{s_k00, y, z\}$ and define $V_k \overset{df}{=} \{s_k00, v\}$.

This finishes the construction of V_k. We define $W_n \overset{df}{=} \bigcup_{k \in [1,l]} V_k$. Finally, L is defined as the union of all W_n.

Note that, by the construction, $W_n \subseteq \Sigma^{t(n)}$, which shows (i). Observe that the construction also ensures (ii). We argue for $L \in$ EXP: Since $l \leq \log\log m$, there are not more than $2^{m \log\log m}$ possibilities for choosing the strings $s_1, \dots, s_l$. For each such possibility we have to simulate $O(l^2)$ computations $M_i(y)$ and $M_j(y)$. This can be done in exponential time in m. For the definition of each V_k we have to simulate a constant number of computations $M_i(y)$ and $M_j(y)$. This shows that L is printable in exponential time. Hence $L \in$ EXP. From the construction it follows that $L \cap \Sigma^m \leq 2l \leq 2\log\log m$. In particular, $L \in$ SPARSE. It remains to show that L is not weakly T-mitotic.

Assume that L is weakly T-mitotic. So L can be partitioned into $L = L_1 \cup L_2$ (a disjoint union) such that

(iii) $L_1 \leq_T^p L_2$ via machine M_i and

(iv) $L_2 \leq_T^p L_1$ via machine M_j.

Let $n = \langle i, j \rangle$, $m = t(n)$, and $O = W_0 \cup \dots \cup W_{n-1}$, i.e., $O = L \cap \Sigma^{<t(n)}$. Let $O_1, O_2, \dots, O_l$ be the list of all subsets of O (again lexicographically ordered according to their characteristic sequences). Let $s_1, \dots, s_l$ and $V_1, \dots, V_l$ be as in the definition of W_n. Choose $k \in [1, l]$ such that $L_1 \cap \Sigma^{<t(n)} = O_k$. Let $Q_k = O - O_k$. So $L_2 \cap \Sigma^{<t(n)} = Q_k$. Clearly, V_k must be defined according to one of the cases above.

Assume that V_k was defined according to Case 1: So $V_k = \emptyset$ and, in particular, $s_k00 \notin L_1$. Without loss of generality assume that $M_i^{Q_k}(s_k00)$ accepts. $M_i^{L_2}(s_k00)$

has running time $m^i + i < m^m + m < t(n+1)$. Hence $M_i^{L_2}(s_k00)$ behaves like $M_i^{L_2 \cap \Sigma^{\leq t(n)}}(s_k00)$. Since s_k was chosen according to Claim 4.3, for all $r \in [1,l] - \{k\}$, $M_i^{Q_k}(s_k00)$ does not query the oracle for words in $W(s_r)$. Note that $W(s_k) \cap L = V_k = \emptyset$. Therefore, $M_i^{L_2}(s_k00)$ behaves like $M_i^{L_2 \cap \Sigma^{< t(n)}}(s_k00)$, which is the same as $M_i^{Q_k}(s_k00)$. The latter accepts, but $s_k00 \notin L_1$. This contradicts (iii).

Assume that V_k was defined according to Case 2: So $V_k = \{s_k00, u\}$, where $u \in \{s_k01, s_k10, s_k11\}$. Assume $V_k \subseteq L_1$. Then as above, $M_i(s_k00)$ with oracle L_2 behaves the same way as $M_i(s_k00)$ with oracle Q_k. The latter rejects, because we are in Case 2. So $s_k00 \notin L_1$, which contradicts our assumption. Analogously the assumption $V_k \subseteq L_2$ implies a contradiction. Therefore,

$$\text{either } (s_k00 \in L_1 \wedge u \in L_2) \text{ or } (u \in L_1 \wedge s_k00 \in L_2). \tag{4.7}$$

Assume that V_k was defined according to Case 2a: So for all $y \in \{s_k01, s_k10, s_k11\}$, $M_i^{Q_k \cup \{s_k00\}}(y)$ rejects. In particular, $M_i^{Q_k \cup \{s_k00\}}(u)$ rejects. Assume $u \in L_1$ and $s_k00 \in L_2$. So $M_i^{L_2}(u)$ rejects, since it behaves the same way as $M_i^{Q_k \cup \{s_k00\}}(u)$. By (iii) this contradicts $u \in L_1$. Therefore, by (4.7) we must have $s_k00 \in L_1$ and $u \in L_2$. In Case 2a, V_k is defined such that

$$s_k01 \in V_k \Leftrightarrow M_j^{O_k \cup \{s_k00\}}(s_k01) \text{ rejects.}$$

Note that $M_j^{O_k \cup \{s_k00\}}(s_k01)$ and $M_j^{L_1}(s_k01)$ behave the same way. Hence,

$$s_k01 \in V_k \Leftrightarrow M_j^{L_1}(s_k01) \text{ rejects.}$$

If $s_k01 \in V_k$, then $u = s_k01$, and hence $M_j^{L_1}(u)$ rejects. This contradicts (iv). Otherwise, if $s_k01 \notin V_k$, then $M_j^{L_1}(s_k01)$ accepts, and hence $u = s_k01 \notin V_k$. This contradicts the assumption $u \in V_k$.

Assume that V_k was defined according to Case 2b: Here we obtain contradictions analogously to Case 2a.

Assume that V_k was defined according to Case 2c: Choose y and z such that both $M_i^{Q_k \cup \{s_k00\}}(y)$ and $M_j^{O_k \cup \{s_k00\}}(z)$ accept. So $u \in \{s_k01, s_k10, s_k11\} - \{y, z\}$. Assume $s_k00 \in L_2$. Hence $M_i^{L_2}(y)$ and $M_i^{Q_k \cup \{s_k00\}}(y)$ behave the same way, showing that $M_i^{L_2}(y)$ accepts. So $y \in L_1$, which contradicts the definition of V_k. Assume $s_k00 \in L_1$. Hence $M_j^{L_1}(z)$ and $M_j^{O_k \cup \{s_k00\}}(z)$ behave the same way, showing that $M_j^{L_1}(z)$ accepts. So $z \in L_2$, which contradicts the definition of V_k.

This finishes Case 2. From the fact that all possible cases led to contradictions, we obtain that the initial assumption was false. Hence, L is not weakly T-mitotic. ∎

Thus there exist 3-tt-autoreducible sets that are not even weakly T-mitotic.[3] By Theorem 3.11, every 1-tt-autoreducible set is 1-tt-mitotic.

5. Remarks. One might wonder whether Theorem 3.1 still holds if one replaces polynomial-time many-one reductions by logarithmic-space many-one reductions. However, in this logspace setting, the proof of Theorem 3.1 does not go through, since the logspace analogue of Claim 3.6 fails. More precisely, we cannot argue that the function g is computable in logspace. Roughly speaking, g is defined as a polynomial superposition of f. This means that in order to compute g, we have to iterate

[3] In a forthcoming paper we improve this result to 2-tt-autoreducible sets.

f a polynomial number of times. If f is polynomial-time computable and not length-increasing (recall that the length-increasing case is already treated by the function r), then g is computable in polynomial time. In contrast, for a logspace computable f, we cannot iterate f for more than a constant number of times. So in this case, g is not logspace computable. The logspace analogue of Theorem 3.1 is indeed false in a relativized world [10].

REFERENCES

[1] K. Ambos-Spies, *P-mitotic sets*, in Logic and Machines, Lecture Notes in Comput. Sci. 177, E. Börger, G. Hasenjäger, and D. Roding, eds., Springer-Verlag, New York, 1984, pp. 1–23.

[2] R. Beigel and J. Feigenbaum, *On being incoherent without being very hard*, Comput. Complexity, 2 (1992), pp. 1–17.

[3] L. Berman and J. Hartmanis, *On isomorphism and density of NP and other complete sets*, SIAM J. Comput., 6 (1977), pp. 305–322.

[4] H. Buhrman, L. Fortnow, D. van Melkebeek, and L. Torenvliet, *Separating complexity classes using autoreducibility*, SIAM J. Comput., 29 (2000), pp. 1497–1520.

[5] H. Buhrman, A. Hoene, and L. Torenvliet, *Splittings, robustness, and structure of complete sets*, SIAM J. Comput., 27 (1998), pp. 637–653.

[6] H. Buhrman and L. Torenvliet, *On the structure of complete sets*, in Proceedings of the 9th Annual Structure in Complexity Theory Conference, Amsterdam, The Netherlands, 1994, IEEE Computer Society Press, Piscataway, NJ, pp. 118–133.

[7] H. Buhrman and L. Torenvliet, *Separating complexity classes using structural properties*, in Proceedings of the 19th IEEE Conference on Computational Complexity, Amherst, MA, 2004, IEEE Computer Society Press, Piscataway, NJ, pp. 130–138.

[8] H. Buhrman and L. Torenvliet, *A Post's program for complexity theory*, Bull. Eur. Assoc. Theor. Comput. Sci. EATCS, 85 (2005), pp. 41–51.

[9] R. Cole and U. Vishkin, *Deterministic coin tossing with applications to optimal parallel list ranking*, Inform. and Control, 70 (1986), pp. 32–53.

[10] C. Glaßer, *Logspace Mitoticity*, Technical Report 400, Institut für Informatik, Universität Würzburg, Würzburg, Germany, 2007.

[11] C. Glaßer, M. Ogihara, A. Pavan, A. L. Selman, and L. Zhang, *Autoreducibility, mitoticity, and immunity*, in Proceedings of the 30th International Symposium on Mathematical Foundations of Computer Science, Lecture Notes in Comput. Sci. 3618, Springer-Verlag, New York, 2005, pp. 387–398.

[12] C. Glaßer, M. Ogihara, A. Pavan, A. L. Selman, and L. Zhang, *Autoreducibility, mitoticity, and immunity*, J. Comput. System Sci., 73 (2007), pp. 735–754.

[13] R. Karp and R. Lipton, *Turing machines that take advice*, Enseign. Math., 28 (1982), pp. 191–209.

[14] A. H. Lachlan, *The priority method* I, Z. Math. Logik Grundlag. Math., 13 (1967), pp. 1–10.

[15] R. E. Ladner, *A completely mitotic nonrecursive r.e. degree*, Trans. Amer. Math. Soc., 184 (1973), pp. 479–507.

[16] R. E. Ladner, *Mitotic recursively enumerable sets*, J. Symbolic Logic, 38 (1973), pp. 199–211.

[17] S. Mahaney, *Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis*, J. Comput. System Sci., 25 (1982), pp. 130–143.

[18] M. Ogiwara and O. Watanabe, *On polynomial-time bounded truth-table reducibility of NP sets to sparse sets*, SIAM J. Comput., 20 (1991), pp. 471–483.

[19] B. Trakhtenbrot, *On autoreducibility*, Dokl. Akad. Nauk SSSR, 192 (1970), pp. 1224–1227; translation in Soviet Math. Dokl., 11 (1970), pp. 814–817.

[20] A. Yao, *Coherent functions and program checkers*, in Proceedings of the 22nd Annual Symposium on Theory of Computing, 1990, pp. 89–94.

SIAM J. COMPUT.
Vol. 37, No. 5, pp. 1536–1564

LEARNING MIXTURES OF PRODUCT DISTRIBUTIONS OVER DISCRETE DOMAINS[*]

JON FELDMAN[†], RYAN O'DONNELL[‡], AND ROCCO A. SERVEDIO[§]

Abstract. We consider the problem of learning *mixtures of product distributions over discrete domains* in the distribution learning framework introduced by Kearns et al. [*Proceedings of the 26th Annual Symposium on Theory of Computing (STOC)*, Montréal, QC, 1994, ACM, New York, pp. 273–282]. We give a poly(n/ϵ)-time algorithm for learning a mixture of k arbitrary product distributions over the n-dimensional Boolean cube $\{0,1\}^n$ to accuracy ϵ, for any constant k. Previous polynomial-time algorithms could achieve this only for $k = 2$ product distributions; our result answers an open question stated independently in [M. Cryan, *Learning and Approximation Algorithms for Problems Motivated by Evolutionary Trees*, Ph.D. thesis, University of Warwick, Warwick, UK, 1999] and [Y. Freund and Y. Mansour, *Proceedings of the 12th Annual Conference on Computational Learning Theory*, 1999, pp. 183–192]. We further give evidence that no polynomial-time algorithm can succeed when k is superconstant, by reduction from a difficult open problem in PAC (probably approximately correct) learning. Finally, we generalize our poly(n/ϵ)-time algorithm to learn any mixture of $k = O(1)$ product distributions over $\{0, 1, \dots, b-1\}^n$, for any $b = O(1)$.

Key words. computational learning theory, PAC learning, mixture distributions, product distributions

AMS subject classifications. 68Q32, 68T05

DOI. 10.1137/060670705

1. Introduction.

1.1. Framework and motivation.

In this paper we study *mixture distributions*. Given distributions $\mathbf{X}^1, \dots, \mathbf{X}^k$ over $\mathbf{R}^n$ and mixing weights $\pi^1, \dots, \pi^k$ that sum to 1, a draw from the mixture distribution $\mathbf{Z}$ is obtained by first selecting i with probability π^i and then making a draw from $\mathbf{X}^i$. Mixture distributions arise in many practical scientific situations as diverse as medicine, geology, and artificial intelligence; indeed, there are several textbooks devoted to the subject [23, 19].

Assuming that data arises as a mixture of some distributions from a class of distributions $\mathcal{C}$, it is natural to try to *learn* the parameters of the mixture components. Our work addresses the learning problem in the PAC-style (probably approximately correct) model introduced by Kearns et al. [18]. In this framework we are given a

[*] Received by the editors September 26, 2006; accepted for publication (in revised form) October 19, 2007; published electronically February 1, 2008. An extended abstract of this work appeared in the *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science* (FOCS), Pittsburgh, PA, 2005, pp. 501–510. This version contains proofs that were omitted from the extended abstract because of space limitations.

http://www.siam.org/journals/sicomp/37-5/67070.html

[†] Google, New York Office, 76 Ninth Avenue, 4th floor, New York, NY 10011 (jonfeld@ieor.columbia.edu). Some of this author's work was done while at the Department of Industrial Engineering and Operations Research, Columbia University, supported by an NSF Mathematical Sciences Postdoctoral Research Fellowship.

[‡] School of Computer Science, Carnegie Mellon University, 7121 Wean Hall, Pittsburgh, PA 15213 (odonnell@cs.cmu.edu). Some of this author's work was done while at the Institute for Advanced Study, supported in part by the National Science Foundation under agreement CCR-0324906. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

[§] Department of Computer Science, Columbia University, 1214 Amsterdam Ave., Mailcode 0401, New York, NY 10027 (rocco@cs.columbia.edu). This author's research was supported in part by NSF CAREER award CCF-0347282.

class $\mathcal{C}$ of probability distributions over $\mathbf{R}^n$ and access to random data sampled from an unknown mixture $\mathbf{Z}$ of k unknown distributions from $\mathcal{C}$. The goal is to output a *hypothesis* mixture $\mathbf{Z}'$ of k distributions from $\mathcal{C}$, which (with high confidence) is ϵ-close to the unknown mixture. The learning algorithm should run in time poly(n/ϵ). The standard notion of "closeness" between distributions $\mathbf{Z}$ and $\mathbf{Z}'$, proposed by Kearns et al. and used in this work, is the *Kullback–Leibler (KL) divergence* (or *relative entropy*), defined as $\mathrm{KL}(\mathbf{Z}||\mathbf{Z}') := \int_x \mathbf{Z}(x)\ln(\mathbf{Z}(x)/\mathbf{Z}'(x))$.[1]

In this paper we learn mixtures of *product distributions* over the Boolean cube $\{0,1\}^n$, and more generally over the b-ary cube $\{0,\ldots,b-1\}^n$; i.e., the classes $\mathcal{C}$ will consist of distributions $\mathbf{X}^i$ whose n coordinates are independent distributions over $\{0,1\}$ and $\{0,\ldots,b-1\}$, respectively.[2] Such learning problems have been well studied in the past, as we now describe.

1.2. Related work. In [18], Kearns et al. gave efficient algorithms for learning mixtures of *Hamming balls*; these are product distributions over $\{0,1\}^n$ in which all the coordinate means $\mathbf{E}[\mathbf{X}^i_j]$ must be either p or $1-p$ for some unknown p which is fixed over all mixture components. Although these algorithms can handle mixtures with $k = O(1)$ many components, the fact that the components are Hamming balls rather than general product distributions is a very strong restriction. (The algorithms also have some additional restrictions: p has to be bounded away from $1/2$, and a more generous learning scenario is assumed in which the learner is in addition given oracle access to the target distribution $\mathbf{Z}$—i.e., it can submit an input x and get back the probability mass that $\mathbf{Z}$ assigns to x.)

More recently, Freund and Mansour [14] gave an efficient algorithm for learning a mixture of two general product distributions over $\{0,1\}^n$. Very roughly speaking, their algorithm uses a "hold-out" set of attributes to approximately reconstruct the line passing through the two means $\mathbf{E}[\mathbf{X}^1]$, $\mathbf{E}[\mathbf{X}^2]$ of the product distributions $\mathbf{X}^1$ and $\mathbf{X}^2$; the algorithm then performs a one-dimensional search on this line for the optimal pair of centers to maximize the likelihood of the data. Around the same time Cryan [8] and Cryan, Goldberg, and Goldberg [9] gave an efficient algorithm for learning phylogenetic trees in the two-state general Markov model. Their algorithm has several stages including estimating covariances between different pairs of leaves, partitioning the leaves into "related sets," constructing a tree for each related set, and then generating an overall tree topology. For the special case in which the tree topology is a star, this gives an algorithm for learning an arbitrary mixture of two product distributions over $\{0,1\}^n$. Both [14] and [8] stated as an open question the problem of obtaining a polynomial-time algorithm for learning a mixture of $k > 2$ product distributions. Indeed, recent work of Mossel and Roch [20] on learning phylogenetic trees argues that the rank-deficiency of transition matrices is a major source of difficulty, and this may indicate why $k = 2$ has historically been a barrier—a two-row matrix can be rank-deficient only if one row is a multiple of the other, whereas the general case of $k > 2$ is much more complex.

In other related work, there is a vast literature in statistics on the general problem of analyzing mixture data—see [19, 21, 23] for surveys. To a large degree this

[1] The KL divergence is often defined in terms of $\log_2$. It is more convenient for us to use ln, and it is easy to see that this choice does not affect our results. We remind the reader (see, e.g., [7]) that $\|\mathbf{Z}-\mathbf{Z}'\|_1 \le \sqrt{2}\sqrt{\mathrm{KL}(\mathbf{Z}||\mathbf{Z}')}$, where $\|\cdot\|_1$ denotes total variation distance; hence if the KL divergence is small, then the total variation distance is also small.

[2] Of course, the algorithm works for product distributions over Σ^n for any alphabet Σ with $|\Sigma| = b$; i.e., the names of the characters in the alphabet do not matter.

work centers on trying to find the exact best mixture model (in terms of likelihood) which explains a given data sample; this is computationally intractable in general. In contrast, our main goal (and the goal of [18, 14, 9, 8, 20]) is to obtain *efficient* algorithms that produce ϵ-close hypotheses.

We also note that there has been recent interest in learning mixtures of n-dimensional Gaussians from the point of view of *clustering* [10, 11, 2, 24]. In this framework one is given samples from a mixture of "well-separated" Gaussians, and the goal is to classify each point in the sample according to which Gaussian it came from. We discuss the relationship between our scenario and this recent literature on Gaussians in section 11; here we emphasize that throughout this paper we make no "separation" assumptions (indeed, no assumptions at all) on the component product distributions in the mixture.

Finally, the problem of learning discrete mixture distributions may have applications to other areas of theoretical computer science, such as database privacy [22, 6] and quantum complexity [1].

1.3. Our results.

In this paper we give an efficient algorithm for learning a mixture of $k = O(1)$ many product distributions over $\{0,1\}^n$. Our main theorem is the following.

THEOREM 1. *Fix any $k = O(1)$, and let $\mathbf{Z}$ be any unknown mixture of k product distributions over $\{0,1\}^n$. Then there is an algorithm that, given samples from $\mathbf{Z}$ and any $\epsilon, \delta > 0$ as inputs, runs in time $\operatorname{poly}(n/\epsilon) \cdot \log(1/\delta)$ and with probability $1-\delta$ outputs a mixture $\mathbf{Z}'$ of k product distributions over $\{0,1\}^n$ satisfying $\mathrm{KL}(\mathbf{Z}||\mathbf{Z}') \le \epsilon$.*

We emphasize that our algorithm requires none of the additional assumptions—such as minimum mixing weights or coordinate means being bounded away from 0, 1/2, or 1—that appear in some work on learning mixture distributions.

Our algorithm runs in time $(n/\epsilon)^{O(k^3)}$, which is polynomial only if k is constant; however, this dependence may be unavoidable. In Theorem 18 we give a reduction from a difficult open question in computational learning theory (the problem of learning decision trees of superconstant size) to the problem of learning a mixture of any superconstant number of product distributions over $\{0,1\}^n$. This implies that solving the mixture learning problem for any $k = \omega(1)$ would require a major breakthrough in learning theory, and suggests that the dependence on k in the exponent of the running time may be unavoidable.

We also generalize our result to learn a mixture of product distributions over $\{0,\ldots,b-1\}^n$ for any constant b, as follows.

THEOREM 2. *Fix any $k = O(1)$ and $b = O(1)$, and let $\mathbf{Z}$ be any unknown mixture of k product distributions over $\{0,\ldots,b-1\}^n$. Then there is an algorithm that, given samples from $\mathbf{Z}$ and any $\epsilon, \delta > 0$ as inputs, runs in time $\operatorname{poly}(n/\epsilon) \cdot \log(1/\delta)$ and with probability $1-\delta$ outputs a mixture $\mathbf{Z}'$ of k product distributions over $\{0,\ldots,b-1\}^n$ satisfying $\mathrm{KL}(\mathbf{Z}||\mathbf{Z}') \le \epsilon$.*

Taking $b = k$, this gives a polynomial-time algorithm for learning k-state Markov evolutionary trees (METs) with a star topology. (Note that the main result of [9, 8] is an algorithm for learning two-state METs with an arbitrary topology; hence our result is not comparable to theirs.)

2. Overview of our approach.

2.1. The WAM algorithm.

The cornerstone of our overall learning algorithms is an algorithm we call WAM (for weights and means). WAM is an algorithm that takes as input a parameter $\epsilon > 0$ and has access to samples from an unknown mixture

$\mathbf{Z}$ of k product distributions $\mathbf{X}^1, \ldots, \mathbf{X}^k$. Here each $\mathbf{X}^i = (\mathbf{X}^i_1, \ldots, \mathbf{X}^i_n)$ is an $\mathbf{R}^n$-valued random vector with independent coordinates. The goal of WAM is to output accurate estimates for all of the *mixing weights* π^i and *coordinate means* $\mu^i_j := \mathbf{E}[\mathbf{X}^i_j]$. Note that a product distribution over $\{0,1\}^n$ is completely specified by its coordinate means.

More precisely, WAM outputs a *list* of poly(n/ϵ) many *candidates* $(\langle \hat{\pi}^1, \ldots, \hat{\pi}^k \rangle, \langle \hat{\mu}^1_1, \hat{\mu}^1_2, \ldots, \hat{\mu}^k_n \rangle)$; each candidate may be viewed as a possible estimate for the correct mixing weights and coordinate means. We will show that with high probability at least one of the candidates output by WAM is *parametrically accurate*; roughly speaking, this means that the candidate is a good estimate in the sense that $|\hat{\pi}^i - \pi^i| \le \epsilon$ for each i and that $|\hat{\mu}^i_j - \mu^i_j| \le \epsilon$ for each i and j. However, there is a slight twist: if a mixing weight π^i is very low, then WAM may not receive any samples from $\mathbf{X}^i$, and thus it is not reasonable to require that WAM get an accurate estimate for $\mu^i_1, \ldots, \mu^i_n$. On the other hand, if π^i is so low, then it is not very important to get an accurate estimate for $\mu^i_1, \ldots, \mu^i_n$ because $\mathbf{X}^i$ has only a tiny effect on $\mathbf{Z}$. We thus make the following formal definition.

DEFINITION 3. *A candidate* $(\langle \hat{\pi}^1, \ldots, \hat{\pi}^k \rangle, \langle \hat{\mu}^1_1, \hat{\mu}^1_2, \ldots, \hat{\mu}^k_n \rangle)$ *is said to be* parametrically ϵ-accurate *if*

1. $|\hat{\pi}^i - \pi^i| \le \epsilon$ *for all* $1 \le i \le k$*;*
2. $|\hat{\mu}^i_j - \mu^i_j| \le \epsilon$ *for all* $1 \le i \le k$ *and* $1 \le j \le n$ *such that* $\pi^i \ge \epsilon$*.*

The main technical theorem in this paper, Theorem 4, shows that so long as the $\mathbf{X}^i$'s take values in a bounded range, WAM will with high probability output at least one candidate that is parametrically accurate. The proof of this theorem uses tools from linear algebra (singular value theory) along with a very careful error analysis.

Remark 1. As will be clear from the proof of Theorem 4, WAM will succeed even if the mixture distributions $\mathbf{X}^i$ are only pairwise independent, not fully independent. This may be of independent interest.

2.2. From WAM to PAC learning (binary case). As we noted already, in the binary case a product distribution on $\{0,1\}^n$ is completely specified by its n coordinate means; thus a candidate can essentially be viewed as a hypothesis mixture of product distributions. (This is not precisely correct, as the candidate mixing weights may not precisely sum to 1 and the candidate means might be outside the range $[0,1]$ by as much as ϵ.) To complete the learning algorithm described in Theorem 1 we must give an efficient procedure that takes the list output by WAM and identifies a candidate distribution that is close to $\mathbf{Z}$ in KL divergence, as required by Theorem 1. We do this in two steps:

1. We first give an efficient procedure that converts a parametrically accurate candidate into a proper hypothesis distribution that is close to $\mathbf{Z}$ in KL divergence. We apply this procedure to each candidate in the list output by WAM, and thus obtain a list of mixtures (hypotheses), at least one of which is close to $\mathbf{Z}$ in KL divergence.
2. We then show that a maximum-likelihood procedure can take a list of hypotheses, at least one of which is good (close to $\mathbf{Z}$ in KL divergence), and identify a single hypothesis which is good.

2.3. Larger alphabets. In the larger alphabet setting, $\mathbf{Z}$ is a mixture of k product distributions $\mathbf{X}^1, \ldots, \mathbf{X}^k$ over $\{0, \ldots, b-1\}^n$. Now each mixture component $\mathbf{X}^i$ is defined by bn parameters $p^i_{j,\ell}$ (with $j = 1, \ldots, n$ and $\ell = 0, \ldots, b-1$), where $p^i_{j,\ell}$ is the probability that a draw from $\mathbf{X}^i_j$ yields ℓ. The simple but useful observation that underlies our extension to $\{0,\dots,b-1\}^n$ is the following: just as any distribution over $\{0,1\}$ is completely specified by its mean, any distribution $\mathbf{X}^i_j$ over $\{0,\dots,b-1\}$ is completely specified by its first $b-1$ moments $\mathbf{E}[\mathbf{X}^i_j]$, $\mathbf{E}[(\mathbf{X}^i_j)^2]$, ..., $\mathbf{E}[(\mathbf{X}^i_j)^{b-1}]$.[3] Our approach is thus to run WAM $b-1$ times; for $\ell = 1,\dots,b-1$ the ℓth run will sample from the given mixture distribution and convert each sample $(z_1,\dots,z_n)$ to the sample $(z_1^\ell,\dots,z_n^\ell)$. We then carefully combine the lists output by the runs of WAM, and follow steps similar to 1 and 2 above to find a good hypothesis in the combined list.

2.4. Outline. Section 3 is dedicated to explaining the ideas behind the WAM algorithm and its proof of correctness. The detailed algorithm and proof are then presented in section 4. We discuss the application of WAM to the b-ary case in section 5. The two steps outlined in section 2.2 are conceptually straightforward, but the details are quite technical; they are given in sections 6 through 8. The pieces are all put together to prove Theorem 2 in section 9 (note that Theorem 1 is a special case of Theorem 2).

In section 10 we detail our reduction from a difficult open question in computational learning theory. We conclude in section 11 with a discussion of applications and future work.

3. The WAM algorithm.

In this section we describe our main algorithm, WAM. We assume a general mixture setting: WAM has access to samples from $\mathbf{Z}$, a mixture of k product distributions $\mathbf{X}^1,\dots,\mathbf{X}^k$ with mixing weights $\pi^1,\dots,\pi^k$. Each $\mathbf{X}^i = (\mathbf{X}^i_1,\dots,\mathbf{X}^i_n)$ is an n-dimensional random variable. We will further assume that all components' coordinates are bounded in the range $[-1,1]$; i.e., $\mathbf{X}^i \in [-1,1]^n$ with probability 1. We have chosen $[-1,1]$ for mathematical convenience; by scaling and translating samples we can get a theorem about any interval such as $[0,1]$ or $[0,(b-1)^{b-1}]$, with an appropriate scaling of ϵ. We write $\mu^i_j := \mathbf{E}[\mathbf{X}^i_j] \in [-1,1]$ for the mean of the jth coordinate of $\mathbf{X}^i$.

Our main theorem is the following.

THEOREM 4. *There is an algorithm* WAM *with the following property: for any* $k = O(1)$ *and any* $\epsilon, \delta > 0$, WAM *runs in time* $\mathrm{poly}(n/\epsilon)\cdot\log(1/\delta)$ *and outputs a list of* $\mathrm{poly}(n/\epsilon)$ *many candidates, at least one of which (with probability at least* $1-\delta$*) is parametrically* ϵ*-accurate.*

We give the full proof of correctness in section 4.2. The remainder of this section is devoted to explaining the main ideas behind the algorithm and its analysis.

3.1. Overview of WAM. There is of course a brute-force way to come up with a list of candidates $(\langle\hat\pi^1,\dots,\hat\pi^k\rangle, \langle\hat\mu^1_1,\hat\mu^1_2,\dots,\hat\mu^k_n\rangle)$, at least one of which is parametrically ϵ-accurate: simply "try all possible values" for the parameters up to additive accuracy ϵ. In other words, try all values $0,\epsilon,2\epsilon,3\epsilon,\dots,1$ for the mixing weights and all values $-1,-1+\epsilon,\dots,1-\epsilon,1$ for the means. We call this approach "gridding." Unfortunately there are $\Theta(n)$ parameters in a candidate, so this naive gridding strategy requires time (and produces a list of length) $(1/\epsilon)^{\Theta(n)}$, i.e., exponential in n, which is clearly unacceptable.

The basic idea behind WAM is as follows: given all pairwise correlations between the coordinates of $\mathbf{Z}$, it can be shown that there are a *constant* number of "key" parameters that suffice to determine all others. Hence in polynomial time we can

[3] This is the case since the distribution can be recovered from the moments by solving a system of linear equations based on a Vandermonde matrix, which has full rank.

empirically estimate all the correlations, try all possibilities for the constantly many key parameters, and then determine the remaining $\Theta(n)$ parameters.

The main challenge in implementing this idea is that it is not at all a priori clear that the error incurred from gridding the key parameters does not "blow up" when these are used to determine the remaining parameters. The heart of our analysis involves showing that it suffices to grid the key parameters to granularity $\mathrm{poly}(\epsilon/n)$ in order to get final error ϵ.

3.2. The algorithm, and intuition for the analysis. We will now go over the steps of the algorithm WAM and at the same time provide an "intuitive" discussion of the analysis. A concise description of the steps of WAM is given at the start of section 4 for the reader's convenience. Throughout this section we will assume for the sake of discussion that the steps we take incur no error; a sketch of the actual error analysis appears in section 3.3.

The first step of WAM is to "grid" the values of the mixing weights $\{\pi^i\}$ to granularity $\epsilon_{\mathrm{wts}} := \epsilon^3$. Since there are only constantly many mixing weights, this costs just a multiplicative factor of $\mathrm{poly}(1/\epsilon)$ in the running time. The remainder of the algorithm then assumes that the mixing weights are known. These mixing weights are of course approximate, but for the purposes of this intuitive description of WAM, we will simply assume that we have exactly correct values for $\{\pi^i\}$.

The next step is simple: suppose that some s of the k mixing weights we have are smaller than ϵ. By the definition of being "ϵ-parametrically accurate," we are not obliged to worry about coordinates with such small mixing weights; hence we will simply forget about these mixture components completely and treat k as $k - s$ in what follows. (We assign arbitrary values for the candidate means of the forgotten components.) We may henceforth assume that $\pi^i \geq \epsilon > 0$ for all i.

The next step of algorithm WAM is to use samples from $\mathbf{Z}$ to estimate the pairwise correlations between the coordinates of $\mathbf{Z}$. Specifically, for all pairs of coordinates $1 \leq j < j' \leq n$, the algorithm WAM empirically estimates

$$\mathrm{corr}(j, j') = \mathbf{E}[\mathbf{Z}_j \mathbf{Z}_{j'}].$$

The estimation will be done to within additive accuracy $\epsilon_{\mathrm{matrix}} = \mathrm{poly}(\epsilon/n)$; specifically, $\epsilon_{\mathrm{matrix}} := \tau^{k+1}$, where $\tau := \epsilon^2/n^2$. With high (i.e., $1-\delta$) confidence we will get good such estimates in time $\mathrm{poly}(n/\epsilon) \cdot \log(1/\delta)$. Again, for the purposes of this intuitive description of WAM we will henceforth assume that we have exactly correct values for each value $\mathrm{corr}(j, j')$. (As an aside, this is the only part of the algorithm that uses samples from $\mathbf{Z}$; as we will shortly see, this justifies Remark 1.)

Observe that since $\mathbf{X}^i_j$ and $\mathbf{X}^i_{j'}$ are (pairwise) independent we have

$$\mathrm{corr}(j, j') = \mathbf{E}[\mathbf{Z}_j \mathbf{Z}_{j'}] = \sum_{i=1}^{k} \pi^i \mathbf{E}[\mathbf{X}^i_j \mathbf{X}^i_{j'}] = \sum_{i=1}^{k} \pi^i \mathbf{E}[\mathbf{X}^i_j] \mathbf{E}[\mathbf{X}^i_{j'}] = \sum_{i=1}^{k} \pi^i \mu^i_j \mu^i_{j'}.$$

Let us define

$$\tilde{\mu}^i_j = \sqrt{\pi^i} \mu^i_j$$

and write $\tilde{\mu}_j = (\tilde{\mu}^1_j, \tilde{\mu}^2_j, \ldots, \tilde{\mu}^k_j) \in [-1, 1]^k$ for $1 \leq j \leq n$. We thus have

$$\mathrm{corr}(j, j') = \tilde{\mu}_j \cdot \tilde{\mu}_{j'},$$

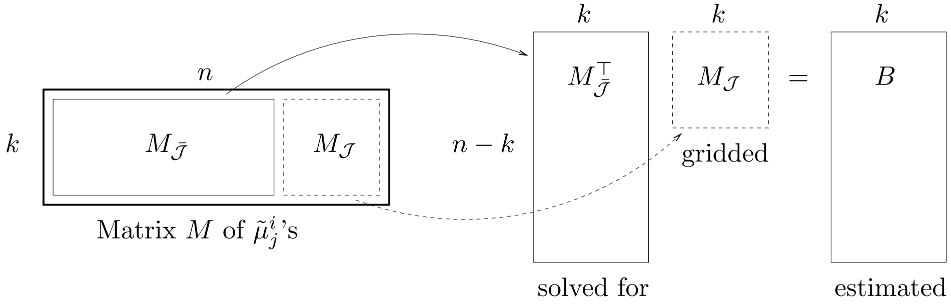


FIG. 1. *The full rank case. We solve for the unknown $\tilde{\mu}^i_j$'s in $M_{\bar{\mathcal{J}}}$ using the gridded values in $M_{\mathcal{J}}$ and the values in B estimated directly from the samples.*

where $\cdot$ denotes the dot product in $\mathbf{R}^k$. The remaining task for WAM is to determine all the values μ^i_j. Since WAM already has values for each π^i and each $\pi^i \geq \epsilon > 0$, it suffices for WAM to determine all the values $\tilde{\mu}^i_j$ and then divide by $\sqrt{\pi^i}$.

At this point WAM has empirically estimated values for all the pairwise dot products $\tilde{\mu}_j \cdot \tilde{\mu}_{j'}$, $j \neq j'$, and as mentioned, for intuitive purposes we are assuming that all of these estimates are exactly correct. Let M denote the $k \times n$ matrix whose (i, j) entry is the unknown $\tilde{\mu}^i_j$; i.e., the jth column of M is $\tilde{\mu}_j$. The statement that WAM has all the dot products $\tilde{\mu}_j \cdot \tilde{\mu}_{j'}$ for $j \neq j'$ is equivalent to saying that WAM has all the *off-diagonal* entries of the Gram matrix $M^\top M$. We are thus led to what is essentially the central problem WAM solves:

Central task. Given (estimates) for the off-diagonal entries of the $n \times n$ Gram matrix $M^\top M$, generate (estimates of) all possible candidates for the entries of the $k \times n$ matrix M.

(Note: The diagonal entries of $M^\top M$ are the quantities $\tilde{\mu}_j \cdot \tilde{\mu}_j = \sum_{i=1}^k \pi^i (\mu^i_j)^2$, and there is no obvious way to estimate these quantities using samples from $\mathbf{Z}$. Also there are n such quantities, which is too many to "grid over." Nevertheless, the fact that we are missing the diagonal entries of $M^\top M$ will not play an important role for WAM.)

In general, a complete $n \times n$ Gram matrix determines the original $k \times n$ matrix up to isometries on $\mathbf{R}^k$. Such isometries can be described by $k \times k$ orthonormal matrices, and these k^2 "degrees of freedom" roughly correspond to the constantly many key parameters that we grid over in the end. A geometric intuition for the central task is the following: there are n unknown vectors in $\mathbf{R}^k$ and we have all the "angles" (more precisely, the dot products) between them. Thus fixing k of the vectors (hence k^2 unknown coordinates) is enough to completely determine the remainder of the vectors.

The full rank case. We proceed with our intuitive description of WAM and show how to solve the central task *when M has full rank.* Having done this, we will give the actual steps of the algorithm that show how the full rank assumption can be removed.

So suppose for now that M has full rank. Then there exists some set of k columns of M that are linearly independent, say $\mathcal{J} = \{j_1, \ldots, j_k\} \subset [n]$. Algorithm WAM tries all $\binom{n}{k} = \text{poly}(n)$ possibilities for the set $\mathcal{J}$ and then grids over the vectors $\tilde{\mu}_{j_1}, \ldots, \tilde{\mu}_{j_k}$ with granularity $\epsilon_{\text{matrix}} = \text{poly}(\epsilon/n)$ in each coordinate. As usual for the purposes of intuition, we assume that we now have $\tilde{\mu}_{j_1}, \ldots, \tilde{\mu}_{j_k}$ exactly correct.

Let $M_{\mathcal{J}}$ be the $k \times k$ matrix given by the $\mathcal{J}$-columns of M, and let $M_{\bar{\mathcal{J}}}$ be the $k \times (n - k)$ matrix given by deleting the $\mathcal{J}$-columns of M. WAM now has the entries of $M_{\mathcal{J}}$ and must compute the remaining unknowns, $M_{\bar{\mathcal{J}}}$. Since WAM has all of the off-diagonal entries of $M^{\top}M$, it has all of the values of $B = M_{\bar{\mathcal{J}}}^{\top} M_{\mathcal{J}}$. (See Figure 1.) However, the columns of $M_{\mathcal{J}}$ are linearly independent, so $M_{\mathcal{J}}$ is invertible, and hence WAM can compute $M_{\bar{\mathcal{J}}}^{\top} = BM_{\mathcal{J}}^{-1}$ in poly(n) time. Having done this, WAM has all the entries of M, and so the central task is complete, as is the algorithm.

The general case. Of course, in general, M does not have full rank. This represents the main conceptual problem we faced in rigorously solving the central task. Indeed, we believe that handling rank-deficiency is the chief conceptual problem for the whole learning mixtures question, and that our linear algebraic methods for overcoming it (the description of which occupies the remainder of section 3) are the main technical contribution of this paper.

Suppose rank$(M) = r < k$. By trying all possible values (only constantly many), algorithm WAM can be assumed to know r. Now by definition of rank$(M) = r$ there must exist $k - r$ orthonormal vectors $u_{r+1}, \ldots, u_k \in [-1, 1]^k$ which are orthogonal to all columns of M. WAM grids over these vectors with granularity ϵ_{matrix}, incurring another multiplicative poly(n/ϵ) factor in the running time. As usual, assume for the intuitive discussion that we now have the u_j's exactly. Let these vectors be adjoined as columns to M, forming M'. But now the matrix M' has full rank; furthermore, WAM knows all the off-diagonal elements of $(M')^{\top}M'$, i.e., all the pairwise dot products of M''s columns, since all of the new dot products which involve the u_j's are simply 0! Thus we now have an instance of the central task with a full rank matrix, a case we already solved. (Technically, n may now be as large as $n + (k - 1)$, but this is still $O(n)$, and hence no time bound is affected.) Solving the central task on M' (which contains all the entries of M) completes the algorithm WAM in the rank-deficient case.

3.3. Sketch of the actual analysis of WAM. The preceding intuitive discussion of algorithm WAM neglected all error analysis. Correctly handling the error analysis is the somewhat subtle issue we discuss in this section. As mentioned, the full proof is given in section 4.2.

The main issue in the error analysis comes in understanding the right notion of the rank of M—since all of our gridding inevitably yields only approximations of the entries of M, the actual notion of rank is far too fragile to be of use. Recall the outline of the algorithm in our idealized intuition (rank-deficient case):

$$\begin{aligned}
&r = \text{dimension of subspace in which } \tilde{\mu}_j\text{'s lie}\\
&\qquad \Rightarrow \text{augment } M \text{ by } k - r \text{ orthogonal } u_i\text{'s, forming } M' \Rightarrow M' \text{ now full rank}\\
&\qquad\quad \Rightarrow \text{find nonsingular } k \times k \text{ submatrix } M'_{\mathcal{J}} \Rightarrow \text{solve linear system } {M'}_{\bar{\mathcal{J}}}^{\top} M'_{\mathcal{J}} = B.
\end{aligned}$$

For the purposes of the error analysis, we reinterpret the operation of WAM as follows:

$$\begin{aligned}
(1)\quad & r^* = \text{dimension of subspace in which the } \tilde{\mu}_j\text{'s ``essentially'' lie}\\
&\qquad \Rightarrow \text{augment } M \text{ by } k - r \text{ ``essentially'' orthogonal } u_i\text{'s, forming } M'\\
&\qquad\qquad \Rightarrow M' \text{ now ``strongly'' full rank}\\
&\Rightarrow \text{find ``strongly'' nonsingular } k \times k \text{ submatrix } M'_{\mathcal{J}} \Rightarrow \text{solve linear system } {M'}_{\bar{\mathcal{J}}}^{\top} M'_{\mathcal{J}} = B.
\end{aligned}$$

The real difficulty of the error analysis comes in the last step: controlling the error incurred from solving the linear system. Since we will have only approximately correct values for the entries of $M'_{\mathcal{J}}$ and B, we need to analyze the additive error arising from solving a perturbed linear system. Standard results from numerical analysis (see Proposition 9 in section 4.1) allow us to bound this error by a function of (i) the error in $M'_{\mathcal{J}}$ and B, and (ii) the smallest *singular value* of $M'_{\mathcal{J}}$, denoted by $\sigma_k(M')$. More precisely, as we describe in Proposition 9, the error is bounded by the errors in $M_{\mathcal{J}}$ and B normalized by $\sigma_k(M')$.

Let us briefly recall some notions related to singular values. Given any $k \times n$ matrix M, the first (largest) singular value of M is $\sigma_1(M) = \max_{\|u_1\|_2=1} \|u_1^\top M\|_2$, and a u_1 achieving this maximum is taken as the first *(left) singular vector* of M. The second singular value of M is $\sigma_2(M) = \max_{\|u_2\|_2=1,\, u_2 \perp u_1} \|u_2^\top M\|_2$, and u_2 is the second left singular vector of M. In general, the ith singular value and vector are given by maximizing over all $\|u_i\|_2 = 1$ orthogonal to all $u_1, \ldots, u_{i-1}$. In a well-defined sense (the Frobenius norm), the smallest singular value $\sigma_k(M)$ measures the distance of M from being singular.

WAM's final error bounds arise from dividing the error in its estimates for $M'_{\mathcal{J}}$ and B by the smallest singular value of $M'_{\mathcal{J}}$. The error in the estimates for the entries of $M'_{\mathcal{J}}$ come from gridding, and thus can essentially be made as small as desired; WAM makes them smaller than ϵ_{matrix}. The errors in B come from two sources: some of the entries of B are estimates of quantities $\tilde{\mu}_j \cdot \tilde{\mu}_{j'} = \text{corr}(j, j')$, and again these errors can be made essentially as small as desired, smaller than ϵ_{matrix}. However, the other errors in B come from approximating the quantities $\tilde{\mu}_j \cdot u_i$ by 0, i.e., assuming that the augmenting vectors are orthogonal to the columns of M.

As the reader may by now have guessed, the vectors with which WAM attempts to augment M will be the last $k - r^*$ singular vectors of M, $u_{r^*+1}, \ldots, u_k$. The hope is that for an appropriate choice of r^* these singular vectors will be "essentially" orthogonal to the columns of M, and that the resulting M' will be "strongly" full rank, in the sense that $\sigma_k(M')$ will be somewhat large (cf. (1)). One can show (see Proposition 8 of section 4.1) that the extent to which the u_i's are orthogonal to the columns of M is controlled by the $(r^* + 1)$th singular value of M; i.e., $|\tilde{\mu}_j \cdot u_i| \leq \sigma_{r^*+1}(M)$ for all $i \geq r^* + 1$; this is precisely the error we incur for the zero entries in B. On the other hand, one can also show that the augmented M' has smallest singular value at least $\sigma_{r^*}(M)$. Thus we are motivated to choose r^* so as to get a large multiplicative gap between $\sigma_{r^*}(M)$ and $\sigma_{r^*+1}(M)$, as follows.

DEFINITION 5. *Given $\tau > 0$, the* τ-essential rank *of M is*

$$r^*(M) = r^*_\tau(M) = \min\{0 \leq r \leq k : \sigma_{r+1}(M)/\sigma_r(M) \leq \tau\},$$

where we take $\sigma_0(M) = 1$ and $\sigma_{k+1}(M) = 0$.

One might think that if the additive error incurred from solving the linear system were to be roughly $\sigma_{r^*}(M)/\sigma_{r^*+1}(M)$, then it should suffice to select τ on the order of poly(ϵ). However, there is still a missing piece of the analysis: although the smallest singular value of M' becomes at least $\sigma_{r^*}(M)$ after adjoining the u_j's, we use only a $k \times k$ *submatrix* $M'_{\mathcal{J}}$ to solve the linear system. Is it the case that if M' has a large smallest singular value then its "best" $k \times k$ submatrix also has a somewhat large smallest singular value? We need a quantitative version of the fact that a nonsingular $k \times n$ matrix has a $k \times k$ nonsingular submatrix (again, cf. (1)).

This does not seem to be a well-studied problem, and indeed there are some open questions in linear algebra surrounding the issue. It is possible to derive an extremely weak quantitative result of the required nature using the Cauchy–Binet formula. We instead give the following quantitatively strong version.

COROLLARY 6. *Let A be a $k \times n$ real matrix with $\sigma_k(A) \geq \epsilon$. Then there exists a subset of columns $\mathcal{J} \subseteq [n]$ with $|\mathcal{J}| = k$ such that $\sigma_k(A_{\mathcal{J}}) \geq \epsilon/\sqrt{k(n-k)+1}$.*

(We call the result a corollary because our proof in section 4.1 is derived from a 1997 linear algebraic result of Goreinov, Tyrtyshnikov, and Zamarashkin [15]. Incidentally, it is conjectured in their paper that $\sqrt{k(n-k)+1}$ can be replaced by $\sqrt{n}$.)

With this result in hand it becomes sufficient to take $\tau = \epsilon^2/n^2$, as described in the previous section. Now the error analysis can be completed:

- If M has a singular value gap of τ and so has essential rank $r^* < k$, then when WAM tries out the appropriate r^* and singular vectors, the error it incurs from solving the linear system is roughly at most $O(\sqrt{n}\tau) = O(\epsilon^2/n^{3/2})$, and as we show at the end of section 4.2, having this level of control over errors in solving the linear system for the unknown $\tilde{\mu}^i_j$'s lets us obtain the final μ^i_j values to the required ϵ-accuracy.
- On the other hand, if M has no singular value gap smaller than τ, then its smallest singular value is at least τ^k; thus it suffices to take $\epsilon_{\text{matrix}} = \tau^{k+1} = \text{poly}(\epsilon/n)$ to control the errors in the full rank case.

See section 4.2 for the detailed proof of correctness.

4. Algorithm WAM.

Algorithm WAM has access to samples from the mixture $\mathbf{Z}$ and takes as input parameters $\epsilon, \delta > 0$.

Algorithm WAM.

```
1. Let $\epsilon_{\text{wts}} = \epsilon^3$, $\tau = \epsilon^2/n^2$, and $\epsilon_{\text{matrix}} = \tau^{k+1}$.
2. Grid over the mixing weights, producing values $\hat{\pi}^1, \ldots, \hat{\pi}^k \in [0,1]$ accurate to within $\pm\epsilon_{\text{wts}}$. If $s$ of these weights are smaller than $\epsilon - \epsilon_{\text{wts}}$, eliminate them and treat $k$ as $k-s$ in what follows.
3. Make empirical estimates $\widehat{\text{corr}}(j,j')$ for all correlations $\text{corr}(j,j') = \mathbf{E}[\mathbf{Z}_j \mathbf{Z}_{j'}] = \tilde{\mu}_j \cdot \tilde{\mu}_{j'}$ for $j \neq j'$ to within $\pm\epsilon_{\text{matrix}}$, with confidence $1-\delta$.
4. Let $M$ be the $k \times n$ matrix of unknowns $(M_{ij}) = (\tilde{\mu}^i_j)$, and try all possible integers $0 \le r^* \le k$ for the essential rank of $M$.
5. Grid over $k - r^*$ vectors $\hat{u}_{r^*+1}, \ldots, \hat{u}_k \in [-1,1]^k$ to within $\pm\epsilon_{\text{matrix}}$ in each coordinate and augment $M$ with these as columns, forming $\widehat{M'}$.
6. Try all possible subsets of exactly $k$ column indices of $\widehat{M'}$; write these indices as $\mathcal{J} = J \cup J'$, where $J$ corresponds to columns from the original matrix $M$ and $J'$ corresponds to augmented columns. Grid over $[-1,1]$ for the entries of $M$ in columns $J$ to within $\pm\epsilon_{\text{matrix}}$, yielding $\{\hat{\tilde{\mu}}^i_j : i \in [k], j \in J\}$. Let $\widehat{M'_{\mathcal{J}}}$ denote the matrix of estimates for all the columns in $\mathcal{J}$. (See Figure 2.)
7. Let $\bar{\mathcal{J}}$ denote the columns of $M$ other than $J$, and let $M_{\bar{\mathcal{J}}}$ denote the matrix of remaining unknowns formed by these columns. Let $\widehat{B}$ be the matrix with rows indexed by $\bar{\mathcal{J}}$ and columns indexed by $\mathcal{J}$ whose $(j,j')$ entry is the estimate $\widehat{\text{corr}}(j,j')$ of $\tilde{\mu}_j \cdot \tilde{\mu}_{j'}$ if $j' \in J$, or is $0$ if $j' \in J'$. Using the entries of $\widehat{B}$ and $\widehat{M'_{\mathcal{J}}}$ (all of which are known), solve the system $M_{\bar{\mathcal{J}}}^{\top} \widehat{M'_{\mathcal{J}}} = \widehat{B}$ to obtain estimates $\hat{\tilde{\mu}}^i_j$ for the entries of $M_{\bar{\mathcal{J}}}$ (which are the unknown $\tilde{\mu}^i_j$'s), thus producing estimates $\hat{\tilde{\mu}}^i_j$ for all entries of $M$. (If the matrix $\widehat{M'_{\mathcal{J}}}$ is singular, simply abandon the current
```

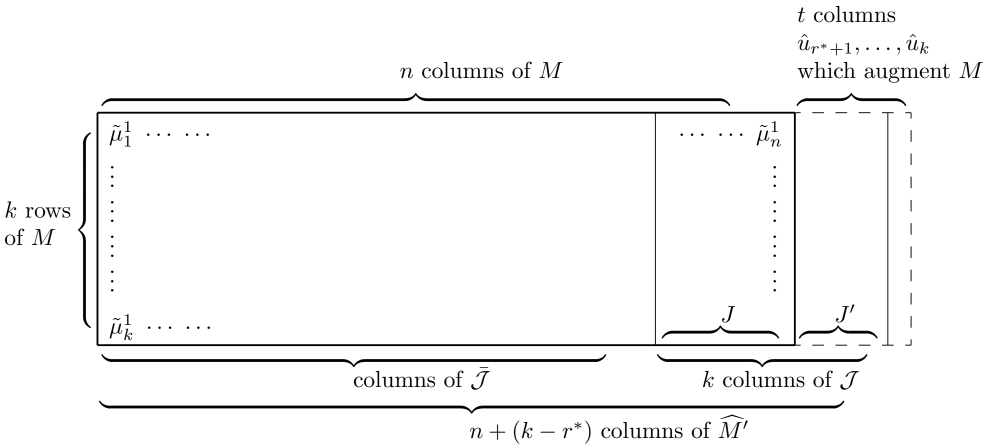

t columns
$\hat{u}_{r^*+1}, \ldots, \hat{u}_k$
which augment M
n columns of M
k rows of M
$\tilde{\mu}_1^1$
$\tilde{\mu}_n^1$
$\tilde{\mu}_k^1$
J
J'
columns of $\bar{\mathcal{J}}$
k columns of $\mathcal{J}$
$n + (k - r^*)$ columns of $\widehat{M}'$

FIG. 2. *A depiction of the matrix used by* WAM. *For ease of illustration the columns J of M are depicted as being the rightmost columns of M, and the columns J' from the augmenting columns $\hat{u}_{k-t+1}, \ldots, \hat{u}_k$ are depicted as being the leftmost of those augmenting columns.*

```
   gridding.)
8. From the estimated values $\hat{\tilde{\mu}}_j^i$, compute the estimates $\hat{\mu}_j^i = \hat{\tilde{\mu}}_j^i / \sqrt{\hat{\pi}^i}$
   for all $i, j$.  (Note that $\hat{\pi}^i$ is never 0, since each is at least
   $\epsilon - \epsilon_{\rm wts} > 0$.)
9. Output the candidate $(\langle \hat{\pi}^1, \ldots, \hat{\pi}^k \rangle, \langle \hat{\mu}_1^1, \hat{\mu}_2^1, \ldots, \hat{\mu}_n^k \rangle)$.
```

4.1. Linear algebra necessities. In this section we give the results from linear algebra and numerical analysis necessary for the analysis of WAM.

Let $A = (a_{ij})$ be any $k \times n$ real matrix, and write its singular value decomposition as $A = U\Sigma V$. Here U is a $k \times k$ matrix with orthonormal columns $u_1, \ldots, u_k$, Σ is a $k \times k$ diagonal matrix with $\sigma_1(A) \geq \cdots \geq \sigma_k(A) \geq 0$ on the diagonal, and V is a $k \times n$ matrix with orthonormal rows. We let $\sigma_1(A) \geq \cdots \geq \sigma_k(A) \geq 0$ denote the singular values of A, and let $u_1, \ldots, u_k$ denote the corresponding left singular vectors of A, i.e., the columns of U. If it is clear from context, we simply write σ_i for $\sigma_i(A)$. Recall that

- the vectors $u_1, \ldots, u_k$ form an orthonormal basis for $\mathbf{R}^k$;
- $\sigma_1 = \max_{\|x\|_2=1} \|x^\top A\|_2$ and $\sigma_k = \min_{\|x\|_2=1} \|x^\top A\|_2$.

The *Frobenius norm* $\|A\|_F$ of a $k \times n$ matrix A is defined as $\|A\|_F = \sqrt{\sum_{i,j} (A_{i,j})^2}$. We recall the well-known fact that $\sigma_k(A)$ equals the Frobenius norm distance from the $k \times n$ matrix A to the nearest rank-deficient matrix $\tilde{A}$, i.e.,

$$\sigma_k(A) = \min_{\mathrm{rank}(\tilde{A}) < k} \|A - \tilde{A}\|_F.$$

The *spectral norm* $\|A\|_2$ of a $k \times n$ matrix A is $\|A\|_2 = \max_{\|x\|_2=1} \|Ax\|$. It is well known that $\|A\|_2 = \sigma_1$ and $\|A\|_F = \sqrt{\sigma_1^2 + \cdots + \sigma_k^2}$; note that this implies $\|A\|_2 \leq \|A\|_F$.

Our first necessary result is a quantitative version of the elementary fact that a full rank $k \times n$ matrix has a full rank $k \times k$ submatrix. We will use the following theorem of Goreinov, Tyrtyshnikov, and Zamarashkin [15].

THEOREM 7 (see [15]). *Let V be a $k \times n$ real matrix with orthonormal rows. Then there is a $k \times k$ submatrix V_J which has $\sigma_k(V_J) \geq 1/\sqrt{k(n-k)+1}$.*

The result we need is an easy corollary, Corollary 6 given above.

Proof of Corollary 6. Recall that by the singular value decomposition we have $A = U\Sigma V$, where U is a $k \times k$ matrix with orthonormal columns, Σ is a $k \times k$ diagonal matrix with diagonal entries $\sigma_1, \dots, \sigma_k$, and V is a $k \times n$ matrix with orthonormal rows. Let V_J be the $k \times k$ submatrix of V whose existence is asserted by Theorem 7, so $\sigma_k(V_J) \geq 1/\sqrt{k(n-k)+1}$. We have $\sigma_k(U) = 1$ (since U is an orthogonal matrix) and $\sigma_k(\Sigma) \geq \epsilon$, so

$$\sigma_k(U\Sigma V_J) \geq \sigma_k(U)\sigma_k(\Sigma)\sigma_k(V_J) \geq \epsilon/\sqrt{k(n-k)+1},$$

where the inequality holds since $\sigma_k(PQ) \geq \sigma_k(P)\sigma_k(Q)$ for any $k \times k$ matrices P, Q. (This is easily seen from the variational characterization $\sigma_k(P) = \min_{\|x\|_2=1} \|x^\top P\|_2$.) The corollary follows by observing that $U\Sigma V_J$ is the $k \times k$ submatrix of A whose columns are in J. □

The next result we will need is the characterization of what happens when the last $k - r^*$ left singular vectors of a matrix are adjoined to it.

PROPOSITION 8. *Let A be a $k \times n$ matrix with columns $a_1, \dots, a_n$. Fix any r^*, and let $u_{r^*+1}, \dots, u_k$ be the left singular vectors corresponding to the smallest singular values $\sigma_{r^*+1}, \dots, \sigma_k$ of A. Let A' be A with the vectors $u_{r^*+1}, \dots, u_k$ adjoined as columns. Then*

$$\sigma_k(A') \geq \min\{1, \sigma_{r^*}(A)\},$$

and for all $r^* + 1 \leq \ell \leq k$ and for all columns a_j of A we have

$$|a_j \cdot u_\ell| \leq \sigma_{r^*+1}(A).$$

Proof. Write the singular value decomposition $A = U\Sigma V$, where U is a $k \times k$ matrix with orthonormal columns $u_1, \dots, u_k$, Σ is a $k \times k$ diagonal matrix with $\sigma_1 \geq \cdots \geq \sigma_k \geq 0$ on the diagonal, and V is a $k \times n$ matrix with orthonormal rows. It follows that for any vector $x \in \mathbf{R}^k$ we have

$$\|x^\top A\|_2^2 = \sigma_1^2(x^\top u_1)^2 + \cdots + \sigma_k^2(x^\top u_k)^2.$$

Let R denote the $k \times (k - r^*)$ matrix whose columns are $u_{r^*+1}, \dots, u_k$, so we have $A' = [A\ R]$. It is easily verified that the left singular vectors of R are simply $u_{r^*+1}, \dots, u_k$, while the singular values of R are all 1. Consequently we have

$$\|x^\top R\|_2^2 = (x^\top u_{r^*+1})^2 + \cdots + (x^\top u_k)^2$$

for any $x \in \mathbf{R}^k$.

Now recall the variational characterization of $\sigma_k(A')$, namely $\sigma_k(A') = \min_{\|x\|_2=1} \|x^\top A'\|_2$. Since $\|x^\top A'\|_2 = \sqrt{\|x^\top A\|_2^2 + \|x^\top R\|_2^2}$, we have

$$\sigma_k(A') = \min_{\|x\|_2=1} \sqrt{\sigma_1^2(x^\top u_1)^2 + \cdots + \sigma_k^2(x^\top u_k)^2 + (x^\top u_{r^*+1})^2 + \cdots + (x^\top u_k)^2}. \tag{2}$$

Since $u_1, \dots, u_k$ form an orthonormal basis for $\mathbf{R}^k$ we have that $(x^\top u_1)^2 + \cdots + (x^\top u_k)^2 = 1$ for all $\|x\|_2 = 1$. If we let $\alpha_x = (x^\top u_{r^*+1})^2 + \cdots + (x^\top u_k)^2$, then the quantity inside the square root of (2) is at least $\sigma_{r^*}^2(1 - \alpha_x) + \alpha_x \geq \min\{\sigma_{r^*}^2, 1\}$. This proves the first inequality of the proposition.

For the second inequality, we observe that $a_j \cdot u_\ell = u_\ell^\top U\Sigma v_j$, where v_j is the jth column of V. Since U is orthonormal and $\Sigma_{\ell,\ell} = \sigma_\ell$ we thus have

$$|u_\ell^\top U\Sigma v_j| = |\sigma_\ell v_{\ell,j}| \le \sigma_\ell \le \sigma_{r^*+1},$$

where the first inequality holds since the rows of V are orthonormal, and hence each entry of V must be at most 1 in magnitude. □

The final result we will need involves controlling the error in a perturbed linear system in terms of the smallest singular value. Although we could not find the following statement in the literature, it should be considered a very basic result from numerical analysis.

PROPOSITION 9. *Let A be a nonsingular $k \times k$ matrix, b be a k-dimensional vector, and x be the solution to $Ax = b$. Assume that $\|x\|_\infty \le 1$. Suppose A' is a $k \times k$ matrix such that each entry of $A - A'$ is at most ϵ_{matrix} in magnitude, and assume that $\epsilon_{\text{matrix}} < \sigma_k(A)/2k$. Let b' be a k-dimensional vector satisfying $\|b - b'\|_\infty \le \epsilon_{\text{rhs}}$. Let x' be the solution to $A'x' = b'$. Then we have*

$$\|x - x'\|_\infty \le O(k^{3/2})\frac{\epsilon_{\text{matrix}} + \epsilon_{\text{rhs}}}{\sigma_k(A)}.$$

Proof. Write $E = A - A'$ and $\eta = b - b'$. By our assumption on A' we have $\|E\|_F = \|A - A'\|_F \le k\epsilon_{\text{matrix}}$. By our assumption on ϵ_{matrix} this is at most $\sigma_k(A)/2$. It follows that $\sigma_k(A') \ge \sigma_k(A)/2$, and in particular A' is nonsingular. Thus x' is indeed well defined, and we may write

$$\begin{aligned} Ax - Ax' = Ex' + \eta \Rightarrow x - x' &= A^{-1}(Ex' + \eta) \\ \Rightarrow \|x - x'\|_2 &\le \|A^{-1}\|_2(\|E\|_2\|x'\|_2 + \|\eta\|_2) \\ &\le \frac{1}{\sigma_k(A)}\Big((k\epsilon_{\text{matrix}})(\|x - x'\|_2 + \|x\|_2) + (\sqrt{k}\epsilon_{\text{rhs}})\Big) \\ \Rightarrow (\sigma_k(A) - k\epsilon_{\text{matrix}})\|x - x'\|_2 &\le k\epsilon_{\text{matrix}}\|x\|_2 + \sqrt{k}\epsilon_{\text{rhs}}. \end{aligned}$$

We now use $\sigma_k(A) - k\epsilon_{\text{matrix}} \ge \sigma_k(A)/2$ to conclude

$$\|x - x'\|_2 \le \frac{2(k\epsilon_{\text{matrix}}\|x\|_2 + \sqrt{k}\epsilon_{\text{rhs}})}{\sigma_k(A)}.$$

Finally, $\|x - x'\|_\infty \le \|x - x'\|_2$ and $\|x\|_2 \le \sqrt{k}$ complete the proof. □

4.2. Proof of Theorem 4. We go through the algorithm step by step, as it appears at the start of section 4. In step 1 of WAM, we define constants $\epsilon_{\text{wts}} = \epsilon^3$, $\tau = \epsilon^2/n^2$, and $\epsilon_{\text{matrix}} = \tau^{k+1}$, which we use throughout the proof.

In step 2 of WAM, the algorithm will grid over estimates $\hat{\pi}^i$ that satisfy $|\hat{\pi}^i - \pi^i| \le \epsilon_{\text{wts}}$ for all i. In this case, any mixing component $\mathbf{X}^i$ whose mixing weight π^i is at least ϵ will not be eliminated. Since we need not be concerned with accuracy for the means of the other mixing components, we can ignore them and assume for the rest of the proof that $\pi^i \ge \epsilon$ for all i.

Now we come to the main work in the proof of correctness of Theorem 4: namely, showing that in steps 3–7 of Algorithm WAM, accurate estimates for the $\tilde{\mu}_j^i$'s are produced. Our goal for most of the remainder of the proof will be to show that we obtain estimates $\hat{\tilde{\mu}}_j^i$ satisfying

$$|\hat{\tilde{\mu}}_j^i - \tilde{\mu}_j^i| \le \tilde{\epsilon} := \epsilon^2$$

for all i.

To that end, let $r^* = r^*_\tau(M)$, the τ-essential rank of M. We will quickly dismiss the two easy cases, $r^* = 0$ and $r^* = k$; we then treat the general case $0 < r^* < k$.

$r^* = 0$ *case.* By definition, in this case $\sigma_1(M) \le \tau \le \tilde{\epsilon}$. Since $\sigma_1(M)$ is at least as large as the magnitude of M's largest entry, we must therefore have $|\tilde{\mu}^i_j| \le \tilde{\epsilon}$ for all i, j. Now when WAM tries $r^* = 0$ in step 4, tries the k standard basis vectors for $\hat{u}_1, \ldots, \hat{u}_k$ in step 5, and chooses all of these vectors for $\mathcal{J}$ in step 6, it will set $\widehat{B} = 0$ in step 7 and get $\hat{\tilde{\mu}}^i_j = 0$ for all i, j when it solves the linear system. But this is indeed within an additive $\tau \le \tilde{\epsilon}$ of the true values, as desired.

$r^* = k$ *case.* By definition, it's not hard to see that in this case we must have $\sigma_k(M) \ge \tau^k$. Now consider when WAM tries $r^* = k$ in step 4. Step 5 becomes vacuous. By Corollary 6 there is some set of k columns $\mathcal{J} = J$ such that $\sigma_k(M_{\mathcal{J}}) \ge \sigma_k(M)/\sqrt{k(n-k)+1} \ge \tau^k/n$. In step 6, WAM will try out this $\mathcal{J}$ and grid the associated entries to within $\pm\epsilon_{\text{matrix}}$. In step 7 the algorithm will use only $\widehat{\text{corr}}$'s in forming $\widehat{B}$, and these will also be correct to within an additive $\pm\epsilon_{\text{matrix}}$. We can now use Proposition 9—note that $\epsilon_{\text{matrix}} = \tau^{k+1} \le (\tau^k/n)/2k \le \sigma_k(M_{\mathcal{J}})/2k$, as necessary. This gives estimates in step 7 satisfying

$$|\hat{\tilde{\mu}}^i_j - \tilde{\mu}^i_j| \le O(k^{3/2}) \frac{2\epsilon_{\text{matrix}}}{\tau^k/n} = O(k^{3/2} n \tau) \le \tilde{\epsilon},$$

as desired.

$0 < r^* < k$ *case.* In this case, by definition of the essential rank, we have

$$\tau \sigma_{r^*}(M) \ge \sigma_{r^*+1}(M) \ge \tau^k. \tag{3}$$

In step 4 WAM will try out the correct value for r^*, and in step 5 WAM will grid over vectors $\hat{u}_{r^*+1}, \ldots, \hat{u}_k$ that are within $\pm\epsilon_{\text{matrix}}$ in each coordinate of the actual last left singular vectors of M, $u_{r^*+1}, \ldots, u_k$. Let M' denote the matrix M with these true singular vectors adjoined. By Proposition 8 we have

$$\sigma_k(M') \ge \min\{1, \sigma_{r^*}(M)\}. \tag{4}$$

From the crude upper bound $\sigma_{r^*}(M) \le \|M\|_F = \sqrt{\sum_{i,j} (\tilde{\mu}^i_j)^2} \le \sqrt{kn}$, we can restate (4) as simply $\sigma_k(M') \ge \sigma_{r^*}(M)/\sqrt{kn}$. Now applying Corollary 6, we conclude there is a subset $\mathcal{J}$ of M''s columns with $|\mathcal{J}| = k$ such that

$$\sigma_k(M'_{\mathcal{J}}) \ge \sigma_k(M')/\sqrt{k(n-k)+1} \ge \sigma_{r^*}(M)/kn. \tag{5}$$

In step 6, WAM will try this set of columns $\mathcal{J} = J \cup J'$; it will also grid estimates for the entries in this column that are correct up to an additive $\pm\epsilon_{\text{matrix}}$. Note that WAM now has an $\widehat{M'_{\mathcal{J}}}$ that has all entries correct up to an additive $\pm\epsilon_{\text{matrix}}$. Now consider the matrix $\widehat{B}$ WAM forms in step 7. For the columns corresponding to J the entries are given by $\widehat{\text{corr}}$'s, which are correct to within $\pm\epsilon_{\text{matrix}}$. For the columns corresponding to J' the entries are 0's; by the second part of Proposition 8 these are correct up to an additive $\sigma_{r^*+1}(M)$. We now use Corollary 6 to bound the error resulting from solving the system $M^\top_{\mathcal{J}} \widehat{M'}_{\mathcal{J}} = \widehat{B}$ in step 7. To check that the necessary hypothesis is satisfied we combine (3) and (5):

$$\sigma_k(M'_{\mathcal{J}})/2k \ge \sigma_{r^*}(M)/2k^2 n \ge \tau^{k-1}/2k^2 n \ge \tau^{k+1} = \epsilon_{\text{matrix}}.$$

Now Proposition 9 tells us that the $\hat{\tilde{\mu}}^i_j$ produced satisfy

$$|\hat{\tilde{\mu}}^i_j - \tilde{\mu}^i_j| \le O(k^{3/2}) \frac{\epsilon_{\text{matrix}} + \max\{\epsilon_{\text{matrix}}, \sigma_{r^*+1}(M)\}}{\sigma_k(M'_{\mathcal{J}})} \le O(k^{5/2}n) \frac{\epsilon_{\text{matrix}} + \sigma_{r^*+1}(M)}{\sigma_{r^*}(M)},$$

where in the last step we used (5). But by (3) we have $\epsilon_{\text{matrix}}/\sigma_{r^*}(M) \le \epsilon_{\text{matrix}}/\tau^{k-1} = \tau^2$ and also $\sigma_{r^*+1}(M)/\sigma_{r^*}(M) \le \tau$. Thus we have

$$|\hat{\tilde{\mu}}^i_j - \tilde{\mu}^i_j| \le O(k^{5/2}n)\tau \le \tilde{\epsilon},$$

as desired.

It remains to bound the error blowup in step 8. By this point we have values for the π^i's that are accurate to within $\pm\epsilon_{\text{wts}}$, and further, all π^i's are at least ϵ. We also have values for all $\tilde{\mu}^i_j$'s that are accurate to within $\pm\tilde{\epsilon}$. Since the function $g(x,y) = y/\sqrt{x}$ satisfies

$$\sup_{\substack{x\in[\epsilon,1]\\ y\in[-1,1]}} \left|\frac{\partial}{\partial x} g(x,y)\right| = \frac{1}{2}\epsilon^{-3/2} \quad \text{and} \quad \sup_{\substack{x\in[\epsilon,1]\\ y\in[-1,1]}} \left|\frac{\partial}{\partial y} g(x,y)\right| = \epsilon^{-1/2},$$

the mean value theorem implies that in step 8 our resulting estimates $\hat{\mu}^i_j$ are accurate to within additive error

$$\epsilon_{\text{wts}} \cdot \frac{1}{2}\epsilon^{-3/2} + \tilde{\epsilon} \cdot \epsilon^{-1/2} \le \epsilon,$$

as necessary.

This completes the proof of WAM's correctness. As for the running time, it is easy to see that the dominating factor comes from gridding over the entries of M_J and $u_{r^*+1}, \ldots, u_k$. Since there are k^2 entries and we grid to granularity $\epsilon_{\text{matrix}} = \tau^{k+1} = \text{poly}(n/\epsilon)^k$, the overall running time is $\text{poly}(n/\epsilon)^{k^3}$; i.e., $\text{poly}(n/\epsilon)$ for constant k.

5. Estimating higher moments.

In this section we explain our remarks from section 2.3 more thoroughly; specifically, how to use WAM to learn a mixture $\mathbf{Z}$ of k product distributions $\mathbf{X}^1, \ldots, \mathbf{X}^k$ over $\{0, \ldots, b-1\}^n$. Such a distribution can be "parametrically" described by mixing weights $\{\pi^i\}_{i\in[k]}$ and probabilities $\{p^i_{j,\ell}\}$, where $p^i_{j,\ell} = \Pr[\mathbf{X}^i_j = \ell]$.

Running WAM on samples from $\mathbf{Z}$ gives a list of estimates of mixing weights and coordinate means $\mathbf{E}[\mathbf{X}^i_j]$, but these coordinate means are insufficient to completely describe the distributions $\mathbf{X}^i_j$. However, suppose that we run WAM on samples from $\mathbf{Z}^\ell$ (i.e., each time we obtain a draw $(z_1, \ldots, z_n)$ from $\mathbf{Z}$, we actually give $(z^\ell_1, \ldots, z^\ell_n)$ to WAM). It is easy to see that, by doing this, we are running WAM on the π-weighted mixture of distributions $(\mathbf{X}^1)^\ell, \ldots, (\mathbf{X}^k)^\ell$; we will thus get as output a list of candidates for the mixing weights and the *coordinate ℓth moments* $\mathbf{E}[(\mathbf{X}^i_j)^\ell]$ for $\mathbf{Z}$.

Our algorithm for distributions over $\{0, \ldots, b-1\}^n$ uses this approach to obtain a list of candidate descriptions of each of the first $b-1$ coordinate moments of $\mathbf{Z}$. The algorithm then essentially takes the cross-product of these $b-1$ lists to obtain a list of overall candidates, each of which is an estimate of the mixing weights and all $b-1$ moments. Since WAM guarantees that each list contains an accurate estimate, the overall list will also contain an accurate estimate of the mixing weights and of all moments. For each candidate the estimate of the moments is then easily converted to "parametric form" $\{p^i_{j,\ell}\}$, and as we show, any candidate with accurate estimates of the moments yields an accurate estimate of the probabilities $p^i_{j,\ell}$.

We now give the main theorem of the section, the proof of which contains the details of the algorithm.

THEOREM 10. *Fix $k = O(1)$, $b = O(1)$. Let $\mathbf{Z}$ be a mixture of k product distributions $\mathbf{X}^1, \dots, \mathbf{X}^k$ over $\{0, \dots, b-1\}^n$, so $\mathbf{Z}$ is described by mixing weights $\pi^1, \dots, \pi^k$ and probabilities $\{p^i_{j,\ell}\}_{i\in[k],\, j\in[n],\, \ell\in\{0,\dots,b-1\}}$.*

There is an algorithm with the following property: for any $\epsilon, \delta > 0$, the algorithm runs in poly$(n/\epsilon) \cdot \log \frac{1}{\delta}$ time, and with probability $1 - \delta$ outputs a list of candidates $\langle\{\hat{\pi}^i\}, \{\hat{p}^i_{j,\ell}\}\rangle$ such that for at least one candidate in the list, the following hold:

1. *$|\hat{\pi}^i - \pi^i| \le \epsilon$ for all $i \in [k]$; and*
2. *$|\hat{p}^i_{j,\ell} - p^i_{j,\ell}| \le \epsilon$ for all i, j, ℓ such that $\pi^i \ge \epsilon$.*

Proof. For each $\ell = 1, \dots, b-1$, the algorithm runs WAM on the random variable $\mathbf{Z}^\ell$. In each such run, the "ϵ" parameter of WAM is set to $\epsilon' := \epsilon\sigma_b/(O(b^{3/2}) \cdot (b-1)^{b-1})$, where σ_b is a constant we define later, and the "δ" parameter is set to $\delta' := \delta/(b-1)$. From these runs we obtain $(b-1)$ lists $L_1, \dots, L_{b-1}$ of candidates $\langle\{\hat{\pi}^i\}, \{\hat{\mu}^i_{j,\ell}\}_{i,j}\rangle$, where $\hat{\mu}^i_{j,\ell}$ is an estimate of $\mu^i_{j,\ell} = \mathbf{E}[(X^i_j)^\ell]$. The algorithm then uses these $(b-1)$ lists to construct one larger list L of candidates $\langle\{\hat{\pi}^i\}, \{\mu^i_{j,\ell}\}_{i,j,\ell}\rangle$, where each candidate estimates the mixing weights and all $b-1$ moments. This is done by taking all possible combinations of one candidate from each of the $b-1$ lists $L_1, \dots, L_{b-1}$ and combining them as follows: take the mixing weights $\{\hat{\pi}^i\}$ from the candidate from list L_1, and for $\ell = 1, \dots, b-1$, take $\{\mu^i_{j,\ell}\}_{i,j}$ from the candidate from list L_ℓ. The list L will have size $|L| = \prod_{\ell=1}^{b-1} |L_\ell| = \text{poly}(n, 1/\epsilon)$.

Theorem 4 on the WAM algorithm guarantees that with probability at least $1 - (b-1)\delta' = 1 - \delta$, each list L_ℓ contains a candidate whose $\{\hat{\mu}^i_{j,\ell}\}$ are accurate estimates of the ℓth moments. When we choose the accurate candidate from each list, we will obtain an overall candidate in L that is accurate on *all $b-1$ moments.* Define $\epsilon'' := \epsilon'(b-1)^{b-1}/2 = \epsilon\sigma_b/O(b^{3/2})$. Formally, the list L will contain a candidate $\langle\{\hat{\pi}^i\}, \{\hat{\mu}^i_{j,\ell}\}_{i,j,\ell}\rangle$ such that (i) $|\hat{\pi}^i - \pi^i| \le \epsilon''$ for all $i \in [k]$, and (ii) $|\hat{\mu}^i_{j,\ell} - \mu^i_{j,\ell}| \le \epsilon''$ for all i, j, ℓ such that $\pi^i \ge \epsilon''$. (The extra factor of $(b-1)^{b-1}/2$ comes from the need to scale the distributions for WAM so that the means fall into the range $[-1, 1]$.)

To complete the proof of the theorem, we must show how the algorithm converts each candidate $\langle\{\hat{\pi}^i\}, \{\hat{\mu}^i_{j,\ell}\}\rangle$ in the list L into "parametric" form $\langle\{\hat{\pi}^i\}, \{\hat{p}^i_{j,\ell}\}\rangle$ so that the "good" candidate satisfying (i) and (ii) above does not incur much error. It is easy to see that for a given $i \in [k]$, $j \in [n]$, we have $(\mu^i_{j,0}, \dots, \mu^i_{j,b-1}) = (p^i_{j,0}, \dots, p^i_{j,b-1})V$, where V is a $b \times b$ Vandermonde matrix (more precisely, $V_{\alpha,\beta} = (\alpha-1)^{\beta-1}$, with $V_{1,1} = 1$). Following this characterization, the algorithm computes $(\hat{p}^i_{j,0}, \dots, \hat{p}^i_{j,b-1}) = (\hat{\mu}^i_{j,0}, \dots, \hat{\mu}^i_{j,b-1})V^{-1}$ for each i, j to obtain parametric estimates $\{\hat{p}^i_{j,\ell}\}$ for the probabilities $\{p^i_{j,\ell}\}$.

Now applying Proposition 9, we have that for all i, j, ℓ we have $|\hat{p}^i_{j,\ell} - p^i_{j,\ell}| \le \epsilon'' \cdot O(b^{3/2})/\sigma_b = \epsilon$, where σ_b is set equal to $\sigma_b(V)$, the smallest singular value of V. (Since the Vandermonde matrix is nonsingular, even without specifying σ_b we have that it is a positive constant that depends only on b; it can be shown to be at least $b^{-\text{poly}(b)}$.) The running time is dominated by the time to take the cross-product of the lists. This concludes the proof of Theorem 10. □

We remark that the running time dependence on b is of the form $(n/\epsilon)^{\text{poly}(b)}$; since a b in the exponent is inevitable in our cross-product approach, we have refrained from excessive optimization of the dependence on b (by doing things such as representing the alphabet by bth roots of unity rather than equally spaced reals, which would have given a better Vandermonde singular value bound).

6. The road ahead. Since the binary domain $\{0,1\}^n$ corresponds to the $b=2$ case of the general $\{0,\dots,b-1\}^n$ domain, here we shall deal only with the latter.

Recall that $p^i_{j,\ell}$ is the probability that under the ith product distribution over $\{0,\dots,b-1\}^n$ in the target mixture $\mathbf{Z}$, the jth coordinate takes value ℓ. From Theorem 10, we have a list L of M *candidates* $\langle\{\hat{\pi}^i\},\{\hat{p}^i_{j,\ell}\}\rangle$ such that at least one candidate is *parametrically accurate*—i.e., satisfies the following:

1. $|\hat{\pi}^i-\pi^i|\le\epsilon$ for all $i=1\dots k$; and
2. $|\hat{p}^i_{j,\ell}-p^i_{j,\ell}|\le\epsilon$ for all $i\in[k]$, $j\in[n]$ and $\ell\in\{0,\dots,b-1\}$ such that $\pi^i\ge\epsilon$.

In section 7, we show how to convert a candidate into a true mixture of product distributions, in such a way that any parametrically accurate candidate becomes a mixture distribution with small KL divergence from the target distribution (see Theorem 11). Applying this conversion procedure to the list from Theorem 10, we get a list of M hypothesis mixture distributions such that at least one hypothesis in the list has small KL divergence from the target $\mathbf{Z}$ (see Theorem 15).

Then in section 8 we show how a maximum likelihood procedure can find a KL-accurate hypothesis (one with small KL divergence from $\mathbf{Z}$) from among a list of hypotheses, one of which is guaranteed to have good KL divergence (see Theorem 16).

In section 9 we combine Theorem 16 with Theorem 15 to obtain Theorem 2.

7. From candidates to hypothesis mixture distributions. The following theorem defines a process that converts a single candidate for the π^i's and $p^i_{j,\ell}$'s of $\mathbf{Z}$ to a true mixture of product distributions over $\{0,\dots,b-1\}^n$ that has at least some minimum mass on every point in $\{0,\dots,b-1\}^n$. (As we will see in section 8, this minimum mass condition is required by the maximum likelihood procedure.) More importantly, the theorem guarantees that if the candidate is parametrically accurate, then the process outputs a mixture distribution with small KL divergence relative to $\mathbf{Z}$.

THEOREM 11.

1. *There is an efficient procedure $\mathcal{A}$ which takes values $\epsilon_{\text{bprobs}},\epsilon_{\text{wts}}>0$ and $\hat{\pi}^i,\hat{p}^i_{j,\ell}$ as inputs and outputs a mixture $\dot{\mathbf{Z}}$ of k product distributions over $\{0,\dots,b-1\}^n$ with mixing weights $\dot{\pi}^i>0$ and probabilities $\dot{p}^i_{j,\ell}>0$ satisfying*
   (a) $\sum_{i=1}^k\dot{\pi}^i=1$, *and for each $i\in[k]$ and $j\in[n]$*, $\sum_{\ell=0}^{b-1}p^i_{j,\ell}=1$;
   (b) $\dot{\mathbf{Z}}(x)\ge(\epsilon_{\text{bprobs}})^n$ *for all* $x\in\{0,\dots,b-1\}^n$.
2. *Furthermore, suppose that $\mathbf{Z}$ is a mixture of k product distributions on $\{0,\dots,b-1\}^n$ with mixing weights $\pi^1,\dots,\pi^k$ and probabilities $p^i_{j,\ell}$, and that the following are satisfied:*
   (a) *for $i=1,\dots,k$ we have $|\pi^i-\hat{\pi}^i|\le\epsilon_{\text{wts}}$, and*
   (b) *for all i,j,ℓ such that $\pi^i\ge\epsilon_{\text{minwt}}$ we have $|p^i_{j,\ell}-\hat{p}^i_{j,\ell}|\le\epsilon_{\text{bprobs}}$.*

   Then for sufficiently small ϵ_{bprobs} and ϵ_{wts} the mixture $\dot{\mathbf{Z}}$ will satisfy

$$\mathrm{KL}(\mathbf{Z}||\dot{\mathbf{Z}})\le\eta(\epsilon_{\text{bprobs}},\epsilon_{\text{wts}},\epsilon_{\text{minwt}}),\tag{6}$$

where

$$\eta(\epsilon_{\text{bprobs}},\epsilon_{\text{wts}},\epsilon_{\text{minwt}}):=n\cdot(12b^3\epsilon^{1/2}_{\text{bprobs}})+k\epsilon_{\text{minwt}}n\ln(b/\epsilon_{\text{bprobs}})+\epsilon^{1/3}_{\text{wts}}.$$

We prove Theorem 11 in section 7.2 after setting up the required machinery in section 7.1.

7.1. Some tools. Here we give some propositions which will be used in the proof of Theorem 11.

The following simple proposition bounds the KL divergence between two product distributions in terms of the KL divergences between their coordinates.

Proposition 12. *Suppose that* $\mathbf{P}_1, \ldots, \mathbf{P}_n$ *and* $\mathbf{Q}_1, \ldots, \mathbf{Q}_n$ *are distributions satisfying* $\mathrm{KL}(\mathbf{P}_i || \mathbf{Q}_i) \le \epsilon_i$ *for all* i*. Then* $\mathrm{KL}(\mathbf{P}_1 \times \cdots \times \mathbf{P}_n || \mathbf{Q}_1 \times \cdots \times \mathbf{Q}_n) \le \sum_{i=1}^n \epsilon_i$.

Proof. We prove the case $n = 2$:

$$\begin{aligned}
\mathrm{KL}(\mathbf{P}_1 \times \mathbf{P}_2 || \mathbf{Q}_1 \times \mathbf{Q}_2) &= \sum_x \sum_y \mathbf{P}_1(x)\mathbf{P}_2(y) \ln \frac{\mathbf{P}_1(x)\mathbf{P}_2(y)}{\mathbf{Q}_1(x)\mathbf{Q}_2(y)} \\
&= \sum_x \sum_y \mathbf{P}_1(x)\mathbf{P}_2(y) \ln \frac{\mathbf{P}_1(x)}{\mathbf{Q}_1(x)} + \sum_x \sum_y \mathbf{P}_1(x)\mathbf{P}_2(y) \ln \frac{\mathbf{P}_2(y)}{\mathbf{Q}_2(y)} \\
&= \sum_y \mathbf{P}_2(y)\, \mathrm{KL}(\mathbf{P}_1 || \mathbf{Q}_1) + \sum_x \mathbf{P}_1(x)\, \mathrm{KL}(\mathbf{P}_2 || \mathbf{Q}_2) \\
&\le \epsilon_1 + \epsilon_2.
\end{aligned}$$

The general case follows by induction. □

Very roughly speaking, the following proposition states that if $\mathbf{P}$ is a π-weighted mixture of distributions $\mathbf{P}^1, \ldots, \mathbf{P}^k$ and $\mathbf{Q}$ is a γ-weighted mixture of distributions $\mathbf{Q}^1, \ldots, \mathbf{Q}^k$, then if each $\mathbf{Q}^i$ is "close" to the corresponding $\mathbf{P}^i$ and the π-weighting is "close" to the γ-weighting, then $\mathbf{Q}$ is "close" to $\mathbf{P}$. To make this precise we need several technical conditions as stated in the proposition.

Proposition 13. *Let* $\pi^1, \ldots, \pi^k$, $\gamma^1, \ldots, \gamma^k \ge 0$ *be mixing weights satisfying* $\sum \pi^i = \sum \gamma^i = 1$*. Let* ϵ_1, ϵ_2, ϵ_3, $\epsilon_{\mathcal{I}}$, ϵ_{all} *be positive constants. Let* $\mathcal{I} = \{i : \pi^i \ge \epsilon_3\}$*. Let* $\mathbf{P}^1, \ldots, \mathbf{P}^k$ *and* $\mathbf{Q}^1, \ldots, \mathbf{Q}^k$ *be distributions. Suppose that*

1. $|\pi^i - \gamma^i| \le \epsilon_1$ *for all* $i \in [k]$*;*
2. $\gamma^i \ge \epsilon_2$ *for all* $i \in [k]$*;*
3. $\mathrm{KL}(\mathbf{P}^i || \mathbf{Q}^i) \le \epsilon_{\mathcal{I}}$ *for all* $i \in \mathcal{I}$*;*
4. $\mathrm{KL}(\mathbf{P}^i || \mathbf{Q}^i) \le \epsilon_{\mathrm{all}}$ *for all* $i \in [k]$*.*

Then, letting $\mathbf{P}$ *denote the* π*-mixture of the* $\mathbf{P}^i$*'s and* $\mathbf{Q}$ *the* γ*-mixture of the* $\mathbf{Q}^i$*'s, for any* $\epsilon_4 > \epsilon_1$ *we have*

$$\mathrm{KL}(\mathbf{P} || \mathbf{Q}) \le \epsilon_{\mathcal{I}} + k\epsilon_3\epsilon_{\mathrm{all}} + k\epsilon_4 \ln \frac{\epsilon_4}{\epsilon_2} + \frac{\epsilon_1}{\epsilon_4 - \epsilon_1}.$$

Proof.

$$\begin{aligned}
\mathrm{KL}(\mathbf{P} || \mathbf{Q}) &= \sum_x \left(\sum_i \pi^i \mathbf{P}^i(x) \right) \ln \frac{\sum_i \pi^i \mathbf{P}^i(x)}{\sum_i \gamma^i \mathbf{Q}^i(x)} \\
&\le \sum_x \sum_i \pi^i \mathbf{P}^i(x) \ln \frac{\pi^i \mathbf{P}^i(x)}{\gamma^i \mathbf{Q}^i(x)} \qquad \text{(by the log-sum inequality [7])} \\
&= \sum_i \pi^i \sum_x \left(\mathbf{P}^i(x) \ln \frac{\mathbf{P}^i(x)}{\mathbf{Q}^i(x)} + \mathbf{P}^i(x) \ln \frac{\pi^i}{\gamma^i} \right) \\
&= \sum_i \pi^i\, \mathrm{KL}(\mathbf{P}^i || \mathbf{Q}^i) + \sum_i \pi^i \ln \frac{\pi^i}{\gamma^i} \\
&= \left(\sum_{i \in \mathcal{I}} \pi^i\, \mathrm{KL}(\mathbf{P}^i || \mathbf{Q}^i) \right) + \left(\sum_{i \notin \mathcal{I}} \pi^i\, \mathrm{KL}(\mathbf{P}^i || \mathbf{Q}^i) \right) + \sum_i \pi^i \ln \frac{\pi^i}{\gamma^i}. \qquad (7)
\end{aligned}$$

For the first term of (7), we have

$$\sum_{i\in\mathcal{I}} \pi^i \, \mathrm{KL}(\mathbf{P}^i || \mathbf{Q}^i) \le \epsilon_{\mathcal{I}}.$$

For the second term of (7), we have

$$\sum_{i\notin\mathcal{I}} \pi^i \, \mathrm{KL}(\mathbf{P}^i || \mathbf{Q}^i) \le k\epsilon_3 \cdot \max_{i\in[k]} \{\mathrm{KL}(\mathbf{P}^i || \mathbf{Q}^i)\} \le k\epsilon_3\epsilon_{\mathrm{all}}.$$

For the third term of (7), letting $\mathcal{I}' = \{i \in \mathcal{I} : \pi^i \ge \epsilon_4\}$, we have

$$\sum_i \pi^i \ln \frac{\pi^i}{\gamma^i} = \sum_{i\notin\mathcal{I}'} \pi^i \ln \frac{\pi^i}{\gamma^i} + \sum_{i\in\mathcal{I}'} \pi^i \ln \frac{\pi^i}{\gamma^i}. \tag{8}$$

For the first sum in (8) we have

$$\sum_{i\notin\mathcal{I}'} \pi^i \ln \frac{\pi^i}{\gamma^i} \le k\epsilon_4 \ln \frac{\epsilon_4}{\epsilon_2}.$$

For the second sum in (8), note first that for any i such that $\pi^i < \gamma^i$, the contribution to the second sum is negative. For any other i, we have $\pi^i \ge \gamma^i$ and therefore $\gamma^i \ge \pi^i - \epsilon_1$. Consequently we have $\frac{\pi^i}{\gamma^i} \le \frac{\pi^i}{\pi^i - \epsilon_1} = 1 + \frac{\epsilon_1}{\pi^i - \epsilon_1} \le 1 + \frac{\epsilon_1}{\epsilon_4 - \epsilon_1}$. Hence the second sum in (8) is at most

$$\sum_{i\in\mathcal{I}'} \pi^i \ln \frac{\pi^i}{\gamma^i} \le \sum_{i\in\mathcal{I}'} \pi^i \ln \left(1 + \frac{\epsilon_1}{\epsilon_4 - \epsilon_1}\right) \le \frac{\epsilon_1}{\epsilon_4 - \epsilon_1}.$$

Putting all the bounds together, the proof is done. □

Finally, we will also need the following elementary proposition.

PROPOSITION 14. *Let* $\mathbf{P}$ *and* $\mathbf{Q}$ *denote distributions over* $\{0, \dots, b-1\}$*, where* $\mathbf{P}$ *has probabilities* $p_0, \dots, p_{b-1}$ *and* $\mathbf{Q}$ *has probabilities* $q_0, \dots, q_{b-1}$*. Suppose that* $|p_\ell - q_\ell| < \xi \le \frac{1}{4}$ *for all* $\ell \in \{0, \dots, b-1\}$*, and that also* $q_\ell \ge \tau$ *for all* $\ell \in \{0, \dots, b-1\}$*, where* $\tau < \xi$*. Then* $\mathrm{KL}(\mathbf{P}||\mathbf{Q}) \le 2\xi^{1/2} + b\xi^{3/2}/\tau$.

Proof. Let $L_{small} = \{\ell \in \{0, \dots, b-1\} : p_\ell \le \xi^{1/2}\}$ and $L_{big} = \{0, \dots, b-1\} \setminus L_{small}$. We bound the contribution to $\mathrm{KL}(\mathbf{P}||\mathbf{Q})$ from L_{small} and L_{big} separately.

Now for the L_{small} case. For all ℓ, it is easy to see that $\ln \frac{p_\ell}{q_\ell} \le \ln \frac{\xi+\tau}{\tau} = \ln(1+\frac{\xi}{\tau}) \le \frac{\xi}{\tau}$. Thus each $\ell \in L_{small}$ contributes at most $p_\ell \ln \frac{p_\ell}{q_\ell} \le \frac{\xi^{3/2}}{\tau}$. Since $|L_{small}| \le b$, the total contribution to $\mathrm{KL}(\mathbf{P}||\mathbf{Q})$ from L_{small} is at most $b\frac{\xi^{3/2}}{\tau}$.

If $\ell \in L_{big}$, then we have

$$\frac{p_\ell}{q_\ell} \le \frac{p_\ell}{p_\ell - \xi} = 1 + \frac{\xi}{p_\ell - \xi} \le 1 + \frac{\xi}{\xi^{1/2} - \xi} \le 1 + 2\xi^{1/2},$$

where the last inequality holds since $\xi^{1/2} \le \xi^{1/2}/2$ (since $\xi \le \frac{1}{4}$). We thus have that the total contribution to $\mathrm{KL}(\mathbf{P}||\mathbf{Q})$ from $\ell \in L_{big}$ is at most $\ln(1 + 2\xi^{1/2}) \le 2\xi^{1/2}$. This proves the proposition. □

7.2. Proof of Theorem 11. We construct a mixture $\dot{\mathbf{Z}}$ of product distributions $\dot{\mathbf{Z}}^1, \dots, \dot{\mathbf{Z}}^k$ by defining new mixing weights $\dot\pi^i$ and probabilities $\dot p^i_{j,\ell}$. The procedure $\mathcal{A}$ is defined as follows:

1. For all $i = 1, \dots, k$ let

$$\ddot\pi^i = \begin{cases} \hat\pi^i & \text{if } \hat\pi^i \geq \epsilon_{\text{wts}}, \\ \epsilon_{\text{wts}} & \text{if } \hat\pi^i < \epsilon_{\text{wts}}. \end{cases}$$

Now let s be such that $s\sum_{i=1}^k \ddot\pi^i = 1$, and take $\dot\pi^i = s\ddot\pi^i$.

2. For all $i \in [k]$ and $j \in [n]$, let

$$\ddot p^i_{j,\ell} = \begin{cases} \hat p^i_{j,\ell} & \text{if } \hat p^i_{j,\ell} \geq \epsilon_{\text{bprobs}}, \\ \epsilon_{\text{bprobs}} & \text{if } \hat p^i_{j,\ell} < \epsilon_{\text{bprobs}}. \end{cases}$$

Now let t be such that $t\sum_{\ell\in\{0,\dots,b-1\}} \ddot p^i_{j,\ell} = 1$, and take $\dot p^i_{j,\ell} = t\ddot p^i_{j,\ell}$.

It is clear from construction that this yields $\dot\pi^i, \dot p^i_{j,\ell}$ that satisfy condition 1(a) of the theorem. It is also clear that for each $i \in [k]$ we have that the distribution $\dot{\mathbf{Z}}^i$ satisfies $\dot{\mathbf{Z}}^i(x) \geq \epsilon^n_{\text{bprobs}}$ for all $x \in \{0, \dots, b-1\}^n$, and thus the mixture $\dot{\mathbf{Z}}$ must satisfy $\dot{\mathbf{Z}}(x) \geq \epsilon^n_{\text{bprobs}}$ for all these x. This gives part 1(b) of the theorem.

We now turn to part 2, and henceforth assume that the conditions on $\pi^i, \hat\pi^i, p^i_{j,\ell}$, $\hat p^i_{j,\ell}$ from part 2 are indeed all satisfied. Roughly speaking, these conditions tell us that $\hat\pi^i, \hat p^i_{j,\ell}$ are "good" (in the sense that they are parametrically accurate); we will show that the resulting $\dot\pi^i, \dot p^i_{j,\ell}$ are "good" (in the sense of giving rise to a mixture $\dot{\mathbf{Z}}$ that satisfies (6)).

Our goal is to apply Proposition 13 with parameter settings

$$\epsilon_1 = 3k\epsilon_{\text{wts}}, \quad \epsilon_2 = \frac{\epsilon_{\text{wts}}}{2}, \quad \epsilon_3 = \epsilon_{\text{minwt}},$$
$$\epsilon_4 = \epsilon^{1/2}_{\text{wts}}, \quad \epsilon_{\mathcal{I}} = 12nb^3\epsilon^{1/2}_{\text{bprobs}}, \quad \epsilon_{\text{all}} = n\ln(b/\epsilon_{\text{bprobs}}) \tag{9}$$

to bound $\text{KL}(\mathbf{Z}||\dot{\mathbf{Z}})$. To satisfy the conditions of Proposition 13 we must (1) upper bound $|\pi^i - \dot\pi^i|$ for all i, (2) lower bound $\dot\pi^i$ for all i, (3) upper bound $\text{KL}(\mathbf{Z}^i||\dot{\mathbf{Z}}^i)$ for all i such that $\pi^i \geq \epsilon_{\text{minwt}}$, and (4) upper bound $\text{KL}(\mathbf{Z}^i||\dot{\mathbf{Z}}^i)$ for all $i \in [k]$. We now do this.

(1) *Upper bounding* $|\pi^i - \dot\pi^i|$. Fix any $i \in [k]$. If $\hat\pi^i \geq \epsilon_{\text{wts}}$, then we have $\ddot\pi^i = \hat\pi^i$, so $|\pi^i - \ddot\pi^i| \leq \epsilon_{\text{wts}}$. On the other hand, if $\hat\pi^i < \epsilon_{\text{wts}}$, then it must be the case that $\pi^i \leq 2\epsilon_{\text{wts}}$, so we again have $|\pi^i - \ddot\pi^i| \leq \epsilon_{\text{wts}}$. Since $\sum_{i=1}^k \pi^i = 1$ it follows that

$$\left|\sum_{i=1}^k \ddot\pi^i - 1\right| \leq k\epsilon_{\text{wts}} \tag{10}$$

and thus

$$\sum_{i=1}^k \ddot\pi^i \in [1 - k\epsilon_{\text{wts}}, 1 + k\epsilon_{\text{wts}}].$$

By definition of s this gives

$$s \in \left[\frac{1}{1+k\epsilon_{\text{wts}}}, \frac{1}{1-k\epsilon_{\text{wts}}}\right]. \tag{11}$$

Multiplying inequality (10) by s, recalling that $s\sum_{i=1}^{k}\ddot{\pi}^i = 1$, and assuming $\epsilon_{\mathrm{wts}} \le 1/(2k)$, we obtain

$$|1-s| \le sk\epsilon_{\mathrm{wts}} \le \frac{k\epsilon_{\mathrm{wts}}}{1-k\epsilon_{\mathrm{wts}}} \le 2k\epsilon_{\mathrm{wts}}.$$

Thus, we have

$$\begin{aligned} |\pi^i - \dot{\pi}^i| &\le |\pi^i - \ddot{\pi}^i| + |\ddot{\pi}^i - \dot{\pi}^i| \\ &\le \epsilon_{\mathrm{wts}} + |\ddot{\pi}^i - \dot{\pi}^i| \\ &= \epsilon_{\mathrm{wts}} + |(1-s)\ddot{\pi}^i| \\ &\le \epsilon_{\mathrm{wts}} + 2k\epsilon_{\mathrm{wts}}|\ddot{\pi}^i| \\ &\le \epsilon_{\mathrm{wts}} + 2k\epsilon_{\mathrm{wts}}; \end{aligned}$$

certainly, this gives $|\pi^i - \dot{\pi}^i| \le 3k\epsilon_{\mathrm{wts}}$.

(2) *Lower bounding* $\dot{\pi}^i$. To lower bound $\dot{\pi}^i$, we note that since $\ddot{\pi}^i \ge \epsilon_{\mathrm{wts}}$ for all i, under the assumption $\epsilon_{\mathrm{wts}} \le 1/(2k)$, we have

$$\dot{\pi}^i = s\ddot{\pi}^i \ge \frac{1}{1+k\epsilon_{\mathrm{wts}}}\ddot{\pi}^i \ge \frac{\epsilon_{\mathrm{wts}}}{1+k\epsilon_{\mathrm{wts}}} \ge \frac{2\epsilon_{\mathrm{wts}}}{3},$$

where the first inequality follows from (11).

(3) *Upper bounding* $\mathrm{KL}(\mathbf{Z}^i||\dot{\mathbf{Z}}^i)$ *for all* i *such that* $\pi^i \ge \epsilon_{\mathrm{minwt}}$. Fix an i such that $\pi^i \ge \epsilon_{\mathrm{minwt}}$, and fix any $j \in [n]$. Let $\mathbf{P}$ denote the distribution over $\{0,\dots,b-1\}$ with probabilities $p^i_{j,0},\dots,p^i_{j,b-1}$, and let $\mathbf{Q}$ denote the distribution over $\{0,\dots,b-1\}$ with probabilities $\dot{p}^i_{j,0},\dots,\dot{p}^i_{j,b-1}$.

We first show that each $\ddot{p}^i_{j,\ell}$ is close to $\hat{p}^i_{j,\ell}$ and thus also to $p^i_{j,\ell}$. This is done very much as in (1) above. If $\hat{p}^i_{j,\ell} \ge \epsilon_{\mathrm{bprobs}}$, then we have $\ddot{p}^i_{j,\ell} = \hat{p}^i_{j,\ell}$, and so $|p^i_{j,\ell} - \ddot{p}^i_{j,\ell}| \le \epsilon_{\mathrm{bprobs}}$ (by condition 2(b) in the theorem statement). On the other hand, if $\hat{p}^i_{j,\ell} < \epsilon_{\mathrm{bprobs}}$, then it must be the case that $p^i_{j,\ell} \le 2\epsilon_{\mathrm{bprobs}}$, so we again have $|p^i_{j,\ell} - \ddot{p}^i_{j,\ell}| \le \epsilon_{\mathrm{bprobs}}$. Since $\sum_{\ell=0}^{b-1} p^i_{j,\ell} = 1$ it follows that

$$\left|\sum_{\ell=0}^{b-1}\ddot{p}^i_{j,\ell} - 1\right| \le b\epsilon_{\mathrm{bprobs}} \tag{12}$$

and thus

$$\sum_{\ell=0}^{b-1}\ddot{p}^i_{j,\ell} \in [1-b\epsilon_{\mathrm{bprobs}}, 1+b\epsilon_{\mathrm{bprobs}}].$$

By definition of t this gives

$$t \in \left[\frac{1}{1+b\epsilon_{\mathrm{bprobs}}}, \frac{1}{1-b\epsilon_{\mathrm{bprobs}}}\right]. \tag{13}$$

Multiplying inequality (12) by t, recalling that $t\sum_{\ell=0}^{b-1}\ddot{p}^i_{j,\ell} = 1$, and assuming $\epsilon_{\mathrm{bprobs}} \le 1/(2b)$, we obtain

$$|1-t| \le tb\epsilon_{\mathrm{bprobs}} \le \frac{b\epsilon_{\mathrm{bprobs}}}{1-b\epsilon_{\mathrm{bprobs}}} \le 2b\epsilon_{\mathrm{bprobs}}.$$

Thus, we have

$$\begin{aligned}|p^i_{j,\ell} - \dot{p}^i_{j,\ell}| &\leq |p^i_{j,\ell} - \ddot{p}^i_{j,\ell}| + |\ddot{p}^i_{j,\ell} - \dot{p}^i_{j,\ell}|\\ &\leq \epsilon_{\text{bprobs}} + |\ddot{p}^i_{j,\ell} - \dot{p}^i_{j,\ell}|\\ &= \epsilon_{\text{bprobs}} + |(1-t)\ddot{p}^i_{j,\ell}|\\ &\leq \epsilon_{\text{bprobs}} + 2b\epsilon_{\text{bprobs}}|\ddot{p}^i_{j,\ell}|\\ &\leq \epsilon_{\text{bprobs}} + 2b\epsilon_{\text{bprobs}};\end{aligned}$$

certainly, this gives $|p^i_{j,\ell} - \dot{p}^i_{j,\ell}| \leq 3b\epsilon_{\text{bprobs}}$.

Moreover, since $\ddot{p}^i_{j,\ell} \geq \epsilon_{\text{bprobs}}$ for all ℓ and $\dot{p}^i_{j,\ell} = t\ddot{p}^i_{j,\ell}$, where $t > \frac{1}{2}$ (by (13) and $\epsilon_{\text{bprobs}} \leq 1/b$), we also have $\dot{p}^i_{j,\ell} \geq \epsilon_{\text{bprobs}}/2$. We may thus apply Proposition 14 to $\mathbf{P}$ and $\mathbf{Q}$ (taking $\tau = \epsilon_{\text{bprobs}}/2$ and $\xi = 3b\epsilon_{\text{bprobs}}$), and we obtain $\mathrm{KL}(\mathbf{P}||\mathbf{Q}) \leq 2(3b\epsilon_{\text{bprobs}})^{1/2} + b(3b\epsilon_{\text{bprobs}})^{3/2}/(\epsilon_{\text{bprobs}}/2)$. A rough estimation gives that this is at most $12b^3\epsilon^{1/2}_{\text{bprobs}}$. Each $\mathbf{Z}^i$ ($\dot{\mathbf{Z}}^i$, respectively) is the product of n such distributions $\mathbf{P}$ (distributions $\mathbf{Q}$, respectively) over $\{0,\ldots,b-1\}$. Therefore, by Proposition 12, we have $\mathrm{KL}(\mathbf{Z}^i||\dot{\mathbf{Z}}^i) \leq n \cdot (12b^3\epsilon^{1/2}_{\text{bprobs}})$ for all i with $\pi^i \geq \epsilon_{\text{minwt}}$.

(4) *Upper bounding* $\mathrm{KL}(\mathbf{Z}^i||\dot{\mathbf{Z}}^i)$ *for all* $i \in [k]$. This is simple: fix any $i \in [k]$. Since we know that $\dot{\mathbf{Z}}^i(x) \geq \epsilon^n_{\text{bprobs}}$ for all $x \in \{0,\ldots,b-1\}^n$, we immediately have

$$\mathrm{KL}(\mathbf{Z}^i||\dot{\mathbf{Z}}^i) \leq -H(\mathbf{Z}^i) + \ln(1/(\epsilon_{\text{bprobs}})^n) \leq n\ln(b/\epsilon_{\text{bprobs}}),$$

where $H(\mathbf{X}) := \sum_x \mathbf{X}(x)\ln(1/\mathbf{X}(x))$ denotes the"entropy in nats" of the random variable $\mathbf{X}$.

We can now apply Proposition 13 with the parameter settings given by (9). Proposition 13 implies

$$\begin{aligned}\mathrm{KL}(\mathbf{Z}||\dot{\mathbf{Z}}) \leq n \cdot (12b^3\epsilon^{1/2}_{\text{bprobs}}) &+ k\epsilon_{\text{minwt}}n\ln(b/\epsilon_{\text{bprobs}})\\ &+ \left[k\epsilon^{1/2}_{\text{wts}}\ln\frac{\epsilon^{1/2}_{\text{wts}}}{\epsilon_{\text{wts}}/2} + \frac{3k\epsilon_{\text{wts}}}{\epsilon^{1/2}_{\text{wts}} - 3k\epsilon_{\text{wts}}}\right].\end{aligned}$$

Considering the terms of the expression in brackets above, if we set $\epsilon_{\text{wts}} = \frac{c}{k^7}$ for some appropriately chosen small constant c, then we have that

$$k\epsilon^{1/2}_{\text{wts}}\ln\frac{\epsilon^{1/2}_{\text{wts}}}{\epsilon_{\text{wts}}/2} = k\epsilon^{1/2}_{\text{wts}}\ln\frac{2}{\epsilon^{1/2}_{\text{wts}}} \leq \frac{1}{2}\epsilon^{1/3}_{\text{wts}}$$

and

$$\frac{3k\epsilon_{\text{wts}}}{\epsilon^{1/2}_{\text{wts}} - 3k\epsilon_{\text{wts}}} \leq 6k\epsilon^{1/2}_{\text{wts}} \leq \frac{1}{2}\epsilon^{1/3}_{\text{wts}}.$$

Hence

$$\mathrm{KL}(\mathbf{Z}||\dot{\mathbf{Z}}) \leq n \cdot (12b^3\epsilon^{1/2}_{\text{bprobs}}) + k\epsilon_{\text{minwt}}n\ln(b/\epsilon_{\text{bprobs}}) + \epsilon^{1/3}_{\text{wts}}.$$

This concludes the proof of Theorem 11. □

7.3. Some candidate distribution is good.

Here we establish the following.

THEOREM 15. *Let $b = O(1)$, and let $\mathbf{Z}$ be any unknown mixture of k product distributions over $\{0, \ldots, b-1\}^n$. There is a poly$(n/\epsilon)\cdot\log\frac{1}{\delta}$ time algorithm which, given samples from $\mathbf{Z}$, outputs a list of poly(n/ϵ) many mixtures of product distributions over $\{0, \ldots, b-1\}^n$ with the properties that*

1. *every distribution $\mathbf{Z}'$ in the list satisfies $(\frac{\epsilon}{36nb^3})^{2n} \le \mathbf{Z}'(x) \le 1$ for all $x \in \{0, \ldots, b-1\}^n$; and*
2. *with probability $1-\delta$, some distribution $\mathbf{Z}^\star$ in the list satisfies* $\mathrm{KL}(\mathbf{Z}||\mathbf{Z}^\star) \le \epsilon$.

Proof. We will use a specialization of Theorem 10 in which we have different parameters for the different roles that ϵ plays, as follows.

THEOREM 10′. *Fix $k = O(1)$, $b = O(1)$. Let $\mathbf{Z}$ be a mixture of k product distributions $\mathbf{X}^1, \ldots, \mathbf{X}^k$ over $\{0, \ldots, b-1\}^n$, so that $\mathbf{Z}$ is described by mixing weights $\pi^1, \ldots, \pi^k$ and probabilities $\{p^i_{j,\ell}\}_{i\in[k],\, j\in[n],\, \ell\in\{0,\ldots,b-1\}}$.*

There is an algorithm with the following property: for any $\epsilon_{\mathrm{wts}}, \epsilon_{\mathrm{bprobs}}, \epsilon_{\mathrm{minwt}}, \delta > 0$, with probability $1-\delta$ the algorithm outputs a list of candidates $\langle\{\hat{\pi}^i\}, \{\hat{p}^i_{j,\ell}\}\rangle$ such that for at least one candidate in the list, the following hold:

1. *$|\hat{\pi}^i - \pi^i| \le \epsilon_{\mathrm{wts}}$ for all $i \in [k]$; and*
2. *$|\hat{p}^i_{j,\ell} - p^i_{j,\ell}| \le \epsilon_{\mathrm{bprobs}}$ for all i, j, ℓ such that $\pi^i \ge \epsilon_{\mathrm{minwt}}$.*

The algorithm runs in time poly$(n/\epsilon')\cdot\log(1/\delta)$, where $\epsilon' = \min\{\epsilon_{\mathrm{wts}}, \epsilon_{\mathrm{bprobs}}, \epsilon_{\mathrm{minwt}}\}$.

Let $\epsilon, \delta > 0$ be given. We run the algorithm of Theorem 10′ with parameters $\epsilon_{\mathrm{bprobs}} = (\frac{\epsilon}{36nb^3})^2$, $\epsilon_{\mathrm{minwt}} = \frac{\epsilon}{3kn\ln(1296b^7n^2/\epsilon^2)}$, and $\epsilon_{\mathrm{wts}} = \frac{\epsilon^3}{27}$. With these parameters the algorithm runs in time $\mathrm{poly}(n/\epsilon)\cdot\log\frac{1}{\delta}$. By Theorem 10′, we get as output a list of poly(n/ϵ) many candidate parameter settings $\langle\{\hat{\pi}^i\}, \{\hat{\mu}^i_j\}\rangle$ with the guarantee that with probability $1-\delta$ at least one of the settings satisfies

- $|\pi^i - \hat{\pi}^i| \le \epsilon_{\mathrm{wts}}$ for all $i \in [k]$, and
- $|\hat{p}^i_{j,\ell} - p^i_{j,\ell}| \le \epsilon_{\mathrm{bprobs}}$ for all i, j, ℓ such that $\pi^i \ge \epsilon_{\mathrm{minwt}}$.

We now pass each of these candidate parameter settings through Theorem 11. It follows that the resulting distributions each satisfy $\epsilon^n_{\mathrm{bprobs}} = (\frac{\epsilon}{36nb^3})^{2n} \le \mathbf{Z}'(x) \le 1$ for all $x \in \{0,1\}^n$. A routine verification shows that with our choice of $\epsilon_{\mathrm{bprobs}}$, $\epsilon_{\mathrm{minwt}}$, and ϵ_{wts} we have

$$n\cdot(12b^3\epsilon^{1/2}_{\mathrm{bprobs}}) \le \frac{\epsilon}{3}, \quad k\epsilon_{\mathrm{minwt}}n\ln\frac{b}{\epsilon_{\mathrm{bprobs}}} \le \frac{\epsilon}{3}, \quad \text{and} \quad \epsilon^{1/3}_{\mathrm{wts}} \le \frac{\epsilon}{3}.$$

Thus $\eta(\epsilon_{\mathrm{bprobs}}, \epsilon_{\mathrm{wts}}, \epsilon_{\mathrm{minwt}}) \le \epsilon$, and we have that at least one of the resulting distributions $\mathbf{Z}^\star$ satisfies $\mathrm{KL}(\mathbf{Z}||\mathbf{Z}^\star) \le \epsilon$. □

8. Finding a good hypothesis using maximum likelihood.

Theorem 15 gives us a list of distributions, at least one of which is close to the target mixture distribution $\mathbf{Z}$ that we are trying to learn. Now we must *identify* some distribution in the list which is close to the target. In this section we give a simple maximum likelihood algorithm which helps us accomplish this. This is a standard situation (see, e.g., section 4.6 of [14]), and we emphasize that the ideas behind Theorem 16 below are not new. However, we were unable to find in the literature a clear statement of the exact result which we need, so for completeness we give our own statement and proof below.

Let $\mathbf{P}$ be a target distribution over some space X. Let $\mathcal{Q}$ be a set of hypothesis distributions such that at least one $\mathbf{Q}^* \in \mathcal{Q}$ has $\mathrm{KL}(\mathbf{P}||\mathbf{Q}^*) \le \epsilon$. The following algorithm will be used to find a distribution $\mathbf{Q}^{\mathrm{ML}} \in \mathcal{Q}$ which is close to $\mathbf{P}$. Draw a set

$\mathcal{S}$ of samples from the distribution $\mathbf{P}$. For each $\mathbf{Q} \in \mathcal{Q}$, compute the log-likelihood

$$\Lambda(\mathbf{Q}) = \sum_{x \in \mathcal{S}} (-\ln \mathbf{Q}(x)).$$

Now output the distribution $\mathbf{Q}^{\mathrm{ML}} \in \mathcal{Q}$ such that $\Lambda(\mathbf{Q})$ is minimum. This is known as the maximum likelihood (ML) algorithm since it outputs the distribution in $\mathcal{Q}$ which maximizes $\arg\max_{\mathbf{Q} \in \mathcal{Q}} \prod_{x \in S} \mathbf{Q}(x)$.

THEOREM 16. *Let $\beta, \alpha, \epsilon > 0$ be such that $\alpha < \beta$. Let $\mathcal{Q}$ be a set of hypothesis distributions for some distribution $\mathbf{P}$ over the space X such that at least one $\mathbf{Q}^* \in \mathcal{Q}$ has $\mathrm{KL}(\mathbf{P}||\mathbf{Q}^*) \leq \epsilon$. Suppose also that $\alpha \leq \mathbf{Q}(x) \leq \beta$ for all $\mathbf{Q} \in \mathcal{Q}$ and all x such that $\mathbf{P}(x) > 0$.*

Run the ML algorithm on $\mathcal{Q}$ using a set $\mathcal{S}$ of independent samples from $\mathbf{P}$, where $|\mathcal{S}| = m$. Then, with probability $1 - \delta$, where

$$\delta \leq (|\mathcal{Q}| + 1) \cdot \exp\left(-2m \frac{\epsilon^2}{\log^2(\beta/\alpha)}\right),$$

the algorithm outputs some distribution $\mathbf{Q}^{\mathrm{ML}} \in \mathcal{Q}$ which has $\mathrm{KL}(\mathbf{P}||\mathbf{Q}^{\mathrm{ML}}) \leq 4\epsilon$.

Before proving Theorem 16 we give some preliminaries. Let $\mathbf{P}$ and $\mathbf{Q}$ be arbitrary distributions over some space X. We can rewrite the KL divergence between $\mathbf{P}$ and $\mathbf{Q}$ as

$$\mathrm{KL}(\mathbf{P}||\mathbf{Q}) = -H(\mathbf{P}) - \sum_{x \in X} \mathbf{P}(x) \ln \mathbf{Q}(x), \tag{14}$$

where $H(\mathbf{P}) = -\sum_{x \in X} \mathbf{P}(x) \ln \mathbf{P}(x)$ is the "entropy in nats" of $\mathbf{P}$.

Consider the random variable $-\ln \mathbf{Q}(x)$, where x is a sample from the distribution $\mathbf{P}$. Using (14), we can express the expectation of this variable in terms of the KL divergence:

$$\mathbf{E}_{x \in \mathbf{P}}[-\ln \mathbf{Q}(x)] = \mathrm{KL}(\mathbf{P}||\mathbf{Q}) + H(\mathbf{P}). \tag{15}$$

Recall that when the ML algorithm runs on a list $\mathcal{Q}$ of distributions, it uses a set $\mathcal{S}$ of independent samples from $\mathbf{P}$, where $m = |\mathcal{S}|$. For each distribution $\mathbf{Q} \in \mathcal{Q}$, the algorithm computes

$$\Lambda(\mathbf{Q}) = \sum_{x \in \mathcal{S}} (-\ln \mathbf{Q}(x)).$$

So, by (15), we have that the expected "score" of distribution $\mathbf{Q}$ is the following:

$$\mathbf{E}_{\mathcal{S}}[\Lambda(\mathbf{Q})] = m(H(\mathbf{P}) + \mathrm{KL}(\mathbf{P}||\mathbf{Q})). \tag{16}$$

We recall the theorem of Hoeffding [16], as follows.

THEOREM 17 (Hoeffding). *Let $x_1, \ldots, x_n$ be independent bounded random variables such that each x_i falls into the interval $[a, b]$ with probability one. Let $X = \sum_{i=1}^{n} x_i$. Then for any $t > 0$ we have*

$$\Pr[X - \mathbf{E}[X] \geq t] \leq e^{-2t^2/n(b-a)^2} \quad \textit{and} \quad \Pr[X - \mathbf{E}[X] \leq -t] \leq e^{-2t^2/n(b-a)^2}.$$

Now we can prove Theorem 16.

Proof of Theorem 16. Call a distribution $\mathbf{Q} \in \mathcal{Q}$ *good* if $\mathrm{KL}(\mathbf{P}||\mathbf{Q}^{\mathrm{ML}}) \leq 4\epsilon$, and *bad* otherwise. Note that, by assumption, we have at least one good distribution in $\mathcal{Q}$.

The probability δ that the algorithm fails to output some good distribution is at most the probability that either some bad distribution $\mathbf{Q}$ has $\Lambda(\mathbf{Q}) \leq m(H(\mathbf{P}) + 3\epsilon)$ or the good distribution $\mathbf{Q}^*$ has $\Lambda(\mathbf{Q}^*) \geq m(H(\mathbf{P}) + 2\epsilon)$. Thus, by a union bound, we have

$$\begin{aligned} \delta \;\leq\; & |\mathcal{Q}| \cdot \Pr\left[\Lambda(\mathbf{Q}) \leq m(H(\mathbf{P}) + 3\epsilon) \mid \mathrm{KL}(\mathbf{P}||\mathbf{Q}) \geq 4\epsilon\right] \\ & + \Pr\left[\Lambda(\mathbf{Q}^*) \geq m(H(\mathbf{P}) + 2\epsilon)\right]. \end{aligned} \tag{17}$$

For each bad $\mathbf{Q} \in \mathcal{Q}$ which has $\mathrm{KL}(\mathbf{P}||\mathbf{Q}) > 4\epsilon$ we have

$$\begin{aligned} \Pr[\Lambda(\mathbf{Q}) \leq m(H(\mathbf{P}) + 3\epsilon)] &= \Pr[\Lambda(\mathbf{Q}) \leq m(H(\mathbf{P}) + 4\epsilon) - \epsilon m)] \\ &\leq \Pr[\Lambda(\mathbf{Q}) \leq m(H(\mathbf{P}) + \mathrm{KL}(\mathbf{P}||\mathbf{Q})) - \epsilon m)] \qquad (18) \\ &= \Pr[\Lambda(\mathbf{Q}) \leq \mathbf{E}_{\mathcal{S}}[\Lambda(\mathbf{Q})] - \epsilon m] \qquad (19) \\ &\leq \exp\left(-2m\frac{\epsilon^2}{\log^2(\beta/\alpha)}\right). \qquad (20) \end{aligned}$$

Equation (18) follows from the bound on the KL divergence, (19) follows from (16), and (20) follows from the Hoeffding bound (Theorem 17).

Following the same logic for $\mathbf{Q}^*$ where $\mathrm{KL}(\mathbf{P}||\mathbf{Q}^*) \leq \epsilon$, we get

$$\begin{aligned} \Pr[\Lambda(\mathbf{Q}^*) \geq m(H(\mathbf{P}) + 2\epsilon)] &= \Pr[\Lambda(\mathbf{Q}^*) \geq m(H(\mathbf{P}) + \epsilon) + m\epsilon] \\ &\leq \Pr[\Lambda(\mathbf{Q}^*) \geq m(H(\mathbf{P}) + \mathrm{KL}(\mathbf{P}||\mathbf{Q}^*)) + m\epsilon] \\ &= \Pr[\Lambda(\mathbf{Q}^*) \geq \mathbf{E}_{\mathcal{S}}[\Lambda(\mathbf{Q}^*)] + m\epsilon] \\ &\leq \exp\left(-2m\frac{\epsilon^2}{\log^2(\beta/\alpha)}\right). \qquad (21) \end{aligned}$$

Theorem 16 follows from plugging (20) and (21) into (17). □

9. Putting it all together.

All the pieces are now in place for us to prove our main learning result, Theorem 2, for learning mixtures of product distributions over $\{0, \dots, b-1\}^n$.

Proof of Theorem 2. Run the algorithm described in Theorem 15. With probability $1 - \delta$ this produces a list of $T = \mathrm{poly}(n/\epsilon)$ many hypothesis distributions, one of which has KL divergence at most ϵ from $\mathbf{Z}$ and each of which puts weight at least $(\frac{\epsilon}{36nb^3})^{2n}$ on every point in $\{0, \dots, b-1\}^n$. Now run the ML algorithm with $\alpha = (\frac{\epsilon}{36nb^3})^{2n}$, $\beta = 1$, and $m = \mathrm{poly}(n, 1/\epsilon)\ln(T/\delta)$. By Theorem 16, with probability at least $1 - \delta$ the ML algorithm outputs a hypothesis with KL divergence at most 4ϵ from $\mathbf{Z}$. Thus with overall probability $1 - 2\delta$ we get a hypothesis with KL divergence at most 4ϵ from $\mathbf{Z}$, and the total running time is $\mathrm{poly}(n/\epsilon) \cdot \log(1/\delta)$. Replacing ϵ by $\epsilon/4$ and δ by $\delta/2$, we are done. □

Tracing through the proofs, it is easy to check that the running time dependence on k is $(n/\epsilon)^{O(k^3)} \cdot \log \frac{1}{\delta}$.

10. Hardness of learning mixtures of product distributions.

In this section we give evidence that the class of mixtures of $k(n)$ product distributions over the Boolean cube may be hard to learn in polynomial time for any $k(n) = \omega(1)$.

Before describing our results, we recall some standard terminology about Boolean *decision trees*. A decision tree is a rooted binary tree in which each internal node has two children and is labeled with a variable and each leaf is labeled with a bit $b \in \{0, 1\}$. A decision tree T computes a Boolean function $f : \{0, 1\}^n \to \{0, 1\}$ in the obvious way: on input $x \in \{0, 1\}^n$, if variable x_i is at the root of T, we go to either the left or right subtree, depending on whether x_i is 0 or 1. Continue in this fashion until reaching a bit leaf; the value of this bit is $f(x)$.

Our main result in this section is the following theorem.

THEOREM 18. *For any function $k(n)$, if there is a poly(n/ϵ)-time algorithm which learns a mixture of $k(n)$ many product distributions over $\{0, 1\}^n$, then there is a poly(n/ϵ) time uniform distribution PAC learning algorithm which learns the class of all $k(n)$-leaf decision trees.*

We note that after years of intensive research, no poly(n)-time uniform distribution PAC learning algorithm is known which can learn $k(n)$-leaf decision trees for any $k(n) = \omega(1)$; indeed, such an algorithm would be a major breakthrough in computational learning theory.[4] The fastest algorithms to date [12, 3] can learn $k(n)$-leaf decision trees under the uniform distribution in time $n^{\log k(n)}$. This suggests that it may be impossible to learn mixtures of a superconstant number of product distributions over $\{0, 1\}^n$ in polynomial time.

The basic idea behind this theorem is quite simple. Given any $k(n)$-leaf decision tree T, the set of all positive examples for T is a union of at most $k(n)$ many disjoint subcubes of $\{0, 1\}^n$, and thus the uniform distribution over the positive examples is a mixture of at most $k(n)$ product distributions over $\{0, 1\}^n$. If we can obtain a high-accuracy hypothesis mixture $\mathcal{D}$ for this mixture of product distributions, then roughly speaking $\mathcal{D}$ must put "large" weight on the positive examples and "small" weight on the negative examples. We can thus use $\mathcal{D}$ to make accurate predictions of T's value on new examples very simply as follows: given a new example x to classify, we simply compute the probability weight that the hypothesis mixture $\mathcal{D}$ puts on x, and output 1 or 0 depending on whether this weight is large or small.

We now give the formal proof of Theorem 18. The following claim is used in the proof.

CLAIM 1. *Let T be a k-leaf decision tree, let $b \in \{-1, 1\}$ be a bit, let $S = \{x \in \{0, 1\}^n : T(x) = b\}$, and let $\mathcal{U}_S$ denote the uniform distribution over S. Then $\mathcal{U}_S$ is a mixture of k product distributions.*

Proof. We show that $\mathcal{U}_S$ is a mixture of ℓ product distributions, where ℓ is the number of leaves in T which are labeled with bit b. To see this, observe that the k leaves of T partition $\{0, 1\}^n$ into k disjoint subsets, each consisting of those $x \in \{0, 1\}^n$ which reach the corresponding leaf. For a leaf at depth d the corresponding subset is of size 2^{n-d} and consists of those $x \in \{0, 1\}^n$ which satisfy the length-d conjunction defined by the path from the root to that leaf. Thus, choosing a uniform element of S can be performed by the following process: (i) choose a leaf whose label is b, where each leaf at depth d is chosen with probability proportional to $1/2^d$; and then (ii) choose a uniform random example from the set of examples which satisfy the conjunction corresponding to that leaf. The uniform distribution over examples which satisfy a given conjunction is easily seen to be a product distribution $\mathbf{X}$ over $\{0, 1\}^n$ in which $\mathbf{E}[\mathbf{X}_i] \in \{0, \frac{1}{2}, 1\}$ for all $i = 1, \ldots, n$. It follows that the uniform distribution over S is a mixture of ℓ product distributions of this sort. □

Proof of Theorem 18. We suppose that we are given access to an oracle EX$(T, \mathcal{U})$ which, at each invocation, supplies a labeled example $(x, T(x)) \in \{0, 1\}^n \times \{0, 1\}$,

[4] Avrim Blum has offered a \$1000 prize for solving a subproblem of the $k(n) = n$ case and a \$500 prize for a subproblem of the $k(n) = \log n$ case; see [4].

where x is chosen from the uniform distribution $\mathcal{U}$ over $\{0,1\}^n$ and T is the unknown $k(n)$-leaf decision tree to be learned. We describe an efficient algorithm A' which with probability $1-\delta$ outputs a hypothesis $h : \{0,1\}^n \to \{0,1\}$ which satisfies $\Pr_{\mathcal{U}}[h(x) \neq T(x)] \leq \epsilon$. The algorithm A' uses as a subroutine an algorithm A which learns a mixture of $k(n)$ product distributions. Let M be the number of examples required by algorithm A to learn an unknown mixture of $k(n)$ product distributions to L_1-norm accuracy $1 - \frac{\epsilon}{2}$ and confidence $1 - \frac{\delta}{3}$. Recall from section 1.1 that to learn to L_1-norm error ϵ it suffices to learn to KL divergence ϵ^2, and thus we have that $M = \text{poly}(n/\epsilon)$ by our assumption on the running time of A.

Algorithm A' works as follows:

1. Determine $b \in \{-1,1\}$ such that with probability $1 - \frac{\delta}{3}$ tree T outputs b on at least 1/3 of the inputs in $\{0,1\}^n$. Let S denote $\{x \in \{0,1\}^n : T(x) = b\}$, and let $\mathcal{U}_S$ denote the uniform distribution over S.
2. Run algorithm A using samples from the uniform distribution $\mathcal{U}_S$; simulate $\mathcal{U}_S$ by invoking $\text{EX}(T,\mathcal{U})$, and using the only examples with labels $T(x) = b$. To be confident that algorithm A receives at least M examples from $\mathcal{U}_S$, we draw $\Theta(M \log(1/\delta))$ examples from $\text{EX}(T,\mathcal{U})$. Let $\mathcal{D}'$ be the hypothesis distribution which is the output of A.
3. Output the hypothesis $h : \{0,1\}^n \to \{-1,1\}$, which is defined as follows: given x, if $\mathcal{D}'(x) \leq \frac{1}{2 \cdot 2^n}$, then $h(x) = -b$; else $h(x) = b$.

We now verify the algorithm's correctness. Note first that step 1 can easily be performed by making $O(\log \frac{1}{\delta})$ draws from $\text{EX}(T,\mathcal{U})$ to obtain an empirical estimate of $\Pr_{\mathcal{U}}[T(x) = b]$. Assuming that $|S|$ is indeed at least $2^n/3$, a simple Chernoff bound shows that $O(M \log \frac{1}{\delta})$ draws from $\text{EX}(T,\mathcal{U})$ suffice to obtain M examples with label b in step 2 with probability $1 - \frac{\delta}{3}$. We run A on examples generated by $\mathcal{U}_S$, which by Claim 1 is a mixture of k product distributions. Consequently, with overall probability at least $1 - \delta$ the hypothesis $\mathcal{D}'$ generated in step 2 satisfies $\|D' - \mathcal{U}_S\|_1 \leq \frac{\epsilon}{2}$.

Now observe that the hypothesis h in step 3 disagrees with T on precisely those x which either (i) belong to S but have $\mathcal{D}'(x) < \frac{1}{2 \cdot 2^n}$, or (ii) do not belong to S but have $\mathcal{D}'(x) \geq \frac{1}{2 \cdot 2^n}$. Each x of type (i) contributes at least $\frac{1}{2 \cdot 2^n}$ toward $\|\mathcal{D}' - \mathcal{U}_S\|_1$ since $\mathcal{U}_S(x) \geq \frac{1}{2^n}$ for each $x \in S$. Each x of type (ii) also incurs at least $\frac{1}{2 \cdot 2^n}$ toward $\|\mathcal{D}' - \mathcal{U}_S\|_1$. Consequently, since $\|\mathcal{D}' - \mathcal{U}_S\|_1 \leq \frac{\epsilon}{2}$, there are at most $\epsilon 2^n$ points $x \in \{0,1\}^n$ on which h is wrong. Thus, we have shown that, with probability at least $1 - \delta$, the hypothesis h is an ϵ-accurate hypothesis for T with respect to the uniform distribution, as desired. □

Remark 2. We note that our reduction to decision tree learning in fact uses only quite restricted mixtures of product distributions in which (i) the mixture coefficients are proportional to powers of 2, (ii) the supports of the product distributions in the mixture are mutually disjoint, and (iii) each product distribution is a uniform distribution over some subcube of $\{0,1\}^n$ (equivalently, each product distribution has each $\mathbf{E}[\mathbf{X}_i] \in \{-1,0,1\}$). Thus, even this restricted class of mixtures of $k(n)$ product distributions is as hard to learn as $k(n)$-leaf decision trees.

Remark 3. The known results of Blum et al. [5] imply the following unconditional hardness result: the class of $k(n)$-leaf decision trees cannot be learned under the uniform distribution in time less than $n^{\log k(n)}$ in the model of learning from *statistical queries.*

A "statistical query" learning algorithm is allowed to obtain only statistical estimates (accurate to within some specified error tolerance) of properties of the distribution over pairs $(x, T(x))$, and does not have access to actual labeled examples $(x, T(x))$.

The algorithm is “charged” more time for estimates with a higher precision guarantee; this is motivated by the fact that such high-precision estimates would normally be obtained, given access to random examples, by drawing a large sample and making an empirical estimate. (See [17] for a detailed description of the statistical query model.)

Note that our algorithm for learning mixtures of product distributions interacts with the data solely by constructing empirical estimates of probabilities; thus, when this algorithm is used in the reduction of Theorem 18, the resulting algorithm for learning decision trees is easily seen to have an equivalent statistical query algorithm. Thus the results of Blum et al. unconditionally imply that no algorithm with the same basic approach as our algorithm can learn mixtures of $k(n)$ product distributions in time less than $n^{\log k(n)}$.

11. Conclusions and future work.

We have shown how to learn mixtures of any constant number of product distributions over $\{0,1\}^n$, and more generally over $\{0,\dots,b-1\}^n$, in polynomial time.

The methods we use are quite general and can be adapted to learn mixtures of other types of multivariate product distributions which are definable in terms of their moments. Along these lines, we have used the approach in this paper to give a PAC-style algorithm for learning mixtures of $k = O(1)$ axis-aligned Gaussians in polynomial time [13]. (We note that while some previous work on learning mixtures of Gaussians from a clustering perspective can handle $k = \omega(1)$ many component Gaussians, all such work assumes that there is some minimum separation between the centers of the component Gaussians, since otherwise clustering is clearly impossible. In contrast, our result in [13]—in which we do not attempt to do clustering but instead find a hypothesis distribution with small KL divergence from the target mixture—does not require us to assume that the component Gaussians are separated.) We expect that our techniques can also be adapted to learn mixtures of other distributions such as products of exponential distributions or beta distributions.

It is natural to ask whether our approach can be extended to learn mixtures of distributions which are not necessarily product distributions; this is an interesting direction for future work. Note that our main algorithmic ingredient, algorithm WAM, requires only that the coordinate distributions be pairwise independent.

Finally, one may also ask if it is possible to improve the efficiency of our learning algorithms—can the running times be reduced to $n^{O(k^2)}$, to $n^{O(k)}$, or even $n^{O(\log k)}$?

REFERENCES

[1] S. Aaronson, *Multilinear formulas and skepticism of quantum computation*, in Proceedings of the 36th Annual Symposium on Theory of Computing (STOC), Chicago, IL, 2004, ACM, New York, pp. 118–127.

[2] S. Arora and R. Kannan, *Learning mixtures of arbitrary Gaussians*, in Proceedings of the 33rd Annual Symposium on Theory of Computing (STOC), Heraklion, Crete, 2001, ACM, New York, pp. 247–257.

[3] A. Blum, *Rank-r decision trees are a subclass of r-decision lists*, Inform. Process. Lett., 42 (1992), pp. 183–185.

[4] A. Blum, *Learning a function of r relevant variables (open problem)*, in Proceedings of the 16th Annual Conference on Learning Theory and 7th Kernel Workshop, Washington, DC, 2003, pp. 731–733.

[5] A. Blum, M. Furst, J. Jackson, M. Kearns, Y. Mansour, and S. Rudich, *Weakly learning DNF and characterizing statistical query learning using Fourier analysis*, in Proceedings of the 26th Annual Symposium on Theory of Computing (STOC), Montréal, QC, 1994, ACM, New York, pp. 253–262.

[6] S. Chawla, C. Dwork, F. McSherry, A. Smith, and H. Wee, *Towards privacy in public databases*, in Proceedings of the 2nd Theory of Cryptography Conference (TCC), 2005, pp. 363–385.

[7] T. Cover and J. Thomas, *Elements of Information Theory*, Wiley, New York, 1991.

[8] M. Cryan, *Learning and Approximation Algorithms for Problems Motivated by Evolutionary Trees*, Ph.D. thesis, Computer Science, University of Warwick, Warwick, UK, 1999.

[9] M. Cryan, L. A. Goldberg, and P. W. Goldberg, *Evolutionary trees can be learned in polynomial time in the two-state general Markov model*, SIAM J. Comput., 31 (2001), pp. 375–397.

[10] S. Dasgupta, *Learning mixtures of Gaussians*, in Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS), New York, 1999, IEEE Computer Society Press, Piscataway, NJ, pp. 634–644.

[11] S. Dasgupta and L. Schulman, *A two-round variant of EM for Gaussian mixtures*, in Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, Seattle, WA, 2000, Morgan Kaufmann, San Francisco, pp. 143–151.

[12] A. Ehrenfeucht and D. Haussler, *Learning decision trees from random examples*, Inform. and Comput., 82 (1989), pp. 231–246.

[13] J. Feldman, R. O'Donnell, and R. Servedio, *PAC learning mixtures of axis-aligned Gaussians with no separation assumption*, in Proceedings of the 19th Annual Conference on Learning Theory (COLT), Pittsburgh, PA, 2006, Springer, New York, pp. 20–34; available online at http://arxiv.org/abs/cs.LG/0609093.

[14] Y. Freund and Y. Mansour, *Estimating a mixture of two product distributions*, in Proceedings of the 12th Annual Conference on Computational Learning Theory, Santa Cruz, CA, 1999, ACM, New York, pp. 183–192.

[15] S. Goreinov, E. Tyrtyshnikov, and N. Zamarashkin, *A theory of pseudoskeleton approximations*, Linear Algebra Appl., 261 (1997), pp. 1–21.

[16] W. Hoeffding, *Probability inequalities for sums of bounded random variables*, J. Amer. Statist. Assoc., 58 (1963), pp. 13–30.

[17] M. Kearns, *Efficient noise-tolerant learning from statistical queries*, J. ACM, 45 (1998), pp. 983–1006.

[18] M. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R. Schapire, and L. Sellie, *On the learnability of discrete distributions*, in Proceedings of the 26th Annual Symposium on Theory of Computing (STOC), Montréal, QC, 1994, ACM, New York, pp. 273–282.

[19] B. Lindsay, *Mixture Models: Theory, Geometry, and Applications*, Institute for Mathematical Statistics, Beachwood, OH, 1995.

[20] E. Mossel and S. Roch, *Learning nonsingular phylogenies and hidden Markov models*, in Proceedings of the 37th Annual Symposium on Theory of Computing (STOC), Baltimore, MD, 2005, pp. 366–375.

[21] R. A. Redner and H. F. Walker, *Mixture densities, maximum likelihood, and the EM algorithm*, SIAM Rev., 26 (1984), pp. 195–239.

[22] A. Smith, *Personal communication*, 2005.

[23] D. M. Titterington, A. F. M. Smith, and U. E. Makov, *Statistical Analysis of Finite Mixture Distributions*, Wiley, New York, 1985.

[24] S. Vempala and G. Wang, *A spectral algorithm for learning mixtures of distributions*, in Proceedings of the 43rd Annual Symposium on Foundations of Computer Science (FOCS), Vancouver, BC, 2002, IEEE Computer Society Press, Piscataway, NJ, pp. 113–122.

SIAM J. COMPUT.
Vol. 37, No. 5, pp. 1565–1594

HOLOGRAPHIC ALGORITHMS[*]

LESLIE G. VALIANT[†]

Abstract. Complexity theory is built fundamentally on the notion of efficient reduction among computational problems. Classical reductions involve gadgets that map solution fragments of one problem to solution fragments of another in one-to-one, or possibly one-to-many, fashion. In this paper we propose a new kind of reduction that allows for gadgets with many-to-many correspondences, in which the individual correspondences among the solution fragments can no longer be identified. Their objective may be viewed as that of generating interference patterns among these solution fragments so as to conserve their sum. We show that such holographic reductions provide a method of translating a combinatorial problem to finite systems of polynomial equations with integer coefficients such that the number of solutions of the combinatorial problem can be counted in polynomial time if one of the systems has a solution over the complex numbers. We derive polynomial time algorithms in this way for a number of problems for which only exponential time algorithms were known before. General questions about complexity classes can also be formulated. If the method is applied to a #P-complete problem, then polynomial systems can be obtained, the solvability of which would imply $\mathrm{P}^{\#\mathrm{P}} = \mathrm{NC2}$.

Key words. computational complexity, enumeration

AMS subject classifications. 05A15, 68Q10, 68Q15, 68Q17, 68R10, 68W01

DOI. 10.1137/070682575

1. Introduction.

Efficient reduction is perhaps the most fundamental notion on which the theory of computational complexity is built. The purpose of this paper is to introduce a new notion of efficient reduction, called a *holographic reduction*. In a classical reduction an instance of one problem is mapped to an instance of another by replacing its parts by certain *gadgets*. Solution fragments of the first problem will correspond in the gadgets to solution fragments of the second problem. For example, when mapping a Boolean satisfiability problem to a graph theory problem, each way of satisfying a part of the formula will correspond to a way of realizing a solution to the graph theory problem in the gadget. In classical reductions the correspondence between the solution fragments of the two problems is essentially one-to-one, or possibly many-to-one or one-to-many. In a holographic reduction the *sum* of the solution fragments of one problem maps to the *sum* of the solution fragments of the other problem for any one gadget and does so in such a way that the sum of all the overall solutions of the one will map to the sum of all the overall solutions of the other. The gadgets therefore map solution fragments *many-to-many*. The main innovation this allows is that it permits reductions in which correspondences between the solution fragments of the two problems need no longer be identifiable at all. Their effect can be viewed as that of producing interference patterns among the solution fragments, and they are called *holographic gadgets* for that reason.

[*] Received by the editors May 17, 2005; accepted for publication (in revised form) July 27, 2007; published electronically February 8, 2008. This research was supported in part by grants NSF-CCR-98-77049, NSF-CCR-03-10882, and NSF-CCF-04-27129 and by the National Security Agency (NSA) and Advanced Research and Development Activity (ARDA) under Army Research Office (ARO) contract DAAD19-01-1-0506. Preliminary versions of this paper appeared in *Proceedings of the 45th Symposium on Foundations of Computer Science*, IEEE, Los Alamitos, CA, 2004, pp. 306–315, and as Electronic Colloquium on Computational Complexity Report 99 (2005).

http://www.siam.org/journals/sicomp/37-5/68257.html

[†] School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138 (valiant@seas.harvard.edu).

A holographic reduction from a problem A to a problem B is of particular interest when for problem B the sum of the solutions is efficiently computable, since then a polynomial time algorithm for summing the solutions of A is implied. We call algorithms so derived *holographic algorithms.* In this paper we give holographic alogrithms for a number of problems for which no polynomial time algorithms were known before. We obtain these algorithms by reduction to the algorithm for finding perfect matchings in planar graphs due to Fisher [17], Kasteleyn [31], and Temperley and Fisher [50].

We consider holographic reductions and algorithms to be novel notions in algorithmic theory that do not appear to have been explored before, even in disguise, and that potentially open up new approaches to the central questions of complexity theory.

The most intriguing question, clearly, is whether polynomial time holographic algorithms exist for NP- or #P-complete problems. For such a result a holographic reduction would have to be exhibited from, say, a #P-complete problem, such as planar matchings, to a known polynomial time computable problem, such as planar perfect matchings. We shall show that the existence of such a reduction would be implied by the solvability of a finite system of polynomial equations that defines the holographic gadgets used in the reduction. In this sense the search for fast algorithms can be semimechanized if computer algebra systems are invoked for solving the systems. It suffices to find a fixed set of such gadgets. We note that the search process itself is NP-hard in the size of the tested system. On the other hand, one can expect that any fast algorithms so discovered would rely on algebraic relationships, possibly exotic, which have not been explored before even implicitly.

What is the role of holographic reductions in complexity theory if it is the case that there exist no polynomial algorithms to be discovered for NP- or #P-complete problems? In that eventuality we suggest that any proof of P $\neq$ NP may need to explain, and not only to imply, the unsolvability of our polynomial systems. Furthermore, explanations of such unsolvabilities may then stand equally in the way of any proofs of P $\neq$ P$^{\#\mathrm{P}}$, P $\neq$ BPP, P $\neq$ QBP, P $\neq$ NC2, and P = PSPACE. Since the solvability of a polynomial system for an explicit combinatorial constraint is a very natural mathematical problem, our approach may be viewed as one offering a restricted model of computation in which one expects mathematical questions to be resolvable, and one that we suggest may be difficult to evade.

Holographic algorithms are inspired by the quantum computational model [16, 6]. However, they are executable on classical computers and do not need quantum computers. They can be understood best, perhaps, in terms of cancellations in classical computation. Strassen's algorithm for matrix multiplication [48] offers an early striking example of the power of computations that compute extraneous terms only to cancel them later. It is known that cancellations can provide exponential speedups in computations, and in the several cases that have been analyzed, linear algebra algorithms for computing the determinant play a major role [54, 29, 49]. Further, the actual cancellations that are performed by certain of these determinant algorithms can be made explicit [55, 43]. Holographic algorithms offer a new source of cancellation that is not provided by linear algebra alone. Most importantly the cancellations required for the particular problem at hand can be *custom designed* into the holographic gadgets.

The substance of the holographic method as pursued here involves devising an appropriate *basis* for the reduction and then designing *matchgates* to realize the gadgets. Matchgates have been used previously [56] but only in the context of classical rather than holographic reductions. We note that the sum of solutions in matchgate constructions corresponds to the Pfaffian, which is polynomial time computable. The examples in this paper all refer to planar structures, because in that case we can use the elegant Fisher–Kasteleyn–Temperley (FKT) route to the Pfaffian that makes the design of the gadgets easier. In principle, the FKT technique can be applied to nonplanar structures by having matchgates to simulate crossovers. We note also that the holographic technique may be used, in principle, to reduce problems to any problem in which a quantity is known to be polynomial time computable. For example, in [58] it is used in reductions to the general Pfaffian.

2. List of problems. We first note that the range of natural graph-theoretic problems for which the number of solutions has been known to be countable in polynomial time for arbitrary inputs is very small [28, 62, 52, 53]. The prime examples have been the Kirchhoff matrix tree algorithm for spanning trees in arbitrary graphs and the FKT perfect matching algorithm for planar graphs. For planar graphs there is a positive result known for a further important case, intimately related to the Ising problem in physics, that is known to be obtainable from the perfect matchings problem. This is #PL-CUT—given a planar graph G and a number k the problem is to compute the number of 2-colorings of the nodes of G such that exactly k edges have ends of opposite color [32, 42]. The maximum k for which this number is nonzero is the well-known PL-MAXCUT problem [44, 22].

We now list some problems for which we can provide polynomial time solutions where none apparently were known. They are motivated by their apparent proximity to known NP-, $\oplus$P-, and #P-complete problems. They all have a counting, or #P, aspect, but for some we specify a decision or parity version when that is appropriate. We note that in a graph $G = (V, E)$ a subset $E' \subseteq E$ *saturates* a vertex $v \in V$ if v is the endpoint of some edge in E'.

First we consider a matching problem. Jerrum [26, 27] showed that counting the number of (not necessarily perfect) matchings in a planar graph is #P-complete, and Vadhan [51] subsequently proved that this was true even for planar bipartite graphs of degree 6. For degree 2 the problem can be solved easily and one might have conjectured that all other nontrivial cases are #P-complete. However, we have a polynomial time algorithm for the following.

#X-MATCHINGS. *Input.* A planar weighted bipartite graph $G = (V, E, W)$, where V has bipartition $V1$, $V2$ and the nodes in $V1$ have degree 2.

Output. The sum of the masses of all matchings of all sizes where the mass of a matching is the product of (i) the weights of all the edges present in the matching, as well as of the quantity (ii) "$-(w_1 + \cdots + w_k)$" for all the $V2$ nodes that are not saturated, where $w_1, \ldots, w_k$ are the weights of the edges incident to that (unsaturated) node.

One instance of this is where every $V2$ node has degree 4 and every edge has weight one. Then computing #X-MATCHINGS gives the number of matchings, but each weighted by $(-4)^k$, where k is the number of unsaturated $V2$ nodes. Computing this mod 5, for example, gives the number of matchings mod 5. Another instance is where every $V2$ node has degree 3 and every edge weight one. Then #X-MATCHINGS is the sum of the matchings, each weighted by $(-3)^k$, where k is the number of $V2$ nodes not saturated by that matching.

Now we consider a coloring problem. A *functional orientation of an undirected multigraph G* is an assignment of directions to a set of edges so that there is exactly one edge directed away from each node of G. (Note that if two nodes are connected by two edges, then these can both have (opposite) directions. Also any single edge may be assigned two opposite directions. The edges of G that are not assigned a direction remain undirected.)

PL-FO-2-COLOR. *Input.* A planar multigraph graph $G = (V, E)$ of maximum degree 3.

Output. 1 iff there is some coloring of the nodes with two colors and a functional orientation of G such that every edge that joins two nodes of the same color is directed in at least one direction by the functional orientation.

Comment. The problem of (2,1)-coloring with defects is that of 2-coloring a graph so that no node is adjacent to more than one other node of the same color. This is NP-complete for planar graphs of degree 5 [15]. It can be deduced that PL-FO-2-COLOR is NP-complete for degree 10 by means of the following reduction. For an instance of (2,1)-coloring one replaces each edge by a pair of edges between the same pair of nodes. Then if these nodes are given the same color the rules of PL-FO-2-COLOR ensure that the two edges are both oriented and in opposite directions. But then no other neighbor of either of the nodes can have the same color because the corresponding statement for those would imply that there are two edges directed away from that common node.

Our next two problems can be viewed as planar formula problems in the sense of Lichtenstein [35]: A planar formula is a planar graph where a node can represent a clause or a variable, and an edge links a node representing a variable with a node representing a clause in which it occurs, either negated or not negated.

⊕PL-EVEN-LIN-2. *Input.* A planar formula where each clause is a linear equation over $GF[2]$ with an even number of occurrences of variables, a subset of the clauses that are considered compulsory to satisfy, a setting to a subset of the variables on the outer face to constants, and an integer k.

Output. The parity of the number of solutions that satisfy exactly k of the equations, including all of the compulsory ones, and the boundary conditions.

Comment. This generalizes ⊕PL-CUT, which is the same problem restricted to equations with just two variables and no compulsory equations and can be solved by classical reduction to FKT. The nonplanar version with two variables is NP- and ⊕P-complete via known parsimonious reductions, and strong hardness of approximation results are also known [23]. For odd length equations the corresponding planar problem is ⊕P-complete since the corresponding nonplanar problem can be reduced to it using the construction of crossovers from Jerrum [26, 27]. This construction requires the equations in the crossovers to be compulsory, and without such compulsory equations the completeness of the problem is apparently unresolved.

# PL-3-NAE-SAT. *Input.* A planar formula F consisting of NOT-ALL-EQUAL gates of size 3.

Output. The number satisfying assignments of F.

Comment. For connectives other than NOT-ALL-EQUAL (e.g., OR, EXACTLY ONE) for which the unrestricted decision problem is NP-complete, the corresponding planar decision and counting problems are, in general, NP- and #P-complete, respectively [24]. The existence problem for monotone PL-3-NAE-SAT is reducible to the four color theorem and, therefore, always has a solution [1]. Note, however, that the counting problem for the 4-colorings of planar graphs is #P-complete [61]

PL-NODE-BIPARTITION. *Input.* A planar graph $G = (V, E)$ of maximum degree 3.

Output. The cardinality of a smallest subset $V' \subseteq V$ such that the deletion of V'

and its incident edges results in a bipartite graph.

Comment. This problem is known to be NP-complete for maximum degree 6 [33]. See Lewis and Yannakakis [34] for a general approach to such "node deletion" problems. We note that numerous other planar NP-complete problems, such as Hamiltonian cycles and minimum vertex covers, are NP-complete already for degree 3 (see, e.g., Garey, Johnson, and Stockmeyer [20] and Garey and Johnson [18]).

We now consider "ice" problems that have been widely investigated by statistical physicists. An *orientation* of an undirected graph G is an assignment of a direction to each of its edges. An "ice problem" involves counting the number of orientations such that certain local constraints are satisfied. Pauling [47] originally proposed such a model for planar square lattices, where the constraint was that an orientation had to have two incoming and two outgoing edges at every node. The question of determining how the number of such orientations grows for various such planar *repeating* structures has been analyzed [36, 37, 38, 39, 3]; see also [62].

#PL-3-NAE-ICE. *Input.* A planar graph $G = (V, E)$ of maximum degree 3.

Output. The number of orientations such that no node has all the edges directed toward it or away from it.

We next turn to a covering problem. For a graph $G = (V, E)$ a *cycle* is a sequence of edges through distinct nodes that starts and ends at the same node. A *chain* is a sequence of edges through distinct nodes that starts and ends at distinct nodes. A *cycle-chain cover* in G is a set of cycles and chains that saturates every node of G. For real numbers x, y the (x, y) *cycle-chain sum* of G is the sum over all cycle-chain covers C of $x^i y^j$, where i is the number of cycles in C and j is the number of chains. For example, the $(2, k)$ cycle-chain sum for $k = 0$ or $k = 4$ is complete for $\oplus$P for general graphs since the parity of the number of Hamiltonian cycles is reducible to it. In the planar case it is known that counting the number of Hamiltonian cycles for planar cubic graphs is #P-complete [41]. Their proof can be adapted to show that if nodes of both degrees 2 and 3 are allowed, then planar Hamiltonian cycles is $\oplus$P-complete [59].

#PL-3-(1,1)-CYCLECHAIN. *Input.* A planar regular graph $G = (V, E)$ of maximum degree 3.

Output. The $(1, 1)$ cycle-chain sum.

As we shall further elaborate in section 9, the proofs given there of the last four of these results imply that they can be derived also by classical reduction to #PL-CUT. However, some of these problems also have degree 4 variants for which such classical reductions are not apparent.

3. Evaluating planar matching polynomials.

We shall first describe the basic graph-theoretic notions that we shall use. A (weighted undirected) *graph* G is a triple (V, E, W), where V is the set of n nodes, labeled $\{1, \ldots, n\}$, E is the set of edges where an edge is a pair (i, j) of distinct nodes i, $j \in V$, and W is an assignment of a weight $W(i, j)$ from a field F to each edge (i, j). An edge e is *incident to* or *saturates* a node j if j is one of the pair of nodes of e. A *matching* in G is a set $E' \subseteq E$ of edges such that if e_1 and e_2 are distinct edges in E', then e_1 and e_2 saturate disjoint pairs of nodes. A matching E' *saturates* the union of the node pairs saturated by the member edges of E'. The set of nodes saturated by E' we call $satu(E')$. A matching is *perfect* if it saturates all of V.

With a graph G we associate the *perfect matching* polynomial PerfMatch(G) over

$n(n-1)/2$ variables $\{x_{i,j} | 1 \leq i < j \leq n\}$ as follows:

$$\mathrm{PerfMatch}(G) = \sum_{E'} \prod_{(i,j) \in E'} x_{i,j},$$

where the summation is over all perfect matchings E' of G. We shall also discuss the more general *matching sum* polynomial for graphs $G = (V, E, W, \Lambda)$, where Λ further specifies a labeling of each node i by a weight $\lambda_i \in F$. It is defined as

$$\mathrm{MatchSum}(G) = \sum_{E'} \prod_{i \notin satu(E')} \lambda_i \prod_{i,j \in E'} x_{i,j},$$

where summation is over all, not necessarily perfect, matchings in G. Clearly in the case that every $\lambda_i = 0$, $\mathrm{PerfMatch}(G) = \mathrm{MatchSum}(G)$. We shall call nodes with $\lambda_i \neq 0$ *omittable* since matchings that omit them can contribute to the MatchSum.

For all polynomials we shall assume, where not otherwise specified, that the coefficients are taken from an arbitrary field F.

A remarkable fact, expressed by the following theorem, is that for *planar* graphs PerfMatch(G) can be expressed as a determinant of an easily computed matrix [17, 31, 32, 50, 28]. It follows that PerfMatch(G) can be computed using standard linear algebra algorithms for the determinant.

THEOREM 3.1. *There is a polynomial time computable function f that given a planar embedding of a planar graph $G = (V, E, W)$ defines $f: E \rightarrow \{-1, 1\}$ such that for the antisymmetric matrix M defined so that for all $i < j$*

(i) *if $(i, j) \notin E$ then $M_{i,j} = M_{j,i} = 0$, and*

(ii) *if $(i, j) \in E$ then $M_{i,j} = f(i, j)W(i, j)$ and $M_{j,i} = -f(i, j)W(i, j)$,*

it is the case that $\mathrm{PerfMatch}(G) = \mathrm{Pfaffian}(M) = \sqrt{\mathrm{Det}(M)}$.

In our applications we shall form graphs from fixed sets of standard components called matchgates that simulate particular combinatorial constraints, such as equality. The weights to be used in such matchgates will be elements of F obtained potentially by computationally solving systems of polynomial equations. It is therefore useful to observe that in this general setting the Det(M) and hence also PerfMatch(G) can be solved in polynomial time if the field is that of the complex numbers $\mathbb{C}$. The proof is given in section 12.

THEOREM 3.2. *Let Y be any finite subset of $\mathbb{C}$. Suppose that each element of Y can be computed to n decimal places, i.e., absolute error less than 2^{-n}, in time polynomial in n. Let $\{M_n \mid n \geq 1\}$ be a family of matrices where, for each n, M_n is $n \times n$, has every entry from Y, and has an integer valued determinant. Further, suppose that there is a polynomial time algorithm that given input $\{1^n, i, j\}$ will identify the element from Y that is the (i, j)th entry of M_n. Then there is a polynomial time deterministic algorithm that, given 1^n, will compute the determinant of M_n.*

Computing MatchSum for planar graphs is known to be #P-complete [26, 27]. Since matchings with omittable nodes are more expressive than those without, we might endeavor to use them wherever we can still maintain polynomial time computability. The following generalization of the above two results, which is also proved in section 12, enables us to use omittable nodes on the outer face of a planar graph.

THEOREM 3.3. *Let Y be any finite subset of $\mathbb{C}$. Suppose that each element of Y can be computed to n decimal places, i.e., absolute error less than 2^{-n}, in time polynomial in n. Let $\{G_n \mid n \geq 1\}$ be a family of planar embeddings of planar graphs on n nodes with all omittable nodes on the outer face, with polynomial time identifiable* *weights from Y, and having an integer value of MatchSum(G_n). Then MatchSum(G_n) can be computed in polynomial time, and, in fact, in NC2.*

We note that while we emphasize the case of the field $F = \mathbb{C}$, the whole development applies equally, and without the need for these numerical considerations, if F is a finite field. In that case the consequences are for $\#_k$P the counting class corresponding to #P, but modulo k [52].

We also note that for planar structures there exist algorithms that can perform elimination on $n \times n$ matrices in $O(n^{1.5})$ rather than $O(n^3)$ steps [40].

4. Matchgrids and planar matchgates. Our overall strategy is the following. We transform an instance I of a counting problem, such as #X-MATCHINGS, to an instance Ω of what we call a matchgrid, such that the weighted sum of the perfect matchings of Ω will equal the number of solutions of I. The structure of I is reflected in the structure of Ω, with the individual components of I, nodes and edges in the case of #X-MATCHINGS, each replaced by gadgets that we call matchgates. The weight of the perfect matchings in each matchgate will equal the number of solution fragments of the #X-MATCHING problem.

We now introduce the basic concepts of the theory. We note that while our starting point is the notion of a matchgate, exactly as in [56], that earlier work employed classical rather than holographic reductions. This paper can be read independently of that earlier one. However, we have attempted to keep our notation consistent with it and reference it occasionally.

A planar matchgate Γ is a triple (G, X, Y), where G is a planar embedding of a planar graph (V, E, W), $X \subseteq V$ is a set of *input* nodes, $Y \subseteq V$ is a set of *output* nodes, and X, Y are disjoint. Further, as one proceeds counterclockwise around the outer face, starting from one point one encounters first the input nodes labeled $1, 2, \ldots, \mid X \mid$ and then the output nodes $\mid Y \mid, \ldots, 2, 1$, in that order. The *arity* of the matchgate is $|X| + |Y|$. For $Z \subseteq X \cup Y$ we define the *standard signature of Γ with respect to* Z to be PerfMatch$(G - Z)$, where $G - Z$ is the graph obtained by removing from G the node set Z and all edges that are incident to Z. Further, we define the *standard signature* of Γ to be the $2^{|X|} \times 2^{|Y|}$ matrix $u(\Gamma)$ whose elements are the standard signatures of Γ with respect to Z for the $2^{|X|}2^{|Y|}$ choices of Z. The labeling of the matrix is as follows: Suppose that X and Y have the labeling described; i.e., the nodes are labeled $1, 2, \ldots, |X|$ and $|Y|, \ldots, 2, 1$ in counterclockwise order. Then each choice of Z corresponds to a subset from each of these labeled sets. If each node present in Z is regarded as a 1 and each node absent as a 0, then we have two binary strings of length $|X|$, $|Y|$, respectively, where the nodes labeled 1 correspond to the leftmost binary bit. Suppose that i, j are the numbers represented by these strings in binary. Then the entry corresponding to Z will be the one in row i and column j in the signature matrix $u(\Gamma)$.

We note that in [56] matchgates were defined in a general, not necessarily planar, setting. In that more general case, when we compose matchgates into matchcircuits we need to keep track of sign influences explicitly, since we cannot rely on the FKT method. For that reason we use there the more complex notion of character, while the simpler notion of signature suffices in this paper since we restrict ourselves to planar graphs.

The treatment in [56] is more general also in the second respect that omitted nodes are allowed in matchgates and character is defined in terms of MatchSum rather than PerfMatch. We accommodate this generalization in two limited ways in the current paper. We use them in a thought experiment in the proof of Theorem 4.1. We also

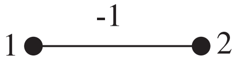


FIG. 1. *A generator matchgate for basis* **b1** *with output nodes* $\{1, 2\}$ *and one edge of weight* -1*. It generates* $n \otimes n + n \otimes p + p \otimes n$*.*

use them explicitly in circuits, as in Theorem 9.6, as allowed by Theorem 3.3 and Corollary 4.2, noting that the omittable nodes have to be on the outer face of the final circuit.

A *basis* of size k is a set of distinct nonzero vectors each of length 2^k with entries from a field F. Often we will have just two basis vectors that represent 0 and 1, respectively, and in that case we shall call them n and p. In this paper all bases will be of size $k = 1$, so that $n = (n_0, n_1)$ and $p = (p_0, p_1)$. The basis $\mathbf{b0} = [n, p] = [(1, 0), (0, 1)]$ we call the *standard basis*. In general, the vectors in a basis do not need to be independent.

In this section we shall use as an illustrative example the basis $\mathbf{b1} = [n, p] = [(-1, 1), (1, 0)]$. The gates we describe will be used in section 8 to implement our first holographic algorithm, one for the #X-MATCHINGS problem. We believe that this basis, though apparently somewhat specialized, is an instructive example. In later sections we shall describe bases, such as **b2**, which appear to be more broadly applicable.

In general if we have two vectors q, r, of length l, m, respectively, then we shall denote the *tensor product* $s = q \otimes r$ to be the vector s of length lm in which $s_{im+j} = q_i r_j$ for $0 \leq i < l$ and $0 \leq j < m$. Thus, for example, for the basis **b1**, $n \otimes p = (-1, 0, 1, 0)$. Clearly $\otimes$ is associative.

We say that a matchgate is a *generator* if it has zero input nodes and nonzero output nodes and a *recognizer* if it has zero output nodes and nonzero input nodes. One can define equally naturally a *transducer* gate that has both nonzero inputs and nonzero outputs, but we do not use these for our examples. From the definition of signature it follows that for generators and recognizers the signature is a vector.

Here we shall introduce generators and recognizers by example. A more formal treatment can be found at the beginning of section 5. Intuitively, a generator can be viewed as emitting n and p particles along its outputs in all possible combinations, each combination with a certain value. A recognizer will absorb combinations of these particles entering via its inputs, again attaching a certain value to each combination. The overall goal is that the sum over all patterns of particles that can be generated of the product of the values of all the generators and recognizers be equal to the value of the function being computed.

We first consider generators. Suppose that a generator has graph G and m output nodes. Then, by definition, its standard signature will be a 2^m-vector. Recall that element j in this vector is the value of $\mathrm{PerfMatch}(G')$, where G' is G but with those output nodes removed that correspond to the index j in the manner described in the definition of standard signature. Consider the generator matchgate Γ shown in Figure 1.

It has $V = \{1, 2\}$, $E = \{(1, 2)\}$, $W(1, 2) = -1$, the input node set $X = \emptyset$, and the output node set $Y = \{1, 2\}$. Then the standard signature $u(\Gamma)$ of Γ is the vector $(-1, 0, 0, 1)$ since if both output nodes are removed then $\mathrm{PerfMatch}(G') = 1$, if neither is removed then $\mathrm{PerfMatch}(G') = -1$, and if exactly one is removed then there is no perfect matching and $\mathrm{PerfMatch}(G') = 0$. Now for the basis **b1** defined above it is

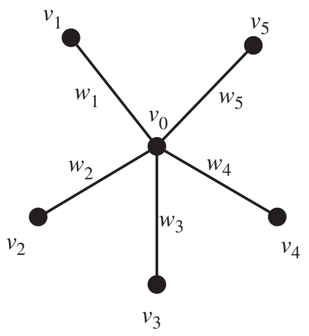


FIG. 2. *A recognizer matchgate for basis* **b1** *with input nodes* $v_1, v_2, \dots, v_5$*, and edge weights* $w_1, w_2, \dots, w_5$*.*

easy to see that $n \otimes n = (1, -1, -1, 1)$, $n \otimes p = (-1, 0, 1, 0)$, and $p \otimes n = (-1, 1, 0, 0)$. The sum of these is $(-1, 0, 0, 1)$, which happens to equal the above stated standard signature of the matchgate. (Note that here we used the convention that PerfMatch of a graph with no nodes is 1. This can be avoided by using as the generator a chain of four nodes, rather than two, and again having the end nodes as output nodes.) Hence we conclude that for this gate and basis **b1** the following holds.

PROPOSITION 4.1. *There exists a generator matchgate* Γ *with* $u(\Gamma) = n \otimes n + n \otimes p + p \otimes n$*, where* (n, p) *is the basis* **b1***.*

In other words, this gate generates the linear sum of the three bit combinations 00, 01, and 10 when interpreted in the basis **b1** representation. The *signature of this generator with respect to the basis* **b1** (a notion further elaborated in section 5, as relation (5.1)) will then be (1, 1, 1, 0) since these are the coefficients of the contributions for the four bit patterns 00, 01, 10, 11, respectively. For $x \in \{n, p\}^2$ we shall denote by $valG(\Gamma, x)$ the signature element corresponding to x. Thus, for the current example, $\mathrm{valG}(\Gamma, n \otimes p) = 1$ and $\mathrm{valG}(\Gamma, p \otimes p) = 0$. (We note that since a basis **b** can be an arbitrary set the signature of a generator with respect to **b** may not be unique. When we discuss a signature any valid signature will do.)

We shall now go on to discuss recognizers. Let us suppose that these have m inputs. The purpose of such recognizers is to have PerfMatch take on appropriate values as the inputs range over the 2^m possible tensor product values $x = x_1 \otimes \cdots \otimes x_m$, where each x_i ranges independently over $\{n, p\}$. Note that x can be viewed as a 2^m-vector in the standard basis. The value of PerfMatch for the recognizer matchgate Γ "evaluated at input" x will by denoted by $valR(\Gamma, x)$. More precisely, if vector u is the standard signature of Γ, and x is the 2^m-vector representing x in the standard basis, then $\mathrm{valR}(\Gamma, x)$ is the inner product $u\ x$. Consider the family of recognizers Γ_k shown in Figure 2. They are defined by the star graph $G_k = (V_k, E_k, W_k)$, where $V_k = \{v_0, v_1, \dots, v_k\}$, $E_k = \{(v_0, v_i) | 1 \le i \le k\}$, the input nodes are $\{v_i | 1 \le i \le k\}$, and the weight of edge (v_0, v_i) is w_i.

PROPOSITION 4.2. *For all* $k > 0$ *and for all* $w_1, \dots, w_k \in F$ *there exists a* k*-input recognizer matchgate* Γ *such that on input* $x = x_1 \otimes \cdots \otimes x_k \in \{n, p\}^k$ *over basis* **b1** $\mathrm{valR}(\Gamma, x)$ *equals*

(i) $-(w_1 + \cdots + w_k)$ *if* $x_1 = \cdots = x_k = n$,

(ii) w_i *if* $x_i = p$*, and* $x_j = n$ *for every* $j \neq i$,

(iii) 0 *for the remaining* $2^k - k - 1$ *values of* $x_1, \dots, x_k$.

Proof. We shall prove that the gate of Figure 2, where the weight of edge (v_0, v_i)

is set to w_i, is such a recognizer. To see this note that the only subsets Z of the input nodes $\{v_i | 1 \leq i \leq k\}$ that can be removed that allow the PerfMatch of the remaining graph to be nonzero are those that contain exactly $k-1$ elements, and for these PerfMatch $= w_i$ if v_i is the node omitted from Z. Hence if two or more of the inputs are $p = (1, 0)$, then PerfMatch is zero. If exactly one input is p, and this is applied at node v_i, and all the others are n, then the only nonzero contribution comes from the node v_i being omitted from Z, and this gives a contribution of $p_0 n_1^{k-1} w_i = w_i$. If all the inputs are n, then there is a nonzero contribution $n_0 n_1^{k-1} w_i = -w_i$ for each possible v_i, and then the total value of PerfMatch is $-(w_1 + \cdots + w_k)$. □

We note that the basic properties of a basis are unchanged if the first and second components of all of its elements are interchanged together, or if they are multiplied by arbitrary constants x and y, respectively. The former transformation can be realized by appending to every input or output node an edge of weight 1. The latter can be realized by appending to such nodes chains of length two weighted by x and y, respectively. Hence any basis $[(-x, y), (x, 0)]$ or $[(x, -y), (0, y)]$ with nonzero x and y is essentially equivalent to **b1**.

PROPOSITION 4.3. *If there is a generator (recognizer) with certain valG(valR) values for size one basis $\{(a_1, b_1), \ldots, (a_r, b_r)\}$, then there is a generator (recognizer) with the same valG(valR) values for any basis $\{(xa_1, yb_1), \ldots, (xa_r, yb_r)\}$ or $\{(xb_1, ya_1), \ldots (xb_r, ya_r)\}$ for any $x, y \in F$.*

We define a *matchgrid* over a basis $\mathbf{b}$ to be a weighted undirected planar graph G that consists of the disjoint union of a set of g generator matchgates $B_1, \ldots, B_g$, r recognizer matchgates $A_1, \ldots, A_r$, and f *connecting edges* $C_1, \ldots, C_f$, where each C_i edge has weight 1 and joins an output node in a generator matchgate with an input node of a recognizer matchgate, such that every input and output node in every constituent matchgate has exactly one such incident connecting edge.

Consider such a matchgrid $\Omega = (A, B, C)$ and assume, for simplicity, that the basis is of size two. We denote by $X = \mathbf{b}^f = (n, p)^f$ the set of 2^f possible combinations of the basis elements n, p that can be transmitted simultaneously along the f connecting edges in the matchgrid. We can break X into $X_1 \otimes \cdots \otimes X_g$, where $X_j = (n, p)^{k(j)}$ and $k(j)$ is the arity of generator B_j and refers to the connecting edges that are incident to that generator. Also if $x \in X$ then we can mirror this decomposition as $x = x_1 \otimes \cdots \otimes x_g$, where x_j is the particular set of basis elements that is transmitted from the outputs of B_j. We can also break the same X into $\bar{X}_1 \otimes \cdots \otimes \bar{X}_r$, where $\bar{X}_j = (n, p)^{l(j)}$ and $l(j)$ is the arity of the recognizer A_j and refers to the connecting edges incident to that recognizer. If $x \in X$ then this decomposition can be mirrored as $x = \bar{x}_1 \otimes \cdots \otimes \bar{x}_r$, where $\bar{x_j}$ is the set of basis elements transmitted into the inputs of A_j.

Now for each $x \in X$ each recognizer A_i will evaluate a value $\mathrm{valR}(A_i, x) = \mathrm{valR}(A_i, \bar{x}_i)$, and each generator B_j will generate the value $\mathrm{valG}(B_j, x) = \mathrm{valG}(B_j, x_j)$. The product of these values for all the generators and all the recognizers is the *value of the matchgrid* at x. The *value of the matchgrid* will be the sum of these products for the various x. This quantity we call the *Holant*:

$$\mathrm{Holant}(\Omega) = \sum_{x \in \mathbf{b}^f} \left[\prod_{1 \leq j \leq g} \mathrm{valG}(B_j, x_j) \right] \left[\prod_{1 \leq i \leq r} \mathrm{valR}(A_i, x) \right].$$

There are two views of a matchgrid, one as a directed weighted graph G and the other as a composition $\Omega = (A, B, C)$ of matchgates and connecting edges. For the former we have already defined various matching polynomials such as PerfMatch and it is these that we shall evaluate in polynomial time. For the latter it is the Holant that expresses the basic intention of the matchgrid, that of performing a weighted sum of potentially exponentially many solutions, indexed by the set X, that obey the local constraints expressed in the matchgates.

The central relationship that is necessary for a holographic algorithm is that the potentially exponential summation that the Holant defines be computable in polynomial time. The following is a paradigmatic expression of this. The reader should note that for the standard basis valG = valR, and the theorem follows immediately. More surprising, and at the heart of our holographic technique, is the fact is that the result holds for *all* bases.

THEOREM 4.1. *For any matchgrid Ω over any basis $\mathbf{b}$, if Ω has weighted graph G, then*

$$\mathrm{Holant}(\Omega) = \mathrm{PerfMatch}(G).$$

Proof. The result is a consequence of linearity. The following is a mechanistic way of presenting the argument.

Suppose for the sake of this proof that we allow a certain subset of the nodes of a matchgate to be "omittable with weight 1" in the sense that its signature will be defined by not just perfect matchings but also by all other matchings that saturate all the nonomittable nodes, but any omittable node may or may not be saturated. In other words we are using the polynomial MatchSum with $\lambda_i = 1$ for the omittable nodes, and $\lambda_i = 0$ for the unomittable nodes. Once we allow omittable nodes we have matchgates for any single basis elements such as p and n: Figure 3 shows a matchgate with omittable node 1 and output node 3. The standard signature is clearly (w_0, w_1) since if node 3 is not in Z then the only allowed matching is the edge (2,3) with weight w_0 and if node 3 is in Z then the only allowed matching is edge (1,2) with weight w_1. Hence we get a matchgate with standard signature $p = (p_0, p_1)$ by fixing $w_0 = p_0$ and $w_1 = p_1$, and one with standard signature $n = (n_0, n_1)$ by fixing $w_0 = n_0$ and $w_1 = n_1$.

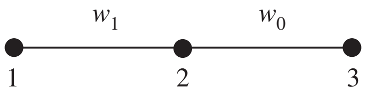


FIG. 3. *A generator matchgate having node* 1 *as an omittable node and node* 3 *as the output node. It has standard signature* (w_0, w_1).

Suppose we pick a *fixed* element $x \in X$ from among the $|\mathbf{b}|^f$ that are potentially generated, and regard it as the tensor product $x_1 \otimes \cdots \otimes x_g$, where x_i corresponds to the basis elements that are involved in generator B_i, and equivalently as a tensor product $\bar{x}_1 \otimes \cdots \otimes \bar{x}_r$, where $\bar{x}_j$ corresponds to the basis elements that are involved in recognizer A_j. Then we can construct from Ω a matchgrid $G(x)$ that replaces each generator B_i having k outputs by k generators of the single basis elements specified by x_i for those k outputs. Further, for each such B_i the parameters in one of these single basis element generators will be set so as to multiply its value by $\mathrm{valG}(B_i, x_i)$ so that these generators for B_i generate x_i with that multiplier $\mathrm{valG}(B_i, x_i)$. Then it follows from the definitions of generators, recognizers, and the way they are assembled according to the definition of matchgrids that

$$\mathrm{MatchSum}(G(x)) = \left[\prod_{1\leq i\leq g} \mathrm{valG}(B_i, x_i)\right]\left[\prod_{1\leq i\leq r} \mathrm{valR}(A_i, x)\right].$$

The reason for this equality is that a fixed vector $x \in X$ is being generated with weight $\Pi\mathrm{valG}(B_i, x)$, where the multiplication is over all the generators B_i. The inner product of this x with the standard signature u of each of the recognizers gives the contribution to MatchSum of the recognizers. But, by definition, the inner product ux equals $\mathrm{valR}(A_i, x)$ for each recognizer A_i.

Now partition X into equivalence classes of $|X_1|$ elements each so that all members of each equivalence class have identical $X_2, \ldots, X_g$ components. For each of these equivalence classes, say, the one defined by $x_2 \in X_2, \ldots, x_g \in X_g$, define the matchgrid $G(x_2, \ldots, x_g)$ as follows: Set x to have components $x_2, \ldots, x_g$ and any $x_1 \in X_1$, and let $G(x_2, \ldots, x_g)$ be $G(x)$ but with the single element generators for x_1 replaced by the generator B_1, which generates the sum of all members of X_1, each with the appropriate weight. Then clearly, summing over all the values of x_1 gives

$$\begin{aligned}&\mathrm{MatchSum}(G(x_2, \ldots, x_g))\\ &\quad= \sum_{x_1\in X_1} \mathrm{valG}(B_1, x_1)\left[\prod_{2\leq i\leq g} \mathrm{valG}(B_i, x_i)\right]\left[\prod_{1\leq i\leq r} \mathrm{valR}(A_i, x)\right],\end{aligned}$$

where x in the last term denotes $x_1 \otimes x_2 \otimes \cdots \otimes x_g$.

We iterate this process for $B_2, \ldots, B_g$ in turn. For example, for B_2 we partition $X_2 \otimes X_3 \otimes \cdots \otimes X_g$ into equivalence classes of $|X_2|$ elements each so that each class has identical $x_3, \ldots, x_g$ components. For each of these equivalence classes, say, that defined by $x_3 \in X_3, \ldots, x_g \in X_g$, we define matchgrid $G(x_3, \ldots, x_g)$ to be $G(x_2, \ldots, x_g)$ for some x_2, but with the single element generators for x_2 replaced by the generator B_2. This will sum all the members of X_2 with the appropriate weights. It then follows that

$$\begin{aligned}&\mathrm{MatchSum}(G(x_3, \ldots, x_g))\\ &\quad= \sum_{x_1\in X_1}\sum_{x_2\in X_2} \mathrm{valG}(B_1, x_1)\mathrm{valG}(B_2, x_2)\left[\prod_{3\leq i\leq g} \mathrm{valG}(B_i, x_i)\right]\left[\prod_{1\leq i\leq r} \mathrm{valR}(A_i, x)\right].\end{aligned}$$

After the last stage we have replaced all the generators of single basis elements and have just one matchgrid left, which is $G() = \Omega$. It follows then from the definition of the Holant that MatchSum(G) equals Holant(G). Note that at that point all the single element generators with omittable nodes have been replaced by the original generators with no omittable nodes, and hence the result also holds for PerfMatch(G) as claimed. □

We use the Holant theorem to express the intention of holographic reductions. A counting problem $\#F$ *has a simple holographic reduction* to planar PerfMatch if there is a transformation that (i) produces all edge weights from a fixed set Y in which each element can be computed to absolute error less than 2^{-n} in time polynomial in n, (ii) produces a weighted graph with the Holant and therefore also PerfMatch equal to $\#F$, and (iii) is computable in NC2.

COROLLARY 4.1. *If #F has a simple holographic reduction to planar PerfMatch then #F ∈ NC2.*

Proof. The instance of #F is first transformed to an instance of planar PerfMatch. By Theorem 3.1 the required solution is given by the square root of the determinant of a matrix that satisfies the conditions of Theorem 3.2. It then follows from Corollary 3.2.1 given in section 12 that this determinant and the required solution can be computed in NC2. □

All the reductions we exhibit in this paper are simple holographic reductions, in which every element of Y is either rational or an algebraic number with an explicitly given polynomial equation. Hence, a solution can be found accurate to 2^{-n} in time polynomial in n, as required (e.g., [45]).

A direct application of the above result that uses the matchgates already described for the nonstandard basis **b1** is Theorem 8.1. The reader may choose to look at section 8 next before proceeding to other sections.

The previous theorem also supports the following generalization.

COROLLARY 4.2. *For any matchgrid Ω with omittable nodes and having weighted graph G, if in the definition of signature of a matchgate PerfMatch is replaced by MatchSum, and this is inherited in the definitions of valG and valR, then*

$$\text{Holant}(\Omega) = \text{MatchSum}(G).$$

Note, however, that the only case we know in which this can be exploited for polynomial time algorithms is when all the omittable nodes are on the outer face and we can invoke Theorem 3.3, as we do in Theorem 9.6.

5. Signatures of planar matchgates. In this section we shall give a more systematic treatment of generators and recognizers.

Consider a graph G with three external nodes numbered 1, 2, 3. For each choice of $i, j, k \in \{0, 1\}$ let u_{ijk} equal the PerfMatch polynomial of G when nodes 1, 2, 3 are deleted, respectively, according to whether i, j, k equal 1 or not. Thus u_{111} denotes PerfMatch(G'), where G' is G with all three external nodes removed. Note that for a generator or recognizer the definition of the standard signature u given in section 4 implies that u equals the 8-vector $(u_{000}, u_{001}, u_{010}, u_{011}, u_{100}, u_{101}, u_{110}, u_{111})$.

For a given basis **b** we denote by $\{\mathbf{b}_{ijk} | i, j, k \in \{0, 1\}\}$ the eight possible external 8-vectors $x_1 \otimes x_2 \otimes x_3$, where x_1, x_2, x_3 range over $\{n, p\}$, respectively. Thus $\mathbf{b}_{010} = n \otimes p \otimes n$ will denote the basis vector n at inputs 1 and 3 and the basis vector p at input 2. The (r, s, t)-element of the 8-vector $\mathbf{b}_{ijk}$ will be denoted by $(\mathbf{b}_{ijk})_{rst}$ and will represent in $x_1 \otimes x_2 \otimes x_3$ the product of the rth component of x_1, the sth component of x_2, and the tth component of x_3 for $r, s, t \in \{0, 1\}$. Thus $(\mathbf{b}_{010})_{110}$ will equal $n_1 p_1 n_0$, for example.

For the special case of the standard basis $n = (1, 0), p = (0, 1)$ clearly the (r, s, t)-element of vector $\mathbf{b}_{ijk}$ will equal 0 unless $r = i$, $s = j$, and $t = k$, in which case it will equal 1.

Let us first consider generators. Suppose that G has standard signature u, and for all $\{i, j, k\} \in \{0, 1\}^3$

$$u_{ijk} = \sum q_{rst} (\mathbf{b}_{rst})_{ijk} \tag{5.1}$$

for some vector of numbers q where summation is over all $\{r, s, t\} \in \{0, 1\}^3$. Then we say that G *generates signature q with respect to basis* **b**. Note that if G has no omittable nodes then it is either even or odd and hence either the even or the odd four elements of $\{0,1\}^3$ have zero values for u_{ijk}.

Let us now consider recognizers. Suppose that G has standard signature $\hat{u}$, and that when the 8-vector $\mathbf{b}_{ijk}$ for some $\{i,j,k\} \in \{0,1\}^3$ is input to G then G evaluates to $\hat{q}_{ijk}$. Then

$$\hat{q}_{ijk} = \sum \hat{u}_{rst}(\mathbf{b}_{ijk})_{rst} \tag{5.2}$$

must hold, where summation is over $\{r,s,t\} \in \{0,1\}^3$. We then say that G *recognizes signature* $\hat{q}$ *over basis* $\mathbf{b}$. Again, if G has no omittable nodes then it is either even or odd and hence either the even or the odd four elements of $\{0,1\}^3$ have zero values for $\hat{u}_{rst}$.

PROPOSITION 5.1. *A gate G with standard signature equal to u will generate and recognize u with respect to the standard basis.*

Proof. This is immediate from the definition of generators and recognizers, and the fact observed above that for the standard basis $(\mathbf{b}_{ijk})_{rst} = \delta_{ir}\delta_{js}\delta_{kt}$ where δ is the Dirac delta function. ∎

For any basis $\mathbf{b}$ and matchgate, whether a generator or recognizer, one can define the signature of the matchgate with respect to the basis to be the vector q that it generates according to relation (5.1), or the vector q that it recognizes according to relation (5.2) above. Thus if the matchgate has arity m, then its signature with respect to $\mathbf{b}$ is a vector of length 2^m. We will denote it, for the $m = 3$ case, typically by $(q_{000}, q_{001}, \ldots, q_{111})$. The standard signature defined in section 3 is just the signature with respect to the standard basis. When we discuss a basis we need to be clear about which basis is involved. However, signatures that differ from each other by a nonzero constant factor can be treated as equivalent since their contribution to the PerfMatch or MatchSum polynomials of any overall matchgrid differ by just that constant multiple.

If the arity m gate is symmetric in its inputs and outputs, then we can define its *symmetric signature with respect to basis* $\mathbf{b}$ to be the vector $[S_0, S_1, \ldots, S_m]$, where S_i is equal to all the elements of the ordinary signature that are indexed by $\{0,1\}^m$ patterns with i occurrences of 1. For example, the gate in Figure 1 is symmetric. With respect to basis **b1** it has ordinary signature $(q_{00}, q_{01}, q_{10}, q_{11}) = (1,1,1,0)$ and symmetric signature $[S_0, S_1, S_2] = [1,1,0]$. The gate in Figure 2 has symmetric instances, such as those where all the weights $w_i = 1$ and $m = 3$, say, in which case the symmetric signature is $[-3,1,0,0]$ with respect to the same basis. We shall use round parentheses for signatures and square parentheses for the abbreviated symmetric version.

6. Realizable signatures for the standard basis. With respect to the standard basis we can characterize the standard signatures that are realizable with planar matchgates of arity up to four. We first note that in any such signature either the odd or the even components must be zero depending on the parity of the number of nodes in the matchgate. Propositions 6.1 and 6.2 therefore show that for arities 2 and 3 all signatures are realizable up to this basic constraint.

PROPOSITION 6.1. *For all F and all $x, y \in F$ there exist matchgates with arity 2 and standard signatures $(u_{00}, u_{01}, u_{10}, u_{11}) = (x,0,0,y)$ and $(0,x,y,0)$.*

Proof. The matchgates of Figure 4, with external nodes $\{1,2\}$, suffice. ∎

PROPOSITION 6.2. *For all F and all $x, y, z,\ t \in F$ there exist matchgates with arity 3 and standard signatures $(u_{000}, u_{001}, u_{010}, u_{011}, u_{100}, u_{101}, u_{110}, u_{111}) = (t,0,0,z,0,y,x,0)$ and $(0,x,y,0,z,0,0,t)$.*

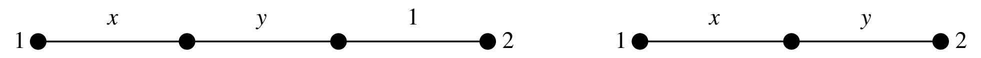


FIG. 4. *Two arity two gates with input/output gates* $\{1, 2\}$.

Proof. For the odd case $(0, x, y, 0, z, 0, 0, t)$ consider Figure 5. Clearly if $t \neq 0$ the left-hand figure has standard signature $(0, x, y, 0, z, 0, 0, t)$ and solves the problem. If $t = 0$ we use the right-hand diagram.

For the even case $(t, 0, 0, z, 0, y, x, 0)$ the signature of the left part of Figure 6 is $(ax+by+cz, 0, 0, z, 0, y, x, 0)$ and therefore solves the problem for all values of x, y, z, t by appropriate choice of a, b, c, unless $x = y = z = 0$ and $t \neq 0$. In that exceptional case we use the right-hand diagram. □

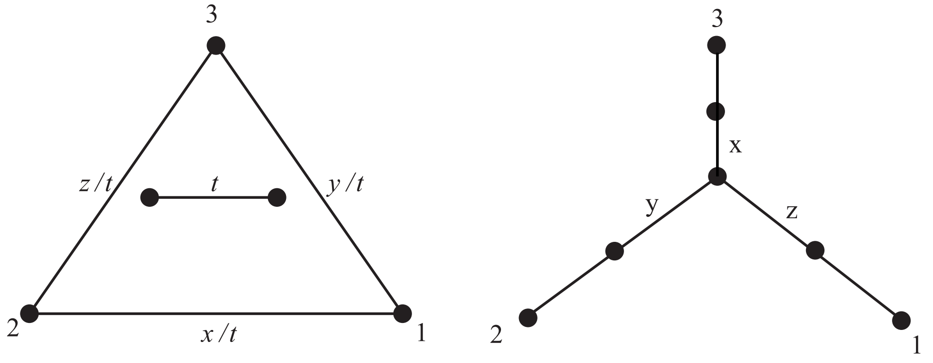


FIG. 5. *Arity three gates for nonzero odd components.*

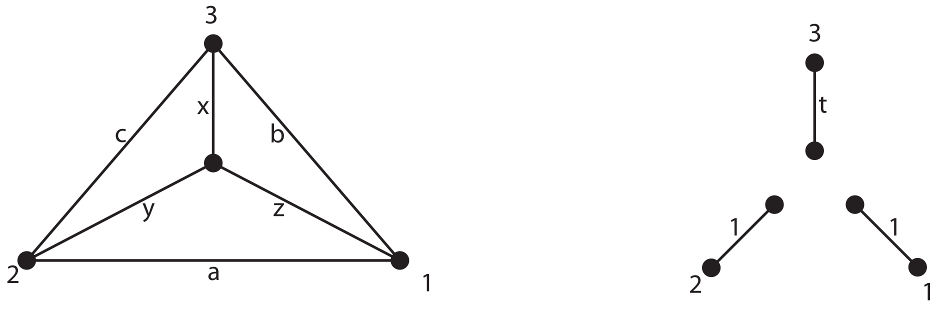


FIG. 6. *Arity three gates for nonzero even components.*

In general we shall refer to the elements of a signature being *even* or *odd* according to whether their index has an even or odd number of 1's. Thus, for example, u_{1010} and u_{0000} are even while u_{0100} and u_{0111} are odd.

PROPOSITION 6.3. *Suppose the elements of the standard signature are represented by* u_{ijkl} *for* $i, j, k, l \in \{0, 1\}$. *For any* F *it is possible to realize by matchgates with arity* 4 *any standard signature such that*

(i) $u_{0000}u_{1111} - u_{0011}u_{1100} + u_{0101}u_{1010} - u_{0110}u_{1001} = 0$, $u_{1111} \neq 0$, *and all the odd elements are zero, or*

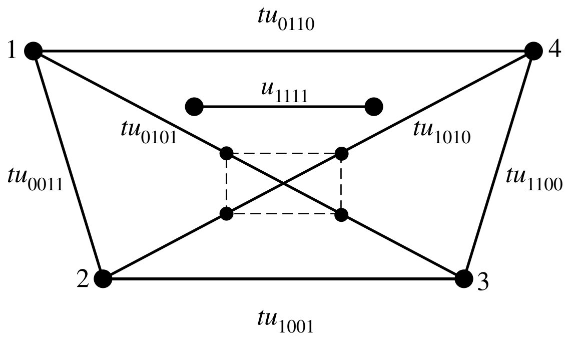


FIG. 7. *An arity four matchgate, where $t = 1/u_{1111}$.*

(ii) $u_{1000}u_{0111} - u_{1011}u_{0100} + u_{1101}u_{0010} - u_{1110}u_{0001} = 0$, $u_{0111} \neq 0$, *and all the even elements are zero.*

Proof. We consider (i) first. The algebraic relationship is the first matchgate identity from [56] and we follow the construction from there shown in Figure 7. First, ignoring the central square we have a nonplanar matchgate. It is easy to see that this matchgate does have the desired values for the seven components $u_{1111}, u_{0110}, u_{1001}$, $u_{0011}, u_{1100}, u_{0101}$, and u_{1010} of the signature. Now if we could somehow simulate the crossing edges $(1,3)$ and $(2,4)$ by a planar graph so as to create a change in sign, the value of the eighth component u_{0000} for this matchgate would be the following:

$$tu_{0110}u_{1001} + tu_{0011}u_{1100} - tu_{0101}u_{1010}.$$

If we substitute $t = 1/u_{1111}$, then we would have the claimed relationship (i). Now to make the graph planar and to simulate the -1 factor, we replace the crossing edges by the planar graph shown in Figure 8. In Figure 8 if we substitute $a = 1$, $b = i$, $c = d = -1/2$, and $e = \sqrt{i}$, where $i^2 = -1$, then nonzero contributions from PerfMatch occur for just the four combinations of each of $(1,3)$ and $(2,4)$ being present or not in Figure 7. Each combination comes with a factor of $+1$, except the one that has both crossing edges present in Figure 7, which contributes a factor of -1 as required. This concludes the construction for the field of complex numbers. (It can also be verified that the same graph with appropriate ± 1 weights will have factors -1, 2, 2, and 4, which can be normalized to -1, 1, 1, and 1, respectively, by appending appropriate graphs at the external nodes. Hence the construction applies for all fields F. Note that the signature of any planar matchgate has to satisfy algebraic identities similar to those of the character [57].)

In order to obtain part (ii) we simply append an extra edge weighted 1 at input 1 and call the other endpoint of the new edge the new input node 1. This transformation leaves the elements of the signature unchanged, except that they are renamed by the process of flipping the first bit of the index in each term e.g., u_{0000} becomes u_{1000}. □

We note that the constraints $u_{1111} \neq 0$ and $u_{0111} \neq 0$ can be eliminated in the following sense. If any of the sixteen components is nonzero, then, by the method of the last paragraph, one can flip bits so that the nonzero entry is moved to the 1111 or 0111 position, and the relation (i) or (ii) holds for the corresponding renaming of the elements.

PROPOSITION 6.4. *For all F and any arity m and any $S_0 \in F$ there is a matchgate with standard symmetric signature $[S_0, \ldots, S_m]$, where $S_1 = \cdots = S_m = 0$.*

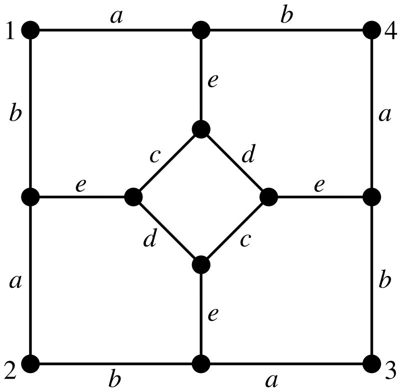


FIG. 8. *An arity four planar matchgate that is used to simulate the crossover in Figure* 7. *The substitution* $a = 1$, $b = i$, $c = d = -1/2$, $e = \sqrt{i}$ *suffices where* $i^2 = -1$.

Proof. The gate consists of $2m$ nodes $v_1, \ldots, v_m$, $u_1, \ldots, u_m$ and m edges (v_i, u_i), where $u_1, \ldots, u_m$ are the output nodes. All the edges have weight one, except for one which has weight S_0. □

7. Realizable signatures for arity two matchgates.

Relations (5.1) and (5.2) in section 5 relate the signatures realizable by an arbitrary basis to those realizable by the standard basis. In section 6 we characterized the signatures that are realizable by the standard basis for gates of arity up to four. In this section we shall spell out some consequences for signatures with respect to arbitrary bases that are realizable by gates of arity 2. These gates will be invoked in several places in the various algorithms described in later sections.

For ease of notation we shall consider the basis to be $\mathbf{b} = [(a, b), (c, d)]$ so that $\mathbf{b}_{00} = (a, b) \otimes (a, b)$, $\mathbf{b}_{01} = (a, b) \otimes (c, d)$, $\mathbf{b}_{10} = (c, d) \otimes (a, b)$, and $\mathbf{b}_{11} = (c, d) \otimes (c, d)$.

Relation (5.1) then describes the following requirements on a generator to have signature $(q_{00}, q_{01}, q_{10}, q_{11})$ with respect to $\mathbf{b}$:

$u_{00} = a^2 q_{00} + acq_{01} + acq_{10} + c^2 q_{11}$,
$u_{01} = abq_{00} + adq_{01} + bcq_{10} + cdq_{11}$,
$u_{10} = abq_{00} + bcq_{01} + adq_{10} + cdq_{11}$, and
$u_{11} = b^2 q_{00} + bdq_{01} + bdq_{10} + d^2 q_{11}$.

By Proposition 6.1 any standard signature is possible as long as either $u_{00} = u_{11} = 0$ or $u_{01} = u_{10} = 0$. Hence there exist generators with signature $(q_{00}, q_{01}, q_{10}, q_{11})$ with respect to basis $\mathbf{b}$ if either

$$a^2 q_{00} + acq_{01} + acq_{10} + c^2 q_{11} = 0 \quad \text{and} \quad b^2 q_{00} + bdq_{01} + bdq_{10} + d^2 q_{11} = 0$$

or

$$abq_{00} + adq_{01} + bcq_{10} + cdq_{11} = 0 \quad \text{and} \quad abq_{00} + bcq_{01} + adq_{10} + cdq_{11} = 0.$$

PROPOSITION 7.1. *For the basis* $\mathbf{b2} = [(1, 1), (1, -1)]$ *for any* $x, y \in F$ *there is a generator for* $(x, y, y, x) = [x, y, x]$.

Proof. The second of the two cases above gives $q_{00} - q_{01} + q_{10} - q_{11} = 0$ and $q_{00} + q_{01} - q_{10} - q_{11} = 0$. Clearly these will be satisfied if $q_{00} = q_{11}$ and $q_{01} = q_{10}$. □

PROPOSITION 7.2. *For the basis* $\mathbf{b2} = [(1, 1), (1, -1)]$ *for any* $x, y \in F$ *there is a generator for* $(x, y, -y, -x)$.

Proof. The first of the two cases above gives $q_{00}+q_{01}+q_{10}+q_{11}=0$ and $q_{00}-q_{01}-q_{10}+q_{11}=0$. Clearly these will be satisfied if $q_{00}+q_{11}=0$, and $q_{01}+q_{10}=0$. □

Moving on to recognizers, we note that the requirements for a recognizer to have signature $(q_{00},q_{01},q_{10},q_{11})$ are given by relation (5.2):

$q_{00}=a^2u_{00}+abu_{01}+abu_{10}+b^2u_{11}$,
$q_{01}=acu_{00}+adu_{01}+bcu_{10}+bdu_{11}$,
$q_{10}=acu_{00}+bcu_{01}+adu_{10}+bdu_{11}$, and
$q_{11}=c^2u_{00}+cdu_{01}+cdu_{10}+d^2u_{11}$.

By Proposition 6.1 any standard signature is possible as long as either $u_{00}=u_{11}=0$ or $u_{01}=u_{10}=0$. Hence there exist recognizers with signature $(q_{00},q_{01},q_{10},q_{11})$ with respect to basis **b** of the two forms

$$(abu_{01}+abu_{10},\ adu_{01}+bcu_{10},\ bcu_{01}+adu_{10},\ cdu_{01}+cdu_{10})$$

and

$$(a^2u_{00}+b^2u_{11},\ acu_{00}+bdu_{11},\ acu_{00}+bdu_{11},\ c^2u_{00}+d^2u_{11}).$$

PROPOSITION 7.3. *If* $\mathbf{b2}=[(1,1),(1,-1)]$ *is a basis for field* F *then for any* $x,y\in F$, *there is a recognizer for* $(x,y,y,x)=[x,y,x]$.

Proof. The second case above gives signature $(u_{00}+u_{11},u_{00}-u_{11},u_{00}-u_{11},u_{00}+u_{11})$. □

PROPOSITION 7.4. *If* $[(a,b),(c,d)]$ *is a basis, then there is a recognizer for* $(0,ad-bc,bc-ad,0)$.

Proof. The proof follows from the first case above if $u_{01}=1$ and $u_{10}=-1$. □

PROPOSITION 7.5. *If* $[(a,b),(c,d)]$ *is a basis, then there is a recognizer for* $(a^2+b^2,ac+bd,ac+bd,c^2+d^2)$.

Proof. The proof follows from the second case with $u_{00}=1$ and $u_{11}=1$. □

PROPOSITION 7.6. *If* $[(a,b),(c,d)]$ *is a basis, then there is a recognizer for* $(a^2-b^2,ac-bd,ac-bd,c^2-d^2)$.

Proof. The proof ollows from the second case with $u_{00}=1$ and $u_{11}=-1$. □

8. The basis $b1=[(1,-1),(1,0)]$. We shall now apply our method to the problem of matchings—not necessarily perfect—in planar graphs. This is also known as the monomer-dimer problem. Considerable efforts had been expended in attempts to reduce it to the planar perfect matching problem. The lack of success achieved was explained by the work of Jerrum [26, 27] who showed that this counting problem was #P-complete. Subsequently Vadhan [51] showed that it remained #P-complete even when the planar graph was bipartite, and its degree was restricted to 6. If the degree is restricted to 2, then the graph consists of a set of cycles and the problem is easily solvable. Any class that allows higher degrees is a natural candidate for #P-compleness. However, we can show that the following such problem is computable in polynomial time.

THEOREM 8.1. *There is a polynomial time algorithm for #X-MATCHINGS.*

Proof. Consider a given planar weighted graph $H=(V,E,W)$, where V has bipartition $V1$, $V2$, where every node $v\in V1$ has degree 2 and every node $v\in V2$ has some arbitrary degree $\deg(v)$. We construct a matchgrid Ω_H over **b1** by replacing each $V1$ node with the generator matchgate of Proposition 4.1, replacing each $V2$ node with the recognizer matchgate of Proposition 4.2, and, for each edge (u,v) in H by having a connecting edge joining an output of the generator for u to an input of the recognizer for v so as to preserve planarity. The edge in the recognizer that is adjacent to this connecting edge will have the same weight w_i as the edge (u, v) has in H.

Now the Holant was defined as

$$\text{Holant}(\Omega_H) = \sum_{x \in X} \left[\prod_{1 \le j \le g} \text{valG}(B_j, x) \right] \left[\prod_{1 \le i \le r} \text{valR}(A_i, x) \right],$$

where j ranges over all the generators, i over all the recognizers, and x over all possible tensor products of the basis elements. But each generator has arity two and generates $n \otimes n$, $n \otimes p$, $p \otimes n$, and $p \otimes p$ with weights 1, 1, 1, and 0, respectively. Hence the nonzero contributions to the Holant will come from edge sets of H such that at most one edge from the set is adjacent to each $V1$ node. But the matchgates at the $V2$ nodes are defined so that $\text{valR}(A_i, x)$ is

(i) 0 if there is more than one edge incident,
(ii) w_i if there is exactly one, and its weight is w_i, and
(iii) $-(w_1 + \cdots + w_k)$ if there are no incident edges.

Hence the value of the Holant is the sum over all matchings E' of H of the mass of E' defined as follows. The mass of E' is the product of the weights of all the edges that are present in it and also of the value of $-(w_1 + \cdots + w_k)$ for every $V2$ node that is not saturated by the matching. Hence, by virtue of Theorem 4.1 and Corollary 4.1, this mass can be computed in polynomial time, and, in fact, in NC2. □

9. The basis *b*2 = [(1, 1), (1, −1)].

In this section we study the basis $\mathbf{b2} = [(1, 1), (1, -1)]$, which has a remarkable range of capabilities. We shall assume that field F does not have characteristic two, since then $\mathbf{b2}$ would have just one distinct element. We first note that by Propositions 7.1 and 7.3, the arity 2 symmetric signature $[x, y, x]$ can be realized for any x, y, both as a generator and as a recognizer. Thus equality has weight x and inequality has weight y. The case $x = 0$ gives inequality gates and the case $y = 0$ equality gates. The arity one signature $[x, x]$ is also realizable, by 2-node matchgates, but the arity one constants [1, 0] and [0, 1] are not—they would require omittable nodes.

If we have a generator over this basis and join to its outputs equality recognizers, then we get a recognizer gate with the same signature as the original generator. Similarly we can convert arbitrary recognizers to generators with the same signature by appending generator equality gates. *Hence for this basis* **b2** *the signatures that can be realized by generators are exactly the same as those that can be realized by recognizers.*

For arity three we shall now enumerate the eight possible combinations of basis elements for the inputs, namely, $\mathbf{b2}_{000}, \ldots, \mathbf{b2}_{111}$, rewrite each as an 8-vector of coefficients with respect to the standard basis, and group them according to some semantics:

THREE POSITIVES								
$(1,-1)\otimes(1,-1)\otimes(1,-1)$:	(1	–1	–1	1	–1	1	1	–1)
ZERO POSITIVES								
$(1,1)\otimes(1,1)\otimes(1,1)$:	(1	1	1	1	1	1	1	1)
ONE POSITIVE								
$(1,-1)\otimes(1,1)\otimes(1,1)$:	(1	1	1	1	–1	–1	–1	–1)
$(1,1)\otimes(1,-1)\otimes 1,1)$:	(1	1	–1	–1	1	1	–1	–1)
$(1,1)\otimes(1,1)\otimes(1,-1)$:	(1	-1	1	–1	1	–1	1	–1)
TWO POSITIVE								
$(1,-1)\otimes(1,1)\otimes(1,-1)$:	(1	–1	1	–1	–1	1	–1	1)
$(1,1)\otimes(1,-1)\otimes(1,-1)$:	(1	–1	–1	1	1	–1	–1	1)
$(1,-1)\otimes(1,-1)\otimes(1,1)$:	(1	1	–1	–1	–1	–1	1	1)
SUMS:								
0 OR 3 POSITIVES:	(2	0	0	2	0	2	2	0)
1 OR 2 POSITIVES:	(6	0	0	–2	0	–2	–2	0)

By taking linear combinations of the rows as specified by relation (5.1) we can determine which combinations are realizable standard signatures. By Proposition 6.2 it is sufficient in the arity three case for a standard signature that either all the odd elements, or all the even elements, be zero.

It is clear that if we add the THREE POSITIVES and the ZERO POSITIVES vectors we get an all-even signature (2, 0, 0, 2, 0, 2, 2, 0). It follows that the symmetric signature [1, 0, 0, 1] is realizable for the basis **b2**. Similarly adding the remaining six vectors also gives an all-even vector, and hence the symmetric signature [0, 1, 1, 0] is also realizable. Further, if we add x times the first two vectors to y times the last six we still get an all-even vector. Hence for all $x, y \in F$, the symmetric signature $[x, y, y, x]$ is realizable.

Theorem 9.1. *There is a polynomial time algorithm for #PL-*3*-NAE-ICE.*

Proof. We represent each degree 3 node of the given graph G by a recognizer matchgate with symmetric signature [0, 1, 1, 0] over **b2**, i.e., the NOT-ALL-EQUAL or NAE gate. For degree 2 nodes we have a recognizer for [0, 1, 0] from Proposition 7.3. For each edge we will have a generator matchgate with symmetric signature [0, 1, 0] from Proposition 7.1. We will have connecting edges between the outputs of the generators and inputs of the recognizers as specified by G. If p on a connecting edge of a recognizer gate represents the orientation toward that gate, and an n an orientation away from it, then clearly each edge of G will be given a consistent orientation by virtue of the binary inequality generator gate [0, 1, 0], which ensures that its two outputs carry opposite basis elements. Further, the recognizer gates will ensure that either one or two of the edges are directed toward it. It follows that the Holant of the given matchgrid will equal the desired value of #PL-3-NAE-ICE. □

Theorem 9.2. *There is a polynomial time algorithm for #PL-*3*-*(1,1)*-CYCLE-CHAIN.*

Proof. Suppose we are given graph G as input to the (1,1) cycle-chain problem. We shall represent each node by a recognizer for $[0, 1, 1, 0]$. We represent each edge of G by a generator for $[1, 0, 1]$. Now if a p generated by a generator signifies that the corresponding edge of G is in the cycle-chain cover, then clearly the edges of

G will have a consistent such association by virtue of the [1, 0, 1] generators. But the recognizers will ensure that either one or two edges of G incident to any one vertex are labeled p. It follows that there is a one-to-one correspondence between labelings of the edges of G by $\{n, p\}$ such that the edges labeled by p form a cycle-chain cover and contributions of 1 to the Holant of the constructed matchgrid. The result follows. □

THEOREM 9.3. *There is a polynomial time algorithm for PL-NODE-BIPARTITION.*

Proof. Suppose we are given graph G as input to the PL-NODE-BIPARTITION problem. We shall represent each node by a recognizer for $[x, y, y, x]$ or $[x, y, x]$, depending on whether the degree is 3 or 2, where x, y are variables to be given various values. (Any node of degree 1 can be simply deleted.) Each edge we represent by a generator for $[0, 1, 0]$. Then as in the proof of Theorem 9.1 we can interpret nonzero contributions to the Holant as orientations of G. Nodes that have all edges directed toward them (sinks) or all edges directed away from them (sources) will give a contribution of x to the Holant, and those that are neither sources nor sinks will have a contribution of y. Now if we fix $y = 1$ then the Holant will be a polynomial $SS(x)$, where the coefficient of x^i will be the number of orientations of the edges of G that have exactly i nodes as either sources or sinks.

Now it is easy to verify that the largest i for which the coefficient of x^i in $SS(x)$ is nonzero is the maximum number of nodes that a bipartite graph can have that is obtained by deleting nodes and incident edges from G. In one direction, if there is an orientation with i sources and sinks, then the graph induced by the nodes that are sources and sinks in G must be bipartite. In the reverse direction, if we have a bipartite subgraph in G where the nodes have bipartition $V1'$ and $V2'$, then we can define an orientation of G where all the nodes $V1'$ are sources and all the nodes $V2'$ sinks, and the orientation of any edge not incident to $V1'$ or $V2'$ can be arbitrary.

Now by giving x any fixed value we can compute the Holant for that value and hence obtain the value of $SS(x)$. By doing this for $|V| + 1$ distinct values of x and performing polynomial interpolation on the $|V| + 1$ values obtained, we can compute all the coefficients of $SS(x)$. The largest i such that the coefficient of x^i in $SS(x)$ is nonzero will give the minimum number $|V| - i$ of nodes whose removal leads to a bipartite graph. □

The following folds in the results for gates with 1, 2, and 3 inputs described above, with some result for gates with 4 inputs detailed below, and the equality gate for any number of inputs.

THEOREM 9.4. *For matchgrids where each matchgate is one of* $[x, x]$, $[x, y, x]$, $[x, y, y, x]$, $[1, 0, 0, 0, 1]$, $[1, 0, -1, 0, 1]$, $[0, 1, 0, -1, 0]$, $[0, 1, \pm\sqrt{2}, 1, 0]$ *, in the case* $2x^2 = yw + y^2$ $[w, x, y, x, w]$, *and* $[1, 0, \ldots 0, 1]$ *for any arity, the Holant can be computed in polynomial time. Here different matchgates may have different values of* $x, y, w \in F$.

Proof. For arities one, two, and three, we have already established that $[x, x]$, $[x, y, x]$, and $[x, y, y, x]$ are realizable.

For arity four we shall enumerate the sixteen possible combinations of basis elements for the inputs, namely, $\mathbf{b2}_{0000}, \ldots, \mathbf{b2}_{1111}$, rewrite each as a 16-vector of coefficients with respect to the standard basis, and group them according to some semantics, as we did for arity three. By taking linear combinations of the rows as specified by relation (5.1), we can again determine which combinations are realizable standard signatures. By Proposition 6.3 it is sufficient in the arity four case for a standard signature that all the odd elements, or all the even elements, be zero, provided in addition that the polynomial relation stated there holds among the eight remaining elements.

ZERO POSITIVES	
(1,1)⊗(1,1)⊗(1,1)⊗(1,1):	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
FOUR POSITIVES	
(1,-1)⊗(1,-1)⊗(1,-1)⊗(1,-1):	(1 -1 -1 1 -1 1 1 -1 -1 1 1 -1 1 -1 -1 1)
CROSSINGS	
(1,-1)⊗(1,1)⊗(1,-1)⊗(1,1):	(1 1 -1 -1 1 1 -1 -1 -1 -1 1 1 -1 -1 1 1)
(1,1)⊗(1,-1)⊗(1,1)⊗(1,-1):	(1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1)
THE OTHER FOUR TWO POSITIVES CASES	
(1,1)⊗(1,1)⊗(1,-1)⊗(1,-1):	(1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1)
(1,-1)⊗(1,-1)⊗(1,1)⊗(1,1):	(1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1)
(1,1)⊗(1,-1)⊗(1,-1)⊗(1, 1):	(1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1)
(1,-1)⊗(1,1)⊗(1,1)⊗(1,-1):	(1 -1 1 -1 1 -1 1 -1 -1 1 -1 1 -1 1 -1 1)
ONE POSITIVE	
(1,1)⊗(1,1)⊗(1,-1)⊗(1,1):	(1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1)
(1,-1)⊗(1,1)⊗(1,1)⊗(1,1):	(1 1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1)
(1,1)⊗(1,1)⊗(1,1)⊗(1,-1):	(1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1)
(1,1)⊗(1,-1)⊗(1,1)⊗(1,1):	(1 1 1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1)
THREE POSITIVES	
(1,1)⊗(1,-1)⊗(1,-1)⊗(1,-1):	(1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1)
(1,-1)⊗(1,-1)⊗(1,1)⊗(1,-1):	(1 -1 1 -1 -1 1 -1 1 -1 1 -1 1 1 -1 1 -1)
(1,-1)⊗(1,-1)⊗(1,-1)⊗(1,1):	(1 1 -1 -1 -1 -1 1 1 -1 -1 1 1 1 1 -1 -1)
(1,-1)⊗(1,1)⊗(1,-1)⊗(1,-1):	(1 -1 -1 1 1 -1 -1 1 -1 1 1 -1 -1 1 1 -1)
SUMS:	
0 OR 4 POSITIVES	(2 0 0 2 0 2 2 0 0 2 2 0 2 0 0 2)
TWO POSITIVES	(6 0 0 -2 0 -2 -2 0 0 -2 -2 0 -2 0 0 6)
ONE POSITIVE	(4 2 2 0 2 0 0 -2 2 0 0 -2 0 -2 -2 -4)
THREE POSITIVES	(4 -2 -2 0 -2 0 0 2 -2 0 0 2 0 2 2 -4)

Each 4-output matchgate will have standard signature $(u_{0000}, \ldots, u_{1111})$. Each gate will be either even or odd and will have at most eight of the elements of their signature nonzero. For convenience we shall here represent the signature of an even gate by the 8-vector $(u_{0000}, u_{0011}, u_{0101}, u_{0110}, u_{1001}, u_{1010}, u_{1100}, u_{1111})$ and the signature of an odd gate by the 8-vector $(u_{0001}, u_{0010}, u_{0100}, u_{0111}, u_{1000}, u_{1011}, u_{1101}, u_{1110})$.

(i) Signature $[1, 0, 0, 0, 1]$: Adding the signatures for the two cases **(0 positives) + (4 positives)** gives for the even case the 8-vector $(2, 2, 2, 2, 2, 2, 2, 2)$ which is feasible by Proposition 6.3(i).

(ii) Signature $[1, 0, -1, 0, 1]$: Forming the linear combination for the eight cases ***z*(2 positives) + (0 or 4 positives)** gives for the even case the 8-vector $(6z+2, 2-2z, 2-2z, 2-2z, 2-2z, 2-2z, 2-2z, 6z+2)$. By Proposition 6.3(i) this is realizable if $(6z + 2) * (6z + 2) = (2 - 2z)(2 - 2z)$, or $36z^2 + 24z + 4 = 4z^2 - 8z + 4$, or $32z^2 + 32z = 0$, or $z = -1$.

(iii) Signature $[0, 1, 0, -1, 0]$: Forming the linear combination for the eight cases **(1 positive) − (3 positives)** gives for the odd case the 8-vector $(4, 4, 4, -4, 4, -4, -4, -4)$. By Proposition 6.3(ii) this is realizable since $-16 + 16 + 16 - 16 = 0$.

(iv) Signature $[0, 1, \pm\sqrt{2}, 1, 0]$: Forming the linear combination for the fourteen cases **(1 positive) + (3 positives) + *y*(2 positives)** gives for the even case the 8-vector $(8 + 6y, -2y, -2y, -2y, -2y, -2y, -2y, -8 + 6y)$. Therefore it is sufficient that $(8 + 6y)(-8 + 6y) = 4y^2$, or $-64 + 36y^2 = 4y^2$, or $y = \sqrt{2}$ or $y = -\sqrt{2}$.

(v) Signature $[w, x, y, x, w]$: Forming the linear combination for the sixteen cases ***w*(0 positives) + *w*(4 positives) + *x*(1 positive) + *x*(3 positives) + *y*(2 positives)** gives for the even case the 8-vector $(2w + 6y + 8x, 2w -$

$2y, 2w-2y, 2w-2y, 2w-2y, 2w-2y, 2w-2y, 2w+6y-8x)$, for which any y, x, and w with $2x^2 = yw + y^2$ will suffice.

Note that relation (v) generalizes relations (i), (ii), and (iv).

Finally, we note that equality gates of any arity m can be obtained by chaining together $m-2$ ternary equality gates $[1, 0, 0, 1]$ using the equality recognizers of Proposition 7.3. □

THEOREM 9.5. *There is a polynomial time algorithm for#PL-3-NAE-SAT.*

Proof. The construction follows that for Theorem 9.1 except that for NAE nodes we have recognizers with symmetric signatures $[0, 1, 1, 0]$, and for variable nodes we have recognizers for $[1, 0, \ldots, 0, 1]$ gates of the same arity as the number of clauses in which the variable appears. Further, if a variable occurrence is negated we have a $[0, 1, 0]$ generator along the edge that joins the variable recognizer and the NAE recognizer, and if the variable occurrence is not negated then we have $[1, 0, 1]$. □

THEOREM 9.6. *There is a polynomial time algorithm for ⊕PL-EVEN-LIN2.*

Proof. The construction follows that for Theorem 9.1 with some exceptions. First we note that any equation of even length more than four can be reduced to a set of equations all of length four by the introduction of new variables. For example, $z_1 + \cdots + z_6 = 1$ becomes the two equations $z_1 + z_2 + z_3 + y = 0$ and $y + z_4 + z_5 + z_6 = 1$. Now each equation of length four is simulated by a $[1, 0, -1, 0, 1]$ or a $[0, 1, 0, -1, 0]$ gate depending on whether the constant term of the equation being simulated is 0 or 1. For length 2 we use $[1, 0, 1]$ and $[0, 1, 0]$, respectively. The boundary conditions that fix the values of variables can be realized by using 2-node matchgates with one omittable node as shown in Figure 3. We then invoke Corollary 4.2 and Theorem 3.3.

For each original noncompulsory equation we pick an arbitrary variable occurrence in it and simulate it "possibly being faulty" by having as the corresponding link between its variable and equation recognizers a generator for $[1, x, 1]$ if the variable occurs positively and $[x, 1, x]$ if it occurs negated. For all other occurrences of variables the corresponding link is a generator for $[1, 0, 1]$ or $[0, 1, 0]$ as appropriate. The Holant will then be a polynomial in x. The coefficient of x^i will arise from "solutions" of the equations where exactly i variable occurrences, all in distinct noncompulsory equations, have their bits inverted. In other words they arise from solutions that satisfy all but exactly i of the noncompulsory equations. Since each solution contributes ± 1 to the Holant, the result follows. □

We note that the generating function of the Ising problem, or #PL-CUT, is nothing other than the Holant when edges are replaced by $[1,\ y,\ 1]$ gates and nodes by $[1, 0, \ldots, 0, 1]$ gates over **b2**. Hence this offers yet another treatment of the Ising problem for planar structures. In the reverse direction this implies that algorithms based on just these gates can be derived also from #PL-CUT through classical reductions, essentially following our proofs here. The reader can verify that Theorems 9.1, 9.2, 9.3, and 9.5 are all in this category since $[1,\ y,\ y,\ 1]$ can be simulated by three $[1,\ y,\ 1]$ gates. However, if the problems solved in these theorems are generalized to allow the degree 4 gates permitted by Theorem 9.4, then polynomial time algorithms follow for some, perhaps less natural, problems for which no classical reduction to #PL-CUT is apparent.

To conclude this section we now summarize more explicitly what the holographic treatment of an instance G of #PL-CUT involves. The basis used will be **b2**. Each node of G of degree d will be replaced by a recognizer for $[1,\ 0, \ldots,\ 0,\ 1]$ which enforces equality on its d inputs. Clearly (Theorem 9.4) such a recognizer can be constructed by chaining together d-2 recognizers for $[1,\ 0,\ 0,\ 1]$ with generators for $[1,\ 0,\ 1]$. Each edge of G will be simulated by a generator for [1, y, 1]. The simulation consists of replacing the nodes and edges by these gates and joining up the corresponding pairs of input/output nodes of these gates by single edges. The coefficient of y^k in PerfMatch of the resulting weighted graph will give the answer to the #PL-CUT problem by virtue of the Holant theorem. The only technical facts that need to be verified are that for the basis **b2** generators for [1, y, 1] and [1, 0, 1], and recognizers for [1, 0, 0, 1] exist. Proposition 7.1 guarantees the former. Proposition 6.2 guarantees the latter in conjunction with the observation made at the start of the current section that the sum of "0 or 3 positives" over **b2** corresponds to an all even standard signature.

10. The basis b3 = [(1, 1), (1, −1), (1, 0)].

We now consider the problem PL-FO-2-COLOR and shall employ this basis **b3**.

Theorem 10.1. *There is a polynomial time algorithm for PL-FO-2-COLOR.*

Proof. Given a graph G we assume that all its nodes have degree 2 or 3. At nodes of degree 3 we place matchgates that generate

$$\begin{aligned}&(1,0)\otimes(1,1)\otimes(1,1)+(1,1)\otimes(1,0)\otimes(1,1)+(1,1)\otimes(1,1)\otimes(1,0)\\&+(1,0)\otimes(1,-1)\otimes(1,-1)+(1,-1)\otimes(1,0)\otimes(1,-1)+(1,-1)\otimes(1,-1)\otimes(1,0),\end{aligned}$$

and at nodes of degree 2 those that generate

$$\begin{aligned}&(1,0)\otimes(1,1)+(1,1)\otimes(1,0)\\&+(1,0)\otimes(1,-1)+(1,-1)\otimes(1,0).\end{aligned}$$

We note that all the nonzero terms are even and hence by Proposition 6.2 there are matchgates to generate them. In place of the edges of G we place recognizers that on input (a,b), (c,d) at their respective inputs have value $ac-bd$. The recognizer of Proposition 7.6 suffices.

We say that a node of G represents **0** if it generates $(1,1)$ in some direction and **1** if it generates $(1,-1)$ in some direction. In either case it directs its arrow along the edge on which it sends $(1,0)$ and away from itself.

Clearly the recognizer between two nodes that both represent **0** or both **1** will have value zero unless at least one of the nodes sends an arrow, i.e., $(1,0)$, to the recognizer, in which case its value will be 1. A recognizer between two nodes that represent **0** and **1**, respectively, will have value 2 if there are no arrows toward the recognizer, and 1 otherwise.

The Holant of this matchgrid will be nonzero iff PL-FO-2-COLOR has a solution. Each solution will be counted 2^k times where k is the number of edges in G that have no arrows and whose endpoints represent opposite values. □

11. General complexity-theoretic questions.

We regard the most important among the currently widely held conjectures of complexity theory to be (1) P ≠ NP, (2) P ≠ $\mathrm{P}^{\#\mathrm{P}}$, (3) P ≠ BPP, (4) P ≠ QBP, (5) P $\not\subseteq$ PolyLogSPACE, (6) P ≠ NC, (7) P ≠ PSPACE. We observe that a positive solution to the question $P^{\#P} =?NC$ would resolve all the above seven questions (the first six would be contradicted). (N.B. Regarding question (3) P = BPP is also widely conjectured.)

Now consider polynomial systems of the form

$$S(x)=\{E_1(x),E_2(x),\ldots,E_m(x)\},$$

where x stands for a set of variables $\{x_1,\ldots,x_n\}$ and $E_i(x)$ stands for a polynomial equation with coefficients from the integers. We shall say that such a system is *solvable* if it is satisfied by a set of complex numbers. In order that we may invoke Theorem 3.2, which is needed to ensure that the linear algebra computations be in polynomial time, we need such a system to be efficiently solvable. We say that a system is *efficiently solvable* if for that *one fixed* system there exists an algorithm that for some polynomial $p(n)$ and any $n > 1$ computes some solution to the system to n decimal places of accuracy within $p(n)$ Boolean operations. This is a weak requirement in that the size of the polynomial system can be regarded as a fixed constant. The requirement is only that the cost of computing the solutions to higher and higher accuracy is polynomial time bounded in the number of digits of the accuracy for that one fixed system. (It need not be polynomial time in the size of the system.)

In fact, it can be seen that solvable systems are always also efficiently solvable in our sense: Systems with finite numbers of solutions are efficiently solvable since they can be reduced by elimination to univariate polynomial solving. Further, systems with infinitely many solutions are also efficiently solvable by means of univariate representations [7, Theorem 4.12], [2, Algorithm 11.60]. It is well known that univariate polynomials are efficiently solvable (e.g., [45]).

We now observe that holographic methods can be viewed as providing constructions of natural systems S such that

$$S(x) \text{ is solvable } \Rightarrow \mathrm{P}^{\#\mathrm{P}} = \mathrm{NC2}. \tag{11.1}$$

More particularly, for any one *formulation* F (specifiable by appropriate local constraints) of a combinatorial problem for which the counting problem $\#F$ is #P-complete, and any basis size k, and gate size g, we shall construct a polynomial system $S_{F,k,g}$ such that $S_{F,k,g}$ is solvable iff there is a simple holographic reduction from $\#F$ to planar PerfMatch using basis size k and gate size g.

We can then define S_F to be the family of polynomial systems $\{S_{F,k,g} \mid k = 1, 2, \dots; g = 1, 2, \dots\}$ and make the final claim:

$$\text{Some member of } S_F \text{ is solvable } \Rightarrow \mathrm{P}^{\#\mathrm{P}} = \mathrm{NC2}. \tag{11.2}$$

We shall explain the above claims, in the first instance, in the context of gates of arity up to three and basis size 1, as developed earlier in the paper. In section 5 we have already stated the polynomial constraints (5.1) on generators

$$u_{ijk} = \sum q_{rst}(\mathbf{b}_{rst})_{ijk} \tag{11.3}$$

and (5.2) on recognizers

$$\hat{q}_{ijk} = \sum \hat{u}_{rst}(\mathbf{b}_{ijk})_{rst}, \tag{11.4}$$

where summation is over $\{r, s, t\} \in \{0, 1\}^3$. Note that here q and $\hat{q}$ describe the formulation of the combinatorial problem, $\mathbf{b}$ is the basis, and u and $\hat{u}$ are the standard signatures. Also, from the definition of standard signatures in section 4, the various components of $u, \hat{u}$ equal PerfMatch$(G - Z)$, PerfMatch$(\hat{G} - Z)$ for various choices of Z, assuming, for simplicity, that there is just one kind of generator and one kind of recognizer. Hence the components u_{ijk} of u each equal a polynomial expression, say, $U^g_{ijk}(W)$, over the weights W of the generic gate G of size g, and similarly for $\hat{u}_{ijk}$. Hence the third set of constraints we need is

$$u_{ijk} = U^g_{ijk}(W),\ \hat{u}_{ijk} = \hat{U}^{\hat{g}}_{ijk}(\hat{W}). \tag{11.5}$$

It follows from what we have said that (11.3)–(11.5) are solvable iff there is a holographic reduction from the given formulation of $\#F$ to planar PerfMatch using basis size 1 and gate size g. We note that in section 6 we characterized the polynomial equations that can be realized for arities up to four for gates of any size—not just for fixed values of g.

For example, in section 4 we found a solution in the case that $q = (1, 1, 1, 0)$ and $\hat{q} = (-3, 1, 1, 0, 1, 0, 0, 0)$. (N.B. We had an arity 2 gate for the generators, but arbitrary arity for the recognizers, the $\hat{q}$ here being the instance for arity three.) That gave us a polynomial time algorithm for #X-MATCHINGS. Planar matchings are known to be #P-complete [26, 27], even in the planar bipartite case of maximum degree 6 [51]. Hence the solvability of such a #P-complete case would imply $\mathrm{P}^{\#\mathrm{P}} =$ NC2. We note that the nonsolvability of such systems can be proved mechanically, in principle, using computer algebra systems. With such a system we have verified, for example, that the basis given for #X-MATCHINGS is essentially unique among those of size one.

Now (11.3)–(11.5) as stated are limited to bases that have size 1 and two components, and gates of arity up to three. To allow for h rather than two components the only change needed in (11.3)–(11.5) is that $\{r, s, t\}$ should be summed over $\{1, 2, \ldots, h\}$. It is easy to see that these polynomials can be generalized also to allow for arbitrary basis size and arbitrary arity.

By a formulation of a problem we mean a mapping of "the parts" of the problem to generator and recognizer gates in the manner of the reductions we have given for our various specific problems. Given a #P-complete problem such as planar matchings, there are many possible formulations, and it is not clear which, if any, are the most useful for searching for positive solutions of $\mathrm{P}^{\#\mathrm{P}}$. The formulation given in section 8 mapped the nodes to one of two kinds of matchgates. Another would be to map a group of nodes to one kind of matchgate and the edges to another. Also, as illustrated in some reductions in section 9, the original problem may be mapped to a number of matchgrids and the final answer recovered by polynomial interpolation. Clearly there are numerous such formulations that one might try. Thus for any combinatorial problem such as those we have described one can ask whether some formulation of some variant is both #P-complete and has a solvable equation system.

Our treatment here emphasizes solutions from $\mathbb{C}$ only because these seem the easiest to find mechanically. Clearly, solutions over finite fields would be even better for computational purposes if these can be found, though the positive consequences would be only for the corresponding fields in the first instance [52].

Also, we have defined signatures as matrices whose rows correspond to input configurations of matchgates and whose columns correspond to output configurations. In this paper the matchgates we used were all generators or recognizers, corresponding to column and row vectors, respectively. The treatment can be adapted, clearly, to matchgates that have both inputs and outputs.

Throughout this paper we have emphasized planar structures. However, within the same framework we can deal with nonplanar structures as long as in their formulation we also allow for "crossover" nodes (and simulate them effectively with matchgates).

An entirely orthogonal issue is that in this paper we have used the PerfMatch polynomial at the matchgate level and the FKT method for planar graphs as the combining mechanism. An alternative approach for the whole development is to use the Pfaffian at the matchgate level and the Pfaffian combining approach described in

[56, 58] instead.

We have considered only matchgrids that use the same basis throughout. We could equally use a different basis for each connection in the matchgrid.

In conclusion, we observe that if *any* polynomial system generated in the manner described above for a #P-complete problem is solvable, then it would follow that $\mathrm{P}^{\#\mathrm{P}} = \mathrm{NC2}$ and that the seven conjectures enumerated at the beginning of this section would be resolved. In the apparent absence of alternative general approaches to these complexity issues, we suggest that as long as the solvability of even one such polynomial system remains unresolved, it is rational to regard these complexity questions as being truly open.

12. Numerical considerations.

Proof of Theorem 3.2. We shall use Berkowitz's algorithm for computing the determinant [5] and exploit the fact that, unlike Gaussian elimination, it uses no division. Inspection of Berkowitz's algorithm shows that it uses $3\log_2 n + O(1)$ levels of multiplications of pairs of matrices of sizes at most $n \times n$, where the matrix entries initially are either -1 or members of Y, and at subsequent steps are the entries, sometimes multiplied by -1, of matrix products previously obtained.

For $x \in \mathbb{C}$ let $|x|$ be the modulus of x. Let $D = \max\{1, \max\{|x| : x \in Y\}\}$. Our algorithm will depend on Y only through the value of D. For all matrices it will execute the same sequence of arithmetic operations defined by Berkowitz's algorithm except that the arithmetic will be performed in arithmetic with $g = g(n, Y) = O(n^3)(\log_2 D + \log_2 n)$ decimal places of accuracy in fixed precision arithmetic both to the left and to the right of the decimal point. The roundoff error introduced in each operation is at most 2^{-g} in absolute value.

We want F_i to be an upper bound on the modulus of any value computed at the ith level of the exact algorithm. Clearly $F_0 = D$ and $F_i > (F_{i-1})^2 n^k$ suffice if each level is a matrix multiplication of matrices of size at most $n^k \times n^k$. It follows that if $\exp(i) = 2^i$, then $F_i = (n^k D)^{\exp(i)}$ suffices.

We now want ϵ_i to be an upper bound on the maximum absolute error on an output of level i that can occur through the accumulation of roundoff errors. We take $\epsilon_0 = 2^{-g}$ and will maintain $2^{-g} \le \epsilon_i \le 1/2$ and $F_i \ge 1$ by induction. Now the maximum value that can be taken by a product of true absolute values U and V is

$$\begin{aligned}(U + \epsilon_{i-1})(V + \epsilon_{i-1}) + 2^{-g} &= UV + (U + V)\epsilon_{i-1} + \epsilon_{i-1}^2 + 2^{-g} \\ &\le UV + 2\epsilon_{i-1}F_{i-1} + \epsilon_{i-1}^2 + 2^{-g} = UV + \epsilon_i',\end{aligned}$$

say. The maximum error of a subsequent n^k-fold sum, as required by a matrix multiplication, performed as $n^k - 1$ pairwise operations is $(n^k(\epsilon_i' + 2^{-g})$. Combining these gives that

$$\epsilon_i \le n^k(2\epsilon_{i-1}F_{i-1} + \epsilon_{i-i}^2 + 2.2^{-g}) \le 6n^k F_{i-1}\epsilon_{i-1} \le (6n^k)^i F_{i-1}F_{i-2}\dots F_0\epsilon_0,$$

where for the second inequality we have used $\epsilon_{i-1}^2 \le 2F_{i-1}\epsilon_{i-1}$ (since $F_{i-1} > 1$ and $\epsilon_{i-1} \le \epsilon$), and also $2^{-g} \le F_{i-1}\epsilon_{i-1}$. Since $F_i = (n^k D)^{\exp(i)}$ we deduce that

$$(*) \qquad \epsilon_i \le (6n^k)^i (n^k D)^{\exp(i+1)} 2^{-g}.$$

Now if we want to ensure that the integer value of the determinant is computed correctly for $i = 3\log_2 n + O(1)$ and $k = 1$, then $\epsilon_i < 1/2$ is needed for these parameters. From inequality (*) it follows that $g = O(n^3)(\log_2 D + \log_2 n) + O(\log n)^2$

decimal places of accuracy to the right of the decimal point are enough. Since no term is larger than $F_i = (n^k D)^{\exp(i)}$, it follows that $O(n^3)(\log_2 D + \log_2 n)$ decimal places to the left of the decimal point are sufficient, and hence $O(n^3)(\log_2 D + \log_2 n)$ bit arithmetic overall will suffice. □

COROLLARY 3.2.1. *The algorithm in the above theorem can be implemented in NC2.*

Proof. Each of the $O(\log n)$ stages of the algorithm can be implemented by Boolean circuits of polynomial size and $O(\log n)$ depth, since it requires multiplications and n-fold additions [4]. □

COROLLARY 3.2.2. *Theorem* 3.2 *and Corollary* 3.2.1 *also hold if* Y *is infinite,* M_n *contains elements from some* $Y_n \subseteq Y$*, there is a polynomial* $p(n)$ *such that* $2^{p(n)}$ *upper bounds the absolute value of the elements of* Y_n*, and there is an algorithm that given* n *and the index of an element in* Y_n *computes that element to absolute error less than* 2^{-n} *in time polynomial in* n.

Proof. The proofs above support this stronger statement. □

Proof of Theorem 3.3. Proposition 6.2 implies that there is a 3-input gate for the even parity standard signature $[1, 0, 1, 0]$. By chaining $n - 2$ of these together we get a gate for the even parity n input signature $[1, 0, 1, 0, \cdots]$. By deleting one of the external nodes of such a chain we obtain an $n - 1$ input odd parity signature $[0, 1, 0, 1, \cdots]$.

Given a matchgrid Γ with m nonomittable nodes and r omittable nodes on the boundary we shall create a new matchgrid by adding an r-input parity gate in the outside face of Γ with its external nodes identified with the omittable nodes of Γ. The parity of this parity gate will be chosen odd or even according to whether m is odd or even. Clearly PerfMatch for the augmented matchgrid will equal MatchSum for the original matchgrid, as required. □

13. Note added in July 2007.

Since the first appearance of this paper several results have been obtained that shed further light on holographic algorithms. Cai and Choudhary [9] gave a tensor-based treatment and an alternate proof of the Holant theorem. Cai and Choudhary [10] also showed that any standard signature of arity n can be realized by a planar matchgate with $O(n^4)$ nodes, thus generalizing the corresponding result for $n \leq 4$ of our section 6. Further, Cai and Choudhary [11] showed that the planar matchgrid approach taken here is essentially equivalent to the Pfaffian matchcircuit approach of [58]. In Valiant [60] it was shown that the Cai–Choudhary [10] result could be used to show that a certain elementary class of holographic algorithms is insufficient to compute Boolean satisfiability or the permanent. It leaves open whether more general holographic algorithms can compute these functions, or whether this elementary class is sufficient for other #P-complete problems. Cai and Lu [12] gave explicit characterizations of certain classes of signatures that are realizable. Cai and Lu [13] further showed that any matchgrid that is realizable in some basis is also realizable in a basis of size one. Holographic reductions have also been used to prove some new $\oplus$P-completeness [60] and #P-completeness [63] results.

Acknowledgments. I am grateful to Matthew Cook, Oded Goldreich, and Mark Jerrum for their helpful comments on an earlier draft of this paper.

REFERENCES

[1] R. BARBANCHON, *On unique graph* 3*-colorability and parsimonious reductions in the plane*, Theoret. Comput. Sci., 319 (2004), pp. 455–482.

[2] S. Basu, R. Pollack, and M.-F. Roy, *Algorithms in Real Algebraic Geometry*, Springer-Verlag, Berlin, 2003.

[3] R. J. Baxter, *Exactly Solved Models in Statistical Physics*, Academic Press, London, 1982.

[4] P. W. Beame, S. A. Cook, and H. J. Hoover, *Log depth circuits for division and related problems*, SIAM J. Comput., 15 (1986), pp. 994–1003.

[5] S. J. Berkowitz, *On computing the determinant in small parallel time using a small number of processors*, Inform. Process. Lett., 18 (1984), pp. 147–150.

[6] E. Bernstein and U. Vazirani, *Quantum complexity theory*, SIAM J. Comput., 26 (1997), pp. 1411–1473.

[7] P. Bürgisser, *Completeness and Reduction in Algebraic Complexity*, Springer-Verlag, Berlin, 2000.

[8] P. Bürgisser, M. Clausen, and M. A. Shokrollahi, *Algebraic Complexity Theory*, Springer-Verlag, Berlin, 1996.

[9] J.-Y. Cai and V. Choudhary, *Valiant's Holant theorem and matchgate tensors*, in Theory and Applications of Models of Computation, Lecture Notes in Comput. Sci. 3959, Springer-Verlag, Berlin, 2006, pp. 248–261.

[10] J.-Y. Cai and V. Choudhary, *On the Theory of Matchgate Computations*, in Electronic Colloquium on Computational Complexity, ECC-018, 2006.

[11] J.-Y. Cai and V. Choudhary, *Some results on matchgates and holographic algorithms*, in Automata, Languages, and Programming, Part I, Lecture Notes in Comput. Sci. 4051, Springer-Verlag, Berlin, 2006, pp. 703–714.

[12] J.-Y. Cai and P. Lu, *Holographic algorithms: From art to science*, in Proceedings of the 39th ACM Symposium on Theory of Computing, ACM, New York, 2007, pp. 401–410.

[13] J.-Y. Cai and P. Lu, *Holographic algorithms: The power of dimensionality resolved*, in International Colloquium on Automata, Languages, and Programming, 2007, pp. 631–642.

[14] S. A. Cook, *The complexity of theorem proving procedures*, in Proceedings of the 3rd ACM Symposium on Theory of Computing, ACM, New York, 1971, pp. 151–158.

[15] L. J. Cowen, W. Goddard, and C. E. Jesurum, *Defective coloring revisited*, J. Graph Theory, (1997), pp. 205–219.

[16] D. Deutsch, *Quantum theory, the Church-Turing principle, and the universal quantum computer*, Proc. Roy. Soc. London Ser. A, 400 (1985), pp. 97–117.

[17] M. E. Fisher, *Statistical mechanics of dimers on a plane lattice*, Phys. Rev., 124 (1961), pp. 1664–1672.

[18] M. R. Garey and D. S. Johnson, *The rectilinear Steiner tree problem is NP-complete*, SIAM J. Appl. Math., 32 (1977), pp. 826–834.

[19] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

[20] M. R. Garey, D. S. Johnson, and L. Stockmeyer, *Some simplified NP-complete graph problems*, Theoret. Comput. Sci., 1 (1976), pp. 237–267.

[21] M. R. Garey, D. S. Johnson, and R. E. Tarjan, *The planar Hamiltonian circuit problem is NP-complete*, SIAM J. Comput., 5 (1976), pp. 704–714.

[22] F. Hadlock, *Finding a maximum cut of a planar graph in polynomial time*, SIAM J. Comput., 4 (1975), pp. 221–225.

[23] J. Hastad, *Some optimal inapproximability results*, J. ACM, 48 (2001), pp. 798–859.

[24] H. B. Hunt, M. V. Marathe, V. Radhakrishnan, and R. E. Stearns, *The complexity of planar counting problems*, SIAM J. Comput., 27 (1998), pp. 1142–1167.

[25] F. Jaeger, D. L. Vertigan, and D. J. A. Welsh, *On the computational complexity of the Jones and Tutte polynomials*, Math. Proc. Cambridge Philos. Soc., 108 (1990), pp. 35–53.

[26] M. R. Jerrum, *Two-dimensional monomer-dimer systems are computationally intractable*, J. Statist. Phys., 48 (1987), pp. 121–134.

[27] M. R. Jerrum, *Erratum: "Two-dimensional monomer-dimer systems are computationally intractable,"* J. Statist. Phys., 59 (1990), pp. 1087–1088.

[28] M. R. Jerrum, *Counting, Sampling and Integrating: Algorithms and Complexity*, Birkhäuser, Basel, 2003.

[29] M. R. Jerrum and M. Snir, *Some exact complexity results for straight-line computations over semirings*, J. ACM, 29 (1982), pp. 874–897.

[30] R. M. Karp, *Reducibility among combinatorial problems*, in Complexity of Computer Computations, R. E. Miller and J. W. Thatcher, eds., Plenum Press, New York, 1972, pp. 85–103.

[31] P. W. Kasteleyn, *The statistics of dimers on a lattice*, Physica, 27 (1961), pp. 1209–1225.

[32] P. W. Kasteleyn, *Graph theory and crystal physics*, in Graph Theory and Theoretical Physics, F. Harary, ed., Academic Press, London, 1967, pp. 43–110.

[33] M. S. Krishnamoorthy and N. Deo, *Node-deletion NP-complete problems*, SIAM J. Comput., 8 (1979), pp. 619–625.

[34] J. M. Lewis and M. Yannakakis, *The node deletion problem for hereditary properties is NP-complete*, J. Comput. System Sci., 20 (1980), pp. 219–230.

[35] D. Lichtenstein, *Planar formulae and their uses*, SIAM J. Comput., 11 (1982), pp. 329–343.

[36] E. H. Lieb, *Exact solution of the problem of the entropy of two-dimensional ice*, Phys. Rev. Lett., 18 (1967), pp. 692–694.

[37] E. H. Lieb, *Residual entropy of square ice*, Phys. Rev., 162 (1967), pp. 162–172.

[38] E. H. Lieb, *Exact solution of the F model of an antiferroelectric*, Phys. Rev. Lett., 18 (1967), pp. 1046–1048.

[39] E. H. Lieb, *Exact solution of the two-dimensional Slater KDP model of a ferroelectric*, Phys. Rev. Lett., 19 (1967), pp. 108–110.

[40] R. J. Lipton, D. J. Rose, and R. E. Tarjan, *Generalized nested dissection*, SIAM J. Numer. Anal., 16 (1979), pp. 346–358.

[41] M. Liskiewicz, M. Ogihara, and S. Toda, *The complexity of counting self-avoiding walks in subgraphs of two-dimensional grids and hypercubes*, Theoret. Comput. Sci., 304 (2003), pp. 129–156.

[42] L. Lovász and M. D. Plummer, *Matching Theory*, North–Holland, Amsterdam, 1986.

[43] M. Mahajan and V. Vinay, *Determinant: Old algorithms, new insights*, SIAM J. Discrete Math., 12 (1999), pp. 474–490.

[44] G. I. Orlova and Y. G. Dorfman, *Finding the maximum cut in a graph*, Engineering Cybernetics, 10 (1972), pp. 502–506.

[45] V. Y. Pan, *Solving a polynomial equation: Some history and recent progress*, SIAM Rev., 39 (1997), pp. 187–220.

[46] C. H. Papadimitriou, *Computational Complexity*, Addison–Wesley, Reading, MA, 1994.

[47] L. Pauling, *The structure and entropy of ice and of other crystals with some randomness of atomic arrangement*, J. Amer. Chem. Soc., 57 (1935), p. 2680.

[48] V. Strassen, *Gaussian elimination is not optimal*, Numer. Math., 14 (1969), pp. 354–356.

[49] E. Tardos, *The gap between monotone and nonmonotone circuit complexity is exponential*, Combinatorica, 7 (1987), pp. 141–142.

[50] H. N. V. Temperley and M. E. Fisher, *Dimer problems in statistical mechanics—an exact result*, Philosophical Magazine, 6 (1961), pp. 1061–1063.

[51] S. P. Vadhan, *The complexity of counting in sparse, regular, and planar graphs*, SIAM J. Comput., 31 (2001), pp. 398–427.

[52] L. G. Valiant, *The complexity of computing the permanent*, Theoret. Comput. Sci., 8 (1979), pp. 189–201.

[53] L. G. Valiant, *The complexity of enumeration and reliability problems*, SIAM J. Comput., 8 (1979), pp. 410–421.

[54] L. G. Valiant, *Negation can be exponentially powerful*, Theoret. Comput. Sci., 12 (1980), pp. 303–314.

[55] L. G. Valiant, *Why is Boolean complexity theory difficult?*, in Boolean Function Complexity, M. S. Paterson, ed., London Math. Soc. Lecture Note Ser. 169, Cambridge University Press, Cambridge, UK, 1992, pp. 84–94.

[56] L. G. Valiant, *Quantum circuits that can be simulated classically in polynomial time*, SIAM J. Comput., 31 (2002), pp. 1229–1254.

[57] L. G. Valiant, *Expressiveness of matchgates*, Theoret. Comput. Sci., 289 (2002), pp. 457–471.

[58] L. G. Valiant, *Holographic circuits*, in Proceedings of the 32nd International Colloquium on Automata, Languages, and Programming, Lecture Notes in Comput. Sci. 3980, Springer-Verlag, Berlin, 2005, pp. 1–15.

[59] L. G. Valiant, *Completeness for parity problems*, in Proceedings of the 11th International Computing and Combinatonics Conference, Lecture Notes in Comput. Sci. 3959, Springer-Verlag, Berlin, 2005, pp. 1–9.

[60] L. G. Valiant, *Accidental algorithms*, in Proceedings of the 47th IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 2006, pp. 509–517.

[61] D. L. Vertigan and D. J. A. Welsh, *The computational complexity of the Tutte plane: The bipartite case*, Combin. Probab. Comput., 1 (1992), pp. 181–187.

[62] D. J. A. Welsh, *Complexity: Knots, Colourings and Counting*, Cambridge University Press, Cambridge, UK, 1993.

[63] M. Xia, P. Zhang, and W. Zhao, *The complexity of counting on* 3*-regular planar graphs*, Theoret. Comput. Sci., 384 (2007), pp. 111–125.

SIAM J. COMPUT.
Vol. 37, No. 5, pp. 1595–1612

EXTRA UNIT-SPEED MACHINES ARE ALMOST AS POWERFUL AS SPEEDY MACHINES FOR FLOW TIME SCHEDULING*

HO-LEUNG CHAN†, TAK-WAH LAM†, AND KIN-SHING LIU†

Abstract. We study online scheduling of jobs to minimize the flow time and stretch on parallel machines. We consider algorithms that are given extra resources so as to compensate for the lack of future information. Recent results show that a modest increase in machine speed can provide very competitive performance; in particular, using $O(1)$ times faster machines, the algorithm SRPT (shortest remaining processing time) is 1-competitive for both flow time [C. A. Phillips et al., in *Proceedings of STOC*, ACM, New York, 1997, pp. 140–149] and stretch [W. T. Chan et al., in *Proceedings of MFCS*, Springer-Verlag, Berlin, 2005, pp. 236–247] and HDF (highest density first) is $O(1)$-competitive for weighted flow time [L. Becchetti et al., in *Proceedings of RANDOM-APPROX*, Springer-Verlag, Berlin, 2001, pp. 36–47]. Using extra unit-speed machines instead of faster machines to achieve competitive performance is more challenging, as a faster machine can speed up a job but extra unit-speed machines cannot. This paper gives a nontrivial relationship between the extra-speed and extra-machine analyses. It shows that competitive results via faster machines can be transformed to similar results via extra machines, hence giving the first algorithms that, using $O(1)$ times unit-speed machines, are 1-competitive for flow time and stretch and $O(1)$-competitive for weighted flow time.

Key words. online scheduling, flow time, stretch, competitive analysis, extra-resource augmentation

AMS subject classifications. 68Q25, 68W15, 68W40

DOI. 10.1137/060653445

1. Introduction.

In this paper we revisit the problem of online scheduling of jobs to minimize the flow time and stretch on $m \geq 2$ parallel machines (see [24] for a survey). Each job is released at an unpredictable time and is sequential in nature (i.e., it cannot be executed by more than one machine at a time). We consider the case where the processing time (work) of a job is known when it is released. Preemption is allowed at no cost, i.e., a preempted job can be resumed at the point of preemption on any machine. SRPT (shortest remaining processing time first) is a typical example for scheduling in this setting.

Given a schedule, the flow time of a job is the amount of time between its release time and its completion time, and the stretch is the ratio of the flow time to the processing time. In some applications, each job is given a weight, and the concern is the weighted flow time. Common objectives for job scheduling are to minimize the total (or, equivalently, average) flow time (e.g., [19, 20, 2, 1, 21]), stretch (e.g., [7, 9, 22]), or weighted flow time (e.g., [4, 14, 3]) of all jobs. Minimizing stretch is actually a special case of minimizing weighted flow time if we assign the weight of each job to be the reciprocal of its processing time. An online scheduler is said to be c-competitive for flow time (resp., stretch, weighted flow time) if for any job sequence it guarantees the total flow time (resp., stretch, weighted flow time) to be at most c times that of the optimal offline schedule.

*Received by the editors March 2, 2006; accepted for publication (in revised form) August 23, 2007; published electronically February 8, 2008. A preliminary version of this paper appeared in the *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2006, pp. 334–343.

http://www.siam.org/journals/sicomp/37-5/65344.html

†Department of Computer Science, University of Hong Kong, Hong Kong (hlchan@cs.hku.hk, twlam@cs.hku.hk, ksliu@cs.hku.hk).

Related work. SRPT is perhaps the most well-studied online algorithm for minimizing flow time. For scheduling a single machine ($m = 1$), SRPT is 1-competitive [19]. For $m \geq 2$ machines, Leonardi and Raz [20] showed that SRPT achieves the best possible competitive ratio, which is $\Theta(\min(\log n/m, \log \Delta))$, where n is the number of jobs and Δ is the maximum to minimum ratio of processing times. In the offline context, minimizing total flow time on parallel machines is NP-hard [15], and no algorithm is known to have a constant approximation ratio.

Resource augmentation, pioneered by Kalyanasundaram and Pruhs [17], is a popular approach to studying better performance guarantee for improving the competitiveness of online scheduling (e.g., [23, 21, 13, 11, 16, 6]). Specifically, this approach allows the online scheduler to have extra resources so as to compensate for the lack of future knowledge. The key concerns include (i) whether extra resources can lead to 1-competitive (or even better) performance against the optimal offline algorithm using no extra resources, and (ii) how competitive an arbitrarily small amount of extra resources can be. Extra resources can be in the form of faster machines or extra (unit-speed) machines. Below we denote a machine that can complete $s \geq 1$ units of work in one unit of time as an s-speed machine. For minimizing flow time on parallel machines, Phillips et al. [23] were the first to show that SRPT when given $(2-1/m)$-speed machines is 1-competitive or, in short, $(2-1/m)$-speed 1-competitive. McCullough and Torng [21] later showed that SRPT is indeed α-speed $\frac{1}{\alpha}$-competitive for any $\alpha \geq 2 - 1/m$.

Let us switch to the results on minimizing stretch and weighted flow time on parallel machines (one can refer to [22, 5, 4, 23] for results on a single machine). For the case of stretch, Muthukrishnan et al. [22] have showed that SRPT is 14-competitive and no online algorithm can be 1-competitive. Chekuri, Khanna, and Zhu [14] proposed a different algorithm that is 9.81-competitive. They also gave a lower bound on the competitive ratio for weighted flow time of $\Omega(\min(\sqrt{\Delta}, \sqrt{W}, (n/m)^{1/4}))$, where W is the maximum to minimum ratio of the weights. With resource augmentation, Becchetti et al. [6] showed that HDF (highest density first) is $(2 + 2\epsilon)$-speed $(1 + \frac{1}{\epsilon})$-competitive for weighted flow time. This implies that SJF (shortest job first) is $(2 + 2\epsilon)$-speed $(1 + \frac{1}{\epsilon})$-competitive for stretch. Recently, more results on stretch have become known. Chekuri et al. [13] proved that the nonmigratory algorithm IMD (proposed in [1]) is $(1 + \epsilon)$-speed $O(1 + \frac{1}{\epsilon})$-competitive, and Chan et al. [12] showed that SRPT is indeed 5-speed 1-competitive.

Improving the competitiveness via extra unit-speed machines is more challenging. While a faster machine can speed up a job, multiple unit-speed machines cannot. In other words, we cannot use x unit-speed machines to simulate an x-speed machine, yet the reverse is possible (using time-sharing). The literature contains only a few results on exploiting extra machines to obtain competitive scheduling (see [17, 23, 18, 13]). For flow time scheduling on parallel machines, Chekuri et al. [13] have shown that the algorithm IMD when given $(1+\epsilon)m$ unit-speed machines is $O(1+\frac{1}{\epsilon})$-competitive for both flow time and stretch. Whether $O(m)$ unit-speed machines can make an algorithm 1-competitive for flow time or stretch has been an open problem. There are also results on exploiting extra machines in other problem settings of flow time scheduling [17, 23]. In particular, Kalyanasundaram and Pruhs [17] studied the nonclairvoyant setting on a single machine, and Phillips et al. [23] considered the nonpreemptive setting for parallel machines. The power of extra machines has also been studied in the context of deadline scheduling by Koo et al. [18] and Phillips et al. [23].

To ease our discussion, we adopt the following notation. Let α and τ be any

Table 1

Results on flow time scheduling. Results that are given in this paper are marked with †. Note that $\epsilon > 0$ and s are any real numbers and $h \geq 1$ is any integer.

	Extra speed			Extra machines		
	Speed	Competitive ratio		Machines	Competitive ratio	
Flow time	$(1+\epsilon)$	$O(1+1/\epsilon)$	[13]	$\lceil(1+\epsilon)m\rceil$	$O(1+1/\epsilon)$	[13]
	2	1	[23]	$\lceil(2+\epsilon)m\rceil$	$1+1/\epsilon$	†
	$s \geq 2$	$1/s$	[21]	$34m$	1	†
Stretch	$(1+\epsilon)$	$O(1+1/\epsilon)$	[13]	$\lceil(1+\epsilon)m\rceil$	$O(1+1/\epsilon)$	[13]
	5	1	[12]	$533m$	1	†
Weighted flow time	$2+2\epsilon$	$1+1/\epsilon$	[6]	$\lceil(4+24\epsilon)m\rceil$	$8+1/\epsilon$	†
	$16s$, $s \geq 1$	$1/s$	†			
Waiting time	$s \geq 2$	$1/s$	[21]	$(36h-2)m$, $h \geq 1$	$1/h$	†

positive real constants. An algorithm A is said to be α-speed c-competitive (resp., τ-machine c-competitive) for a certain objective function if, for any job sequence, A using m α-speed machines (resp., $\lceil \tau m \rceil$ unit-speed machines) has a performance at most c times of any optimal offline algorithm using m unit-speed machines. When we consider an algorithm A running on m α-speed machines (resp., $\lceil \tau m \rceil$ unit-speed machines), we refer it as $A(\alpha)$ (resp., $A\langle\tau\rangle$).

Our results. This paper shows a nontrivial relationship between the extra-machine analysis and the extra-speed analysis of flow time scheduling. In particular, two methods are given to transform results on competitiveness via faster machines into similar results via extra unit-speed machines. These transformations give the first algorithms that are $O(1)$-machine 1-competitive for flow time and stretch and $O(1)$-competitive for weighted flow time. See Table 1 for a summary of results. Details are as follows.

Flow time transformation. The first transformation is relatively simple, serving as a warm-up. It aims to preserve the flow time of each individual job. Specifically, given an α-speed algorithm $A(\alpha)$ for some $\alpha > 1$, we want to transform A to an algorithm A' that uses extra unit-speed machines to match the flow time of each job as closely as possible. Specifically, our transformation guarantees that A' when given $O(\alpha)m$ (unit-speed) machines increases the flow time of each job at most $\alpha(1 + o(1))$ times. Since SRPT is α-speed $\frac{1}{\alpha}$-competitive for flow time, the transformation gives an algorithm that is $O(\alpha)$-machine $(1 + o(1))$-competitive (and more precisely, $(2 + \epsilon)$-machine $(1 + \frac{1}{\epsilon})$-competitive for any $\epsilon > 0$). Note that A' also preserves the competitiveness on weighted flow time and stretch. Thus, based on HDF [6], the transformation gives an $O(1)$-machine $O(1)$-competitive algorithm for weighted flow time.

Waiting time transformation. The waiting time of a job is the amount of time the job is waiting for processing before it is completed. To obtain an $O(1)$-machine 1-competitive algorithm for flow time and stretch, we need a more complicated transformation based on the total waiting time of jobs. By definition, an algorithm A is $O(1)$-machine 1-competitive for waiting time if and only if A is $O(1)$-machine 1-competitive for on flow time. Note that using extra unit-speed machines can possibly improve the competitive ratio on waiting time to be smaller than one, but it is impossible for flow time.

Consider any algorithm A using α-speed machines. Denote $L_{A(\alpha)}(I)$ the total waiting time incurred for a job sequence I by $A(\alpha)$. The work of McCullough and

Torng [21] implies that SRPT is α-speed $\frac{1}{\alpha}$-competitive for waiting time, where $\alpha \geq 2 - 1/m$. That is, for any I, $L_{SRPT(\alpha)}(I) \leq \frac{1}{\alpha} L_{OPT}(I)$, where OPT denotes the optimal offline algorithm using m unit-speed machines. Using unit-speed machines to simulate SRPT(α) or any $A(\alpha)$ does not necessarily blow up the total waiting time α times. Ideally we want to transform $A(\alpha)$ to an algorithm A' using $\tau m = O(\alpha)m$ unit-speed machines such that $L_{A'\langle\tau\rangle}(I) \leq c L_{A(\alpha)}(I)$, where c is a constant independent of α. Then, substituting $A(\alpha)$ with SRPT(c), we have $L_{A'\langle\tau\rangle}(I) \leq L_{OPT}(I)$. Such a constant c, however, does not exist.[1]

To obtain an $O(1)$-machine 1-competitive algorithm for waiting time, we aim at a less demanding requirement, namely, $L_{A'\langle\tau\rangle}(I) \leq c\, L_{A(\alpha)}(I) + o(L_{OPT}(I))$. In fact, we find that $c = 2$ is already feasible. Then, substituting A with SRPT and α with $O(c)$, we have $L_{A'\langle\tau\rangle}(I) \leq L_{OPT}(I)$, and thus A' is $O(1)$-machine 1-competitive for waiting time, as well as for flow time.

The second transformation can be extended to give a guarantee for normalized waiting time (i.e., the waiting time divided by the processing time). This leads to an algorithm that is $O(1)$-machine 1-competitive for stretch.

Technically speaking, the transformations are based on two concepts called rate control and waiting time allowance. Roughly speaking, we need rate control when jobs are released in a bulk; the idea is to partially process and spread these jobs in a certain way without blowing up the flow time. The other concept is about estimating the maximum waiting time of each job that would not exceed that of the offline optimal algorithm. Both concepts make scheduling easy. To make these two concepts viable, we exploit a simulation of an α-speed competitive algorithm.

This paper also contributes to the extra-speed analysis of SJF and HDF. In particular, we improve the result in [6] to show that HDF can be 16-speed 1-competitive for weighted flow time.

2. Transformation that preserves flow time.

Throughout this paper, we use I to denote a sequence of jobs, and denote the release time and the processing time (i.e., the required work) of a job J as $r(J)$ and $p(J)$, respectively. Note that both $r(J)$ and $p(J)$ are real numbers. Let $A(\alpha)$ be an algorithm using m α-speed machines, where $\alpha \geq 1$ is any real number. This section shows how to transform $A(\alpha)$ to an algorithm, called $\texttt{Scatter}(A(\alpha), \tau)$, that uses $\lceil \tau m \rceil$ unit-speed machines for any $\tau > \alpha$ and incurs a flow time comparable to $A(\alpha)$ as follows.

LEMMA 1. *Consider any job sequence I. For each job $J \in I$, the flow time of J in the schedule of $\texttt{Scatter}(A(\alpha), \tau)$ is at most $\alpha(1 + \frac{\alpha-1}{\tau-\alpha})$ times in the schedule of $A(\alpha)$.*

Details of $\texttt{Scatter}(A(\alpha), \tau)$ are as follows. $\texttt{Scatter}(A(\alpha), \tau)$ divides the $\lceil \tau m \rceil$ machines into two bands. Band 1 uses m machines and Band 2 $\lceil (\tau - 1)m \rceil$ machines. A newly released job J always goes to Band 1 where it is partially processed. Then J is transferred to Band 2 for completion. Consider any sequence I of jobs. We denote the flow time of a job J in the schedule of $A(\alpha)$ as $F_{A(\alpha)}(J)$. We aim to bound the

[1] We consider a simple example where $\alpha = 2$. Let I be a job sequence such that the ith job is released at time $1 - (1/2)^i$ and the required work is $(1/2)^i$. An algorithm with m 2-speed machines can complete each job before the next one is released, thus incurring zero waiting time. On the other hand, any algorithm using τm unit-speed machines must have some job wait after the $(\tau m + 1)$th job is released, thus incurring nonzero waiting time. Note that even one 2-speed machine can complete all jobs with zero waiting time.

flow time of J in Band 1 and Band 2, denoted as $F_1(J)$ and $F_2(J)$, as follows:

$$\text{(i) } F_1(J) = F_{A(\alpha)}(J);\text{ and }\quad \text{(ii) } F_2(J) \le \frac{(\alpha-1)\tau}{\tau-\alpha} F_1(J).$$

Then it follows that the flow time of J in the schedule of $\texttt{Scatter}(A(\alpha), \tau)$ is $F_1(J) + F_2(J) \le (1 + \frac{(\alpha-1)\tau}{\tau-\alpha}) F_{A(\alpha)}(J) = \alpha(1 + \frac{\alpha-1}{\tau-\alpha}) F_{A(\alpha)}(J)$, which is as stated in Lemma 1.

Simulation. Requirement (i) can be achieved easily by simulating the execution of $A(\alpha)$. Precisely, Band 1 uses m machines and schedules the jobs according to a simulated copy of $A(\alpha)$, which uses m α-speed machines. That is, Band 1 runs a job J if and only if $A(\alpha)$ runs the job J. When $A(\alpha)$ completes J, Band 1 transfers J to Band 2. Thus, $F_1(J) = F_{A(\alpha)}(J)$, and J is processed in Band 1 for exactly $p(J)/\alpha$ units of work.

Rate control. Let $rem(J)$ be the amount of remaining work of a job J when it is transferred to Band 2. Jobs may be released in bulk to Band 1, yet they will each be partially processed before being transferred to Band 2 and will thus spread out eventually. Band 1 controls the rate of work transferred to Band 2 in the sense that jobs released and transferred within any time interval have bounded remaining work (see Lemma 2 for technical details). With rate control, Requirement (ii) can be satisfied easily using a simple strategy, namely, the latest release time first algorithm (LRT), which at any time t processes jobs with latest release time (to Band 1). Ties are broken arbitrarily.

The above discussion of `Scatter` is summarized in Algorithm 1, followed by two lemmas on the work transferred to Band 2 and the flow time in Band 2.

Algorithm 1. $\texttt{Scatter}(A(\alpha), \tau)$, which uses $\lceil \tau m \rceil$ unit-speed machines.

Job Release: A newly released job goes to Band 1.

Band 1: It uses m machines. Jobs are scheduled according to a simulated copy of $A(\alpha)$. When a job J is completed in the simulated $A(\alpha)$, it is transferred to Band 2.

Band 2: It uses $\lceil (\tau-1)m \rceil$ machines and it completes all jobs using LRT.

LEMMA 2 (rate control). *Consider any time interval* $[t, t']$*, let* H *be the set of jobs released within* $[t, t']$ *and transferred to Band* 2 *within* $[t, t']$*. Then* $\sum_{J \in H} rem(J) \le (\alpha - 1)(t' - t)m$*.*

Proof. Each job $J \in H$ has been processed by Band 1 for $\frac{1}{\alpha}p(J)$ units of work during the time interval $[t, t']$. Band 1 can perform at most $m(t' - t)$ units of work during $[t, t']$. Thus, $\sum_{J \in H} \frac{1}{\alpha} p(J) \le m(t' - t)$, and $\sum_{J \in H} rem(J) = \sum_{J \in H} (1 - \frac{1}{\alpha}) p(J) \le (\alpha - 1)(t' - t)m$. □

LEMMA 3 (LRT). *For any job* J*,* $F_2(J) \le (\alpha - 1) \times \frac{\tau}{\tau - \alpha} F_1(J)$*.*

Proof. For any job J, let $t_0 = r(J)$, let t_1 be the time J is transferred from Band 1 to Band 2, and let t_2 be the time J is completed by Band 2. Note that $F_1(J) = t_1 - t_0 \ge p(J)/\alpha$, and $F_2(J) = t_2 - t_1$.

Assume that J waits for a number of time periods in Band 2 before it is completed. Let S be the set of jobs that have ever received processing in Band 2 while J is waiting. For each job $J' \in S$, J' is released no earlier than t_0 (i.e., $r(J') \ge r(J)$), and J' is transferred to Band 2 no later than t_2. Applying Lemma 2 to the interval $[t_0, t_2]$, we have $\sum_{J' \in S} rem(J') \le (\alpha - 1)(t_2 - t_0)m$.

Whenever J waits in Band 2, all the $\lceil (\tau - 1)m \rceil$ machines are processing jobs in

S. The waiting time of J in Band 2 is at most

$$\frac{1}{\lceil(\tau-1)m\rceil}\sum_{J'\in S} rem(J') \le \frac{1}{\lceil(\tau-1)m\rceil}(\alpha-1)(t_2-t_0)m.$$

Therefore,

$$\begin{aligned} F_2(J) &= t_2 - t_1 \\ &\le \frac{\alpha-1}{\alpha}p(J) + \frac{\alpha-1}{\lceil(\tau-1)m\rceil}(t_2-t_0)m \\ &\le (\alpha-1)F_1(J) + \frac{\alpha-1}{\tau-1}(F_1(J)+F_2(J)). \end{aligned}$$

Rearranging the last inequality, we have $F_2(J) \le \frac{(\alpha-1)\tau}{\tau-\alpha}F_1(J)$. □

Based on the results that SRPT is 2-speed $\frac{1}{2}$-competitive for flow time [21], and HDF is 4-speed 2-competitive for weighted flow time [6], we can apply Lemma 1 to obtain the following extra-machine competitive results.

COROLLARY 4. *Consider any $\epsilon > 0$.* (i) *The algorithm* `Scatter`(SRPT(2), 2+ϵ) *is* (2+ϵ)*-machine* (1+1/ϵ)*-competitive for flow time.* (ii) *The algorithm* `Scatter`(HDF(4), $4+24\epsilon$) *is* $(4+24\epsilon)$*-machine* $(8+1/\epsilon)$*-competitive for weighted flow time.*

3. Transformation that preserves waiting time.

The waiting time of a job is the amount of time the job is waiting for processing before it is completed. Recall that SRPT is α-speed $(1/\alpha)$-competitive for flow time, where $\alpha \ge 2 - 1/m$ [21]. In the schedule of SRPT(α), the flow time of a job J is exactly $p(J)/\alpha$ plus the waiting time. Thus, SRPT is also α-speed $(1/\alpha)$-competitive for waiting time.

In this section we show how to transform an algorithm A that uses m α-speed machines, where $\alpha \ge 1$ is any real number, to an algorithm `Scatter_&_Squash`$(A(\alpha), \tau)$ that uses $\lceil \tau m \rceil$ unit-speed machines and incurs a total waiting time comparable to that of $A(\alpha)$. Details are as follows.

LEMMA 5. *Let $\tau = 7\alpha+5k-2$ for any integer $k \ge 1$. Then, for any job sequence I, the total waiting time incurred by* `Scatter_&_Squash`$(A(\alpha),\tau)$ *is at most* $2L_{A(\alpha)}(I) + \frac{1}{k}L_{OPT}(I)$*, where $L_{A(\alpha)}(I)$ and $L_{OPT}(I)$ denote the total waiting time incurred by $A(\alpha)$ and the optimal algorithm OPT using m unit-speed machines, respectively.*

We will prove Lemma 5 in section 3.1. Let us consider its implication first. Suppose that A is α-speed $(1/x)$-competitive for waiting time for some $x \ge 1$. Let $k = \lceil x \rceil$ and $\tau = 7\alpha + 5\lceil x \rceil - 2$. By Lemma 5, `Scatter_&_Squash` gives an $O(\alpha+x)$-machine $(3/x)$-competitive algorithm for waiting time. In other words, based on the result that SRPT is 3-speed (1/3)-competitive for waiting time [21], we immediately obtain a 34-machine 1-competitive algorithm for waiting time. Notice that an algorithm using unit-speed machines is 1-competitive for waiting time if and only if it is 1-competitive for flow time. The competitive ratio of `Scatter_&_Squash` for waiting time can be further reduced to less than one using a more competitive result of SRPT. However, for flow time, the competitive ratio of an algorithm using unit-speed machines is lower bounded by one. The following corollary summarizes these results.

COROLLARY 6. (i) `Scatter_&_Squash` *gives an algorithm that is* 34*-machine* 1*-competitive for flow time.* (ii) *For any integer $h \ge 1$,* `Scatter_&_Squash` *(based on* SRPT(3h)*) gives an algorithm that is* $(36h-2)$*-machine* $(1/h)$*-competitive for waiting time.*

Algorithm 2. `Scatter_&_Squash`$(A(\alpha), \tau)$, where $\tau = 7\alpha + 5k - 2$ for any integer $k \geq 1$.

Job Release: A newly released job goes to Band 1a.

Band 1a: It uses m machines. Jobs are scheduled according to a simulated copy of $A(\alpha)$. At any time t, if a job J is completed in the simulated $A(\alpha)$, J is transferred to Band 1b if $t \leq r(J) + p(J)$; otherwise, J is transferred to Band 2 (with $AWT(J) = L_{1a}(J)$).

Band 1b: It uses $(2k+1)m$ machines. It runs the algorithm EPPBUSY$\langle 2k+1 \rangle$; i.e., at any time, it arbitrarily selects up to $(2k+1)m$ jobs that are still within their earliest processing periods for execution. At the end of the earliest processing period of a job J, if J is not completed, then J is transferred to Band 2 (with $AWT(J) = L_{1a}(J) + L_{1b}(J)$).

Band 2: It uses $\lceil (7\alpha + 3k - 4)m \rceil$ machines. It runs the MIN-AWT algorithm; i.e., at any time, it greedily schedules the jobs with smallest AWT. A job remains in Band 2 until it is completed.

3.1. The algorithm.

As shown in Algorithm 2, `Scatter_&_Squash` divides the machines into 3 bands called Band 1a, Band 1b, and Band 2, using, respectively, m, $(2k+1)m$, and $\lceil (7\alpha + 3k - 4)m \rceil$ machines, where k is any integer ≥ 1. Similar to the algorithm `Scatter`, Band 1 (comprising Band 1a and Band 1b) only partially processes the jobs, and Band 2 ensures that all jobs get completed. For $i =$ 1a, 1b, 1, or 2, we denote $L_i(J)$ the waiting time of a job J in Band i, and let $L_i(I) = \sum_{J \in I} L_i(J)$. Note that $L_1(J) = L_{1a}(J) + L_{1b}(J)$. Consider any job sequence I and any job J in I. Given an algorithm $A(\alpha)$, `Scatter_&_Squash` aims to guarantee that $L_{1a}(J) = L_{A(\alpha)}(J)$; $L_{1b}(I) \leq \frac{1}{2k} L_{OPT}(I)$; and $L_2(J) \leq L_1(J)$. Then Lemma 5 follows. To achieve $L_2(J) \leq L_1(J)$, we ensure that jobs transferred from Band 1 to Band 2 are easy to schedule in the following sense. Let $rem(J)$ be the remaining work of a job J when J is transferred to Band 2.

(a) *Rate control.* For any time interval T, the sum of $rem(J)$ over all jobs released during T and transferred from Band 1 to Band 2 during T is at most $(\alpha - 1)m|T|$.

(b) *Bounded remaining work.* $rem(J) \leq L_1(J)$.

Band 1a uses the simulation technique presented in the last section. It uses m machines and schedules jobs according to a simulated copy of $A(\alpha)$. When a job J is transferred out of Band 1a, $p(J)/\alpha$ units of its work have been processed, and Band 1a incurs exactly the same waiting time as $A(\alpha)$; i.e., $L_{1a}(J) = L_{A(\alpha)}(J)$. By Lemma 2, Band 1a provides the rate control property.

Define the *earliest processing period* of a job J to be the time interval $[r(J), r(J) + p(J)]$. To achieve the bounded remaining work property, we simply ensure that each job is transferred to Band 2 after its earliest processing period. That is, a job transferred out of Band 1a within its earliest processing period is retained in Band 1b until the end of its earliest processing period.

LEMMA 7. *If a job J is transferred from Band* 1 *to Band* 2 *at the end of or after J's earliest processing period, then $rem(J) \leq L_1(J)$.*

Proof. Let $w(J) \geq 0$ be the amount of work done on J in Band 1. J is transferred to Band 2 at $r(J) + w(J) + L_1(J)$, which is at least $r(J) + p(J)$. Thus, $L_1(J) \geq p(J) - w(J) = rem(J)$. □

Band 1 as a whole still satisfies the rate control property because jobs released and transferred to Band 2 within an interval T are a subset of jobs released and transferred out of Band 1a within an interval T. The nontrivial part is how to ensure that the waiting time incurred in Band 1b is comparable to $A(\alpha)$ or OPT. To our surprise, we find that Band 1b, using an arbitrary algorithm with $(2k+1)m$ machines to process jobs during their earliest processing periods, would incur a total waiting time of at most $\frac{1}{2k}$ times of OPT. We denote such an algorithm by EPPBUSY$\langle 2k+1\rangle$. Formally speaking, at any time, EPPBUSY considers only jobs that are still in their earliest processing periods, and it arbitrarily selects one such job for each machine. Notice that EPPBUSY may not complete a job J and does not incur waiting time beyond the earliest processing period of J, yet OPT does both. In section 3.2, we will give a careful charging scheme to relate the waiting times of EPPBUSY and OPT. In summary, Band 1 has the following upper bound on waiting time:

$$L_1(I) = L_{1a}(I) + L_{1b}(I) \leq L_{A(\alpha)}(I) + \tfrac{1}{2k} L_{OPT}(I).$$

For Band 2, we want to complete the remaining work of each job J such that $L_2(J)$ is at most $L_1(J)$. In other words, J is allowed to wait in Band 2 up to $L_1(J)$ units of time. To ease our discussion, we assume that each job transferred to Band 2 is associated with an extra parameter $AWT(J)$ representing the allowed waiting time of J, and $AWT(J)$ is set to $L_1(J)$. Based on the properties of rate control and bounded remaining work, we find that MIN-AWT, a greedy strategy that schedules jobs with smallest AWT (ties are broken arbitrarily), can complete each job within its allowed waiting time if Band 2 is given $\lceil(7\alpha+3k-4)m\rceil$ machines. The above description of `Scatter_&_Squash` is summarized in Algorithm 2. The rest of this subsection is devoted to proving that for each job J in I, MIN-AWT incurs a waiting time at most $L_1(J)$.

MIN-AWT. Consider a job J that is transferred from Band 1 to Band 2, say, at time $\mathit{tsf}(J)$. Recall that $AWT(J)$ is set to $L_1(J)$, and the remaining work of J at $\mathit{tsf}(J)$, denoted $\mathit{rem}(J)$, is at most $AWT(J)$. We want to show that if Band 2 uses MIN-AWT on $O(\alpha+k)$ machines, then J waits no more than $AWT(J)$ units of time in Band 2, or, equivalently, J is completed by the time $d(J) = \mathit{tsf}(J) + \mathit{rem}(J) + AWT(J)$. We call $d(J)$ the deadline of J in Band 2.

We use induction to show that every job is completed by its deadline. Consider jobs in increasing order of deadlines. Let J be a job. Assume that all jobs with deadline earlier than J are completed by their deadlines. We focus on the total waiting time of J up to $d(J)$. During $[\mathit{tsf}(J), d(J)]$, whenever J is waiting, all machines in Band 2 are processing jobs J' with the following properties:

1. $AWT(J') \leq AWT(J)$;
2. J' is transferred to Band 2 no earlier than $\mathit{tsf}(J) - 2AWT(J)$ (otherwise, $d(J') = \mathit{tsf}(J') + \mathit{rem}(J') + AWT(J') < \mathit{tsf}(J) < d(J)$ and J' is completed before $\mathit{tsf}(J)$); and
3. J' is transferred to Band 2 no later than $d(J)$.

Let S be the set of all jobs J' satisfying the above properties. Below we upper bound the sum of $\mathit{rem}(J')$ over all J' in S.

Lemma 8. *$\sum_{J' \in S} \mathit{rem}(J') \leq \delta m\, AWT(J)$, where $\delta = 7\alpha + 3k - 4$.*

Proof. Let $t_1 = \mathit{tsf}(J) - 2AWT(J)$. By definition, jobs in S are transferred to Band 2 within $[t_1, d(J)]$. Let $t_0 = t_1 - xAWT(J)$ for some $x > 1$. We divide the jobs in S according to their release time (to Band 1a). Let $S_1 = \{J' \in S \mid r(J') \geq t_0\}$ and $S_2 = S - S_1$.

Jobs in S_1. We use the rate control property to bound the sum of $rem(J')$ over all J' in S_1. For each job J' in S_1, J' is released during the time interval $[t_0, d(J)]$ and is transferred to Band 2 during $[t_1, d(J)]$. Recall that $t_0 < t_1$. By the rate control property, $\sum_{J' \in S_1} rem(J')$ is at most $(\alpha - 1)m(d(J) - t_0) \leq (\alpha - 1)m(x + 4)AWT(J)$.

Jobs in S_2. In this case, we exploit the bounded remaining work property and the fact that $AWT(J)$ is set to $L_1(J)$. Each job J' in S_2 is released before t_0 and transferred to Band 2 on or after t_1. Thus, J' is kept in Band 1 for a period of length at least $t_1 - t_0 \geq x\, AWT(J)$. Note that $L_1(J')$ (i.e., the waiting time of J' in Band 1) $= AWT(J') \leq AWT(J)$. Thus, J' is processed by Band 1 for at least $(x-1)\, AWT(J)$ units of work from t_0 to t_1. Band 1 has only $(2k + 2)m$ machines and it performs at most $(2k + 2)m(xAWT(J))$ units of work from t_0 to t_1. Thus,

$$\begin{aligned}(2k + 2)mx\, AWT(J) &\geq \sum_{J' \in S_2} (x - 1)AWT(J) \\ &\geq \sum_{J' \in S_2} (x - 1)AWT(J') \\ &\geq \sum_{J' \in S_2} (x - 1)rem(J').\end{aligned}$$

So, $\sum_{J' \in S_2} rem(J')$ is at most $(2k+2)m\frac{x}{x-1}AWT(J)$. In conclusion, $\sum_{J' \in S} rem(J') \leq [(\alpha - 1)(x + 4) + (2k + 2)\frac{x}{x-1}]m\, AWT(J)$. Putting $x = 3$, we obtain Lemma 8. ☐

We are ready to prove that J can be completed by $d(J)$. Whenever J is waiting in Band 2 during $[tsf(J), d(J)]$, all machines of Band 2 are processing jobs belonging to the set S, and the sum of $rem(J')$ over all jobs $J' \in S$ is at most $(7\alpha + 3k - 4)m\, AWT(J)$. Band 2 uses $\lceil(7\alpha + 3k - 4)m\rceil$ machines, and the work due to S can keep J waiting in Band 2 for at most $AWT(J)$ units of time. Thus, J is completed by $d(J)$.

3.2. Analysis of EPPBUSY. `Scatter_&_Squash` uses the algorithm EPPBUSY in Band 1b. To upper bound the waiting time incurred in Band 1b, we first study in this section the waiting time incurred by EPPBUSY when it is used to process a sequence of jobs. This result may be of independent interest.

EPPBUSY$\langle h \rangle$ uses hm machines for any integer $h \geq 2$. It schedules a job only within its earliest processing period and it may not be able to finish each job. Let OPT be the optimal scheduler, which uses m machines to process all jobs to completion and minimizes the total waiting time.

For any job sequence I, let $P(I)$ be the schedule produced by EPPBUSY$\langle h \rangle$ on I, and, similarly, $OPT(I)$ for OPT. Note that a job remains in $P(I)$ only during its earliest processing period, while a job remains in $OPT(I)$ until it is completed. We want to show that the total waiting time of jobs in $P(I)$ is at most $\frac{1}{h-1}$ of that of $OPT(I)$.

We first focus on the schedule $P(I)$. $P(I)$ may contain one or more waiting periods (a waiting period is a period in which at least one job is waiting at any time). Denote these waiting periods as $\lambda_1 = [t_1, t'_1], \lambda_2 = [t_2, t'_2], \lambda_3 = [t_3, t'_3], \ldots$, where $t_1 < t'_1 < t_2 < t'_2 < t_3 < t'_3 < \cdots$. Let $|\lambda_i| = t'_i - t_i$. Note that $P(I)$ accumulates waiting time only during the waiting periods.

DEFINITION 9. *Let* $S = \{\lambda_u, \lambda_{u+1}, \ldots, \lambda_v\}$ *be a collection of consecutive waiting periods. Recall that* h *is the parameter required by EPPBUSY*$\langle h \rangle$. S *is said to be* h*-close if* $t_{u+1} \leq t_u + h|\lambda_u|$, $t_{u+2} \leq t_u + h(|\lambda_u| + |\lambda_{u+1}|), \ldots,$ *and* $t_v \leq t_u + h\sum_{i=u}^{v-1} |\lambda_i|$.

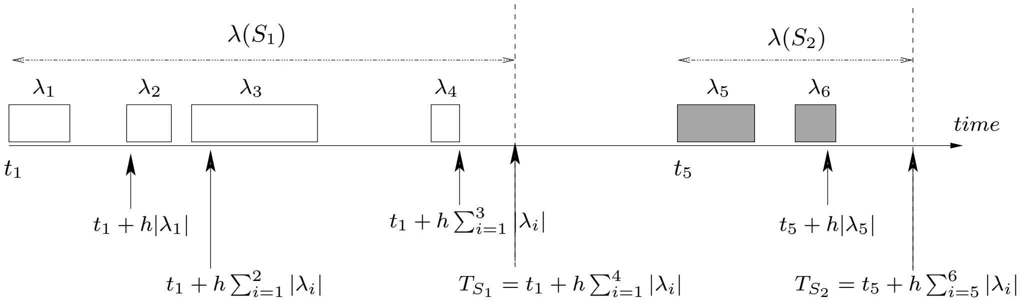

$\lambda(S_1)$
$\lambda(S_2)$
λ_1
λ_2
λ_3
λ_4
λ_5
λ_6
time
t_1
t_5
$t_1 + h|\lambda_1|$
$t_1 + h\sum_{i=1}^{3}|\lambda_i|$
$t_5 + h|\lambda_5|$
$t_1 + h\sum_{i=1}^{2}|\lambda_i|$
$T_{S_1} = t_1 + h\sum_{i=1}^{4}|\lambda_i|$
$T_{S_2} = t_5 + h\sum_{i=5}^{6}|\lambda_i|$

FIG. 1. *Two h-close collections of waiting periods ($h = 2$ in this example).*

Furthermore, define t_S to be the time $(t_u + h\sum_{i=u}^{v}|\lambda_j|)$, and define $\lambda(S)$ to be the interval $[t_u, t_S]$. Note that $|\lambda(S)| = h\sum_{j=u}^{v}|\lambda_j|$. See Figure 1 for an example.

FACT 10. *For any $u \le i \le v$, $t_S - t_i \ge h(|\lambda_i| + |\lambda_{i+1}| + \cdots + |\lambda_v|)$.*

We partition the waiting periods in $P(I)$ into *maximal* h-close collections $S_1 = \{\lambda_1, \lambda_2, \ldots, \lambda_{k_1}\}$, $S_2 = \{\lambda_{k_1+1}, \lambda_{k_1+2}, \ldots, \lambda_{k_2}\}$, ... That is, the next waiting period beyond each S_i has a starting time greater than t_{S_i}. The notion of a maximal h-close collection of waiting periods defines a framework for our analysis. In the following, we show that for each maximal h-close collection S of waiting periods, the waiting time incurred by $P(I)$ within the interval $\lambda(S)$ is at most a factor of $1/(h-1)$ of the waiting time incurred by $OPT(I)$ within $\lambda(S)$.

The following notion further provides a tool for lower bounding the waiting time of $OPT(I)$.

DEFINITION 11. *Consider any interval $\lambda = [t, t']$. Let J be a job with required work $p(J)$. If λ is enclosed in the earliest processing period of J, the work required by J can be partitioned into three chunks of size $t - r(J)$, $t' - t$, and $p(J) - (t' - r(J))$, respectively. The middle chunk is referred to as the* λ-work *of J. In general, when λ is not enclosed in the earliest processing period of J, we let $\lambda' = \lambda \cap [r(J), r(J) + p(J)]$ and define the λ-work of J to be its λ'-work.*

We denote the size of the λ-work of J as $W(J, \lambda)$, i.e., $W(J, \lambda) = |\lambda \cap [r(J), r(J) + p(J)]|$. A fact useful to our analysis is that if $W(J, \lambda) > 0$, the earliest time $OPT(I)$ (or any schedule using unit-speed machines) can start processing the λ-work of J is $\max\{t, r(J)\}$.

Let $S = \{\lambda_u, \lambda_{u+1}, \ldots, \lambda_v\}$ be a maximal h-close collection of waiting periods. Let J be any job. Consider the λ_i-work of J for all $\lambda_i \in S$. Below, we give a way to mark the earliest possible schedule of the λ_i-work of J in $OPT(I)$. Let $\lambda_i = [t_i, t'_i]$ be the first waiting period of S that overlaps with the earliest processing period of J (i.e., $W(J, \lambda_i) > 0$). Note that OPT cannot process the λ_i-work of J earlier than t_i or $r(J)$. We mark the first $W(J, \lambda_i)$ units of work starting from the time $\max\{t_i, r(J)\}$ in the schedule of J in $OPT(I)$. For each subsequent $j > i$, if the λ_j-work of J is nonnull, we identify, in the schedule of J in $OPT(I)$, the first time $t \ge t_j$ when no work has been marked, and we mark another $W(J, \lambda_j)$ units of work starting from t. We have the following lemma on the work marked on the schedule of J in $OPT(I)$.

Within the time interval $\lambda(S)$, we denote the waiting time of J incurred by $P(I)$ as $L_P(J)|_{\lambda(S)}$, and similarly $L_{OPT}(J)|_{\lambda(S)}$ for $OPT(I)$.

LEMMA 12. *Suppose that in the course of marking all the λ_i-work of a job J in $OPT(I)$, a total of y units of work are marked beyond t_S. Then $L_{OPT}(J)|_{\lambda(S)}$ is at least $(h-1)y$.*

Proof. Assume that $\lambda_i = [t_i, t'_i]$ is the first waiting period in S such that part of the λ_i-work of J is marked beyond t_S. Then, $y \leq |\lambda_i| + |\lambda_{i+1}| + \cdots + |\lambda_v|$. In $OPT(I)$, the λ_i-work of J is not completed by time t_S. Thus, within $\lambda(S)$, the waiting time of J is at least $t_S - t'_i = t_S - t_i - |\lambda_i|$. By Fact 10, we conclude that $L_{OPT}(J)|_{\lambda(S)} \geq h(|\lambda_i| + |\lambda_{i+1}| + \cdots + |\lambda_v|) - |\lambda_i| \geq (h-1)y$. □

LEMMA 13. $\sum_{J\in I} L_P(J)|_{\lambda(S)} \leq \frac{1}{h-1} \sum_{J\in I} L_{OPT}(J)|_{\lambda(S)}$.

Proof. With respect to $P(I)$, the total waiting time of all jobs during a waiting period λ_i is exactly the total length of the λ_i-work of all jobs minus the amount of work that EPPBUSY$\langle h\rangle$ processes during λ_i. That is, $\sum_{J\in I} L_P(J)|_{\lambda_i} = \sum_{J\in I} W(J,\lambda_i) - hm|\lambda_i|$. Summing over all waiting periods in S, we have

$$\begin{aligned}\sum_{J\in I} L_P(J)|_{\lambda(S)} &= \sum_{i=u}^{v}\sum_{J\in I} W(J,\lambda_i) - \sum_{i=u}^{v} hm|\lambda_i| \\ &= \sum_{i=u}^{v}\sum_{J\in I} W(J,\lambda_i) - m|\lambda(S)|.\end{aligned}$$

Note that $\sum_{J\in I} L_P(J)|_{\lambda(S)} \geq 0$, and hence $\sum_{i=u}^{v}\sum_{J\in I} W(J,\lambda_i) \geq m|\lambda(S)|$.

Since OPT has only m machines, during $\lambda(S)$, OPT can process at most $m|\lambda(S)|$ units of work. Consider the λ_i-work of all jobs over all λ_i in S. Their total size is $\sum_{i=u}^{v}\sum_{J\in I} W(J,\lambda_i)$, which exceeds $m|\lambda(S)|$. Thus, not all λ_i-work can be marked within $\lambda(S)$ in $OPT(I)$. The total amount of work marked beyond t_S in $OPT(I)$ is at least $\sum_{i=u}^{v}\sum_{J\in I} W(J,\lambda_i) - m|\lambda(S)| = \sum_{J\in I} L_P(J)|_{\lambda(S)}$. By Lemma 12, $L_{OPT}(J)|_{\lambda(S)}$ is at least the total amount of λ_i-work marked beyond t_S for J. Thus, $\sum_{J\in I} L_{OPT}(J)|_{\lambda(S)}$ is at least $(h-1)\sum_{J\in I} L_P(J)|_{\lambda(S)}$. □

COROLLARY 14. *For any job sequence I, let $L_P(I)$ be the total waiting time incurred by EPPBUSY$\langle h\rangle$ and $L_{OPT}(I)$ be that for OPT. Then, $L_P(I) \leq \frac{1}{h-1} L_{OPT}(I)$.*

EPPBUSY in Scatter_&_Squash. We are now ready to analyze the waiting time incurred by Band 1b of `Scatter_&_Squash`. Recall that Band 1b uses $(2k+1)m$ machines to run EPPBUSY$\langle 2k+1\rangle$. We want to prove that for any job sequence I, the total waiting time incurred in Band 1b of `Scatter_&_Squash` is at most $1/(2k)L_{OPT}(I)$.

By definition of `Scatter_&_Squash`, a job J in I is transferred to Band 1b only after it is partially scheduled in Band 1a. Thus, J remains in Band 1b only during a subinterval of its earliest processing period.

Let us compare the schedule of Band 1b with the schedule of I when using a stand-alone copy of EPPBUSY$\langle 2k+1\rangle$. Denote the latter schedule Φ. At any time t, if a job J remains in Band 1b, then t is within J's earliest processing period, and J remains in Φ for possible processing. Thus, jobs remaining in Band 1b are a subset of jobs remaining in Φ. As both Band 1b and the stand-alone EPPBUSY$\langle 2k+1\rangle$ are using $(2k+1)m$ machines, the number of jobs waiting in Band 1b, denoted $\#_{1b}(I,t)$, is at most the number of jobs waiting in Φ, denoted $\#_{\Phi}(I,t)$.

Let $L_{1b}(J)$ and $L_{\Phi}(J)$ be the waiting times of J in the schedules of Band 1b and Φ, respectively. We have

$$\sum_{J\in I} L_{1b}(J) = \int \#_{1b}(I,t)dt \leq \int \#_{\Phi}(I,t)dt = \sum_{J\in I} L_{\Phi}(J) \leq \frac{1}{2k}\sum_{J\in I} L_{OPT}(J).$$

4. Extension to weighted waiting time and stretch.

The normalized waiting time of a job refers to the waiting time divided by the processing time. An algorithm is $O(1)$-machine 1-competitive for stretch if and only if it is $O(1)$-machine 1-competitive for normalized waiting time. To derive an $O(1)$-machine 1-competitive algorithm for stretch, we want `Scatter_&_Squash` to transform a given α-speed algorithm $A(\alpha)$ to an $O(\alpha)$-machine algorithm that preserves the normalized waiting time. In fact, `Scatter_&_Squash` can even be extended to preserve the weighted waiting time when every job is given an arbitrary weight. The idea is quite simple. By the definition of `Scatter_&_Squash` (in section 3.1), the performance guarantee for Band 1a and Band 2 is based on the (unweighted) waiting time of each job, and it remains the same when weighted waiting time is concerned. As a whole, Band 1a still incurs the same amount as $A(\alpha)$ does, and Band 2 incurs no more than Band 1 does. Only Band 1b requires modification to cater to the weighted setting.

Before looking at the details of Band 1b, we prove a lemma that can transform a special relationship of the unweighted waiting times of two schedules into a relationship of their total weighted waiting times. Below, x_i and y_i denote the waiting time of a job in two schedules, and w_i is the weight of the job.

Lemma 15. *Let $x_1, x_2, \dots, x_r$ and $y_1, y_2, \dots, y_r$ be two sequences of nonnegative reals. Let γ be any positive real. Suppose that $\sum_{i=1}^{q} x_i \le \gamma \sum_{i=1}^{q} y_i$ for all $q = 1, \dots, r$. Then for any nondecreasing sequence of positive reals $w_1 \ge w_2 \ge \cdots \ge w_r$, we have $\sum_{i=1}^{r} w_i x_i \le \gamma \sum_{i=1}^{r} w_i y_i$.*

Proof. We prove the lemma by induction on r. The case for $r = 1$ is obvious. Assume that the lemma is true when $r = z$, for some integer $z \ge 1$. When $r = z + 1$ we consider the following two cases.

Case 1. If $x_{z+1} \le \gamma y_{z+1}$, then $\sum_{i=1}^{z+1} w_i x_i = \sum_{i=1}^{z} w_i x_i + w_{z+1} x_{z+1} \le \gamma \sum_{i=1}^{z} w_i y_i$ $+ \gamma w_{z+1} y_{z+1} = \gamma \sum_{i=1}^{z+1} w_i y_i$.

Case 2. Otherwise, $x_{z+1} > \gamma y_{k+1}$. Let $\delta = x_{z+1} - \gamma y_{z+1}$.

$$\begin{aligned}\sum_{1\le i\le z+1} w_i x_i &= \sum_{1\le i<z} w_i x_i + w_z x_z + w_{z+1}(\gamma \times y_{z+1} + \delta)\\ &= \sum_{1\le i<z} w_i x_i + w_z x_z + w_{z+1}\delta + w_{z+1}\gamma y_{z+1}\\ &\le \sum_{1\le i<z} w_i x_i + w_z(x_z + \delta) + \gamma w_{z+1} y_{z+1}.\end{aligned}$$

Define the sequence $(d_1, d_2, \dots, d_z)$ such that $d_i = x_i$ for $i = 1, \dots, z-1$ and $d_z = x_z + \delta$. For any $q = 1, \dots, z-1$, $\sum_{i=1}^{q} d_i = \sum_{i=1}^{q} x_i \le \gamma \sum_{i=1}^{q} y_i$. For $q = z$,

$$\sum_{1\le i\le z} d_i = \sum_{1\le i\le z} x_i + \delta = \sum_{1\le i\le z+1} x_i - \gamma y_{z+1} \le \gamma \sum_{1\le i\le z+1} y_i - \gamma y_{z+1} = \gamma \sum_{1\le i\le z} y_i.$$

Applying the induction hypothesis to d_i and y_i, we have $\sum_{i=1}^{z} w_i d_i \le \gamma \sum_{i=1}^{z} w_i y_i$. Thus, we have $\sum_{i=1}^{z+1} w_i x_i \le \gamma \sum_{i=1}^{z} w_i y_i + \gamma w_{z+1} y_{z+1} = \gamma \sum_{i=1}^{z+1} w_i y_i$. The induction is complete. □

In `Scatter_&_Squash`, Band 1b uses an arbitrary algorithm with $(2k+1)m$ machines to process jobs during their earliest processing periods. We enhance Band 1b by selecting jobs with largest weights. We call this new algorithm EPPHWF (earliest processing period, highest weight first), and denote it as EPPHWF$\langle h \rangle$ when it is equipped with hm processors, where h is an integer at least 2. Intuitively, jobs with large weights will wait less. Let OPT be the optimal algorithm (using m unit-speed machines) for minimizing weighted flow time. Our key observation is that for any job weight w, we can bound the total waiting time of all jobs with weight at least w in EPPHWF$\langle 2k+1\rangle$ to be at most $1/(2k)$ times that of OPT. Then, we can make use of Lemma 15 inductively to show that EPPHWF$\langle 2k+1\rangle$ incurs a total weighted waiting time at most $1/(2k)$ times that of OPT. Details are as follows.

LEMMA 16. *Let $h \geq 2$ be an integer. For any job sequence I, the total weighted waiting time incurred by* EPPHWF$\langle h\rangle$ *is at most $\frac{1}{h-1}$ times that of OPT.*

Proof. Consider any job sequence I. We compare the schedules of EPPHWF$\langle h\rangle$ and OPT. Let $w_1 > w_2 \cdots > w_r$ be the distinct weights of the jobs in I. Consider any integer $q \in \{1, 2, \dots, r\}$. With respect to the schedule of EPPHWF$\langle h\rangle$, let $L(I)|_{w_q}$ be the total *unweighted* waiting time incurred on jobs with weight exactly w_q. $L_{OPT}(I)|_{w_q}$ is defined similarly for the schedule of OPT. Note that for jobs with weight at least w_q, the total weighted waiting time incurred by EPPHWF$\langle h\rangle$ and OPT is $\sum_{i=1}^{q} w_i L(I)|_{w_i}$ and $\sum_{i=1}^{q} w_i L_{OPT}(I)|_{w_i}$, respectively.

We first focus on the unweighted waiting time. Let $I_q \subseteq I$ be the set of jobs having weight at least w_k, where $q \in \{1, \dots, r\}$. EPPHWF$\langle h\rangle$ does not change the schedule of the jobs in I_q when jobs with less weight are removed, so EPPHWF$\langle h\rangle$, when scheduling I_q alone, incurs a total (unweighted) waiting time of $\sum_{i=1}^{q} L(I)|_{w_i}$. EPPHWF is a special case of EPPBUSY, so by Corollary 3.2, we have $\sum_{i=1}^{q} L(I)|_{w_i} \leq \frac{1}{h-1} L_{OPT(I_q)}$, where $L_{OPT(I_q)}$ is the total waiting time in the optimal (unweighted) schedule for I_q. When scheduling I, the waiting time incurred by OPT on the jobs in I_q is $\sum_{i=1}^{q} L_{OPT}(I)|_{w_i}$, which is at least $L_{OPT(I_q)}$. Therefore, for each $q = 1, \dots, r$,

$$\sum_{i=1}^{q} L(I)|_{w_i} \leq \frac{1}{h-1} \sum_{i=1}^{q} L_{OPT}(I)|_{w_i}.$$

By Lemma 15, we transform the above relation of unweighted waiting times to a weighted version:

$$\sum_{i=1}^{r} w_i L(I)|_{w_i} \leq \frac{1}{h-1} \sum_{i=1}^{r} w_i L_{OPT}(I)|_{w_i}.$$

Note that the former is the weighted waiting time of EPPHWF$\langle h\rangle$, and $\sum_{i=1}^{r} w_i L_{OPT} \cdot (I)|_{w_i}$ is the weighted waiting time of OPT. The lemma follows. □

Transformation that preserves weighted waiting time. Let `weighted_SS` be the algorithm `Scatter_&_Squash` with Band 1b using EPPHWF instead of EPPBUSY. Given an α-speed algorithm $A(\alpha)$, `weighted_SS` transforms $A(\alpha)$ to a τ-machine algorithm (recall that $\tau = 7\alpha + 5k - 2$, where $k \geq 1$). Band 1a and Band 2 have the same performance as before. Specifically, for each job, Band 1a still incurs a waiting time (and weighted waiting time) the same as $A(\alpha)$, and Band 2 incurs a waiting time (and weighted waiting time) no more than Band 1 does. By Lemma 16, the total weighted waiting time incurred by Band 1b, which uses $(2k+1)$ machines, is at most $\frac{1}{2k}$ times that of OPT. Thus, the total weighted waiting time has the same bound as before.

LEMMA 17. *Let $\tau = 7\alpha + 5k - 2$ for any integer $k \geq 1$. The weighted waiting time incurred by* `weighted_SS`$(A(\alpha), \tau)$ *is at most* 2 *times that of* $A(\alpha)$ *plus* $1/k$ *times that of* OPT.

Suppose there is an algorithm A that is $O(s)$-speed $\frac{1}{s}$-competitive for weighted waiting time, for any $s \geq 1$. Then by Lemma 17, we can derive an algorithm that is

$O(1)$-machine 1-competitive for weighted waiting time or, equivalently, for weighted flow time. However, such an algorithm A is not known to exist. Even if we restrict our attention to normalized waiting time, we do not know any algorithm that is $O(s)$-speed $\frac{1}{s}$-competitive and that can be used to derive an $O(1)$-machine 1-competitive algorithm for stretch.

Alternative transformation that preserves weighted waiting time. The rest of this paper shows another way to obtain an algorithm that is $O(1)$-machine 1-competitive for stretch. In the next section, we will show a weaker result on using a faster processor to improve the normalized waiting time; specifically, we prove that an algorithm based on SJF is $(1/s)$-competitive when using 4m machines that are $8s$-speed, where $s \geq 1$. To ease our discussion, we say this algorithm is (4-machine, $8s$-speed) $\frac{1}{s}$-competitive. (Note that this result does not imply an algorithm that is $32s$-speed $\frac{1}{s}$-competitive for normalized waiting time.[2]) Furthermore, we can extend the transformation result in Lemma 17 so that the input algorithm to `weighted_SS`, denoted $A[\ell, \alpha]$, uses ℓm machines that are α-speed, where $\ell \geq 1$ is an integer. In this case, Band 1a uses ℓm machines, Band 1b uses $(2k+1)m$ machines, and Band 2 uses $\lceil 7\ell(\alpha-1)m + \frac{3}{2}(\ell+2k+1)m \rceil$ machines. The proof of Lemma 5 can be easily generalized to show that

- for each job, Band 1a still incurs a weighted waiting time the same as $A[\ell, \alpha]$ does, and Band 2 incurs a weighted waiting time no more than Band 1 does; and
- the total weighted waiting time incurred by Band 1b is at most $\frac{1}{2k}$ times that of OPT.

Thus, we have the following result.

LEMMA 18. *Let* $\tau = 7\alpha\ell + 5k - \frac{9}{2}\ell + \frac{5}{2}$, *where k is any positive integer. The weighted waiting time incurred by* `weighted_SS`$(A[\ell, \alpha], \tau)$ *is at most* 2 *times that of* $A[\ell, \alpha]$ *plus* $1/k$ *times that of* OPT.

In the next section, we show that, for the special case of normalized waiting time, SJF is (4-machine, $8s$-speed) $\frac{1}{s}$-competitive for any $s \geq 1$. Choosing s to be 2.2 and applying Lemma 18 with $k = 11$, $\ell = 4$, and $\alpha = 8 \times 2.2$, we obtain an algorithm that is 533-machine 1-competitive for normalized waiting time, as well as the following result on stretch.

COROLLARY 19. *Based on SJF,* `weighted_SS` *gives a* 533*-machine* 1*-competitive algorithm for stretch.*

Remark. We conjecture that HDF is ($O(1)$-machine, $O(s)$-speed) $\frac{1}{s}$-competitive for weighted waiting time, for any $s \geq 1$. If this can be proven, then Lemma 18 can transform HDF to an algorithm that is $O(1)$-machine 1-competitive for weighted waiting time, as well as for weighted flow time.

5. Improved analysis of SJF with faster machines.

This section proves that SJF is (4-machine, $8s$-speed) $\frac{1}{s}$-competitive for normalized waiting time, where $s \geq 1$. The proof is divided into two parts. First, we show how to make use of a result by Becchetti et al. [6] to show that SJF is (2-machine, 4-speed) 2-competitive for normalized waiting time. Next, we show a scaling lemma for SJF by which the waiting time of each job can be reduced by s times if a double number of s times faster machines are used. That is, for any $s \geq 1$, SJF$[2\ell, s\alpha]$, when compared with SJF$[\ell, \alpha]$, reduces the waiting time of every job by at least s times, where $s \geq 1$.

[2] Roughly speaking, if we use a four-times faster machine to simulate four unit-speed machines by time sharing, the flow time of each job is preserved, but the actual time to process a job is shortened by four times. Thus, the waiting time is longer than before.

Then, the fact that SJF is (2-machine, 4-speed) 2-competitive for normalized waiting time would imply that SJF is also (4-machine, $8s$-speed) $\frac{1}{s}$-competitive for normalized waiting time, for any $s \geq 1$.

5.1. SJF and normalized waiting time.

First, we observe the following result by Becchetti et al. [6] on using HDF(4) (i.e., HDF with m 4-speed machines) to schedule jobs with arbitrary weights. Below the notation HDF (or any algorithm) is overloaded to mean the algorithm itself as well as the schedule defined by HDF.

Lemma 20 (see [6]). *Let I be a job sequence with arbitrary weights. Let A be any schedule of I using m unit-speed machines. Consider the two schedule A and the schedule of I in accordance with HDF(4). At any time t, the total weight of the remaining jobs in HDF(4) is at most two times that of A.*

The above lemma implies that in the unweighted setting, at any time, the number of remaining jobs in SJF(4) is at most two times that of A. Furthermore, we have the following result.

Lemma 21. SJF *is (2-machine, 4-speed) 2-competitive for (unweighted) waiting time.*

Proof. Consider any job sequence I. Let A be any schedule of I using m unit-speed machines. At any time t, let $U_t(\text{SJF}[2,4])$ be the number of jobs remaining in SJF$[2,4]$, and define $U_t(\text{SJF}(4))$ and $U_t(A)$ similarly. Then, $U_t(\text{SJF}[2,4]) \leq U_t(\text{SJF}(4)) \leq 2 \times U_t(A)$. Note that SJF$[2,4]$ is using $2m$ machines. At time t, if there is a job waiting in SJF$[2,4]$, then $U_t(\text{SJF}[2,4]) > 2m$, and the number of jobs waiting is $U_t(\text{SJF}[2,4]) - 2m \leq 2(U_t(A) - m)$. Thus, at any time, the number of jobs waiting in SJF$[2,4]$ at most two times of that of A, and the lemma follows. □

Intuitively, SJF gives priority to smaller jobs, and it is competitive not only for the total waiting time, but also for the waiting time of small jobs only. This allows us to derive inductively a bound of the waiting time. Then, using Lemma 15, we transform this bound to a bound on the total normalized waiting time. Details are as follows.

Lemma 22. SJF *is (2-machine, 4-speed) 2-competitive for normalized waiting time.*

Proof. Let I be any job sequence, and let $w_1 < w_2 \cdots < w_r$ be the distinct job sizes in I. Consider the schedule of I in accordance with SJF$[2,4]$, and let $L(I)|_{w_i}$ be the total waiting time of jobs with size exactly w_i. Denote OPT be the optimal algorithm using m unit-speed machines, and define $L'(I)|_{w_i}$ similarly for OPT. Consider any $q \in \{1, 2, \ldots, r\}$. Let I_q be the job sequence including only jobs in I with size w_1, w_2, …, or w_q. Since SJF does not change the schedule of a job due to other jobs with larger size, $\sum_{i=1}^{q} L(I_q)|_{w_i}$ (i.e., the total waiting time incurred by SJF$[2,4]$ on I_q) is exactly equal to $\sum_{i=1}^{q} L(I)|_{w_i}$. By Lemma 21, the total waiting time incurred by SJF$[2,4]$ is at most two times the total waiting time incurred by any schedule of I_q on m unit-speed machines. Thus, $\sum_{i=1}^{q} L(I)|_{w_i} = \sum_{i=1}^{q} L(I_q)|_{w_i} \leq 2 \times \sum_{i=1}^{q} L'(I)|_{w_i}$, where $q = 1, 2, \ldots, r$.

Again, we make use of Lemma 15 to turn the above result into a weighted one: $\sum_{i=1}^{r} \frac{1}{w_i} L_s(I)|_{w_i} \leq 2 \sum_{i=1}^{r} \frac{1}{w_i} L_o(I)|_{w_i}$, or, equivalently, the normalized waiting time of SJF$(2,4)$ is at most two times that of OPT. □

5.2. The scaling lemma.

This section shows that the waiting time incurred by SJF can be scaled down with increasing speed. Precisely, we compare SJF$[\ell, \alpha]$ and SJF$[2\ell, c\alpha]$, where $c \geq 1$ is a real, and we show that the waiting time of each job decreases by c times (note that this result is much stronger than bounding the total waiting time). More interestingly, this result is true for the more general algorithm

HDF (see the lemma below). Note that the density of a job, defined as the ratio of its weight to its processing time, is fixed throughout the life span of a job. HDF schedules jobs with highest densities (we assume that ties are broken by job IDs). SJF is identical to HDF when all jobs are assumed to have a unit weight.

Lemma 23. *Consider any job sequence I with arbitrary weights. For each job J in I, denote the waiting time of J incurred by* $\mathrm{HDF}[2\ell, c\alpha]$ *and* $\mathrm{HDF}[\ell, \alpha]$ *as $L_{(2,c)}(J)$ and $L_{(1,1)}(J)$, respectively. Then $L_{(2,c)}(J) \leq \frac{1}{c} L_{(1,1)}(J)$.*

Proof. Denote $S_1(I)$ and $S_2(I)$ as the schedule of a job sequence I in accordance with $\mathrm{HDF}[\ell, \alpha]$ and $\mathrm{HDF}[2\ell, c\alpha]$, respectively. By definition of HDF, at any time, every job has less remaining work in $\mathrm{HDF}[2\ell, c\alpha]$ than in $\mathrm{HDF}[\ell, \alpha]$, and if a job is waiting in $\mathrm{HDF}[2\ell, c\alpha]$, then the job must also be waiting in $\mathrm{HDF}[\ell, \alpha]$.

Consider a job J in I. Assume that J is completed at time $z(J)$ in $S_1(I)$, and hence no later than $z(J)$ in $S_2(I)$. We call jobs in I that have higher densities than J the *higher-priority* jobs. At any time, $S_1(I)$ (as well as $S_2(I)$) is said to be *busy* if all available machines are running some higher-priority jobs. Note that at any time t in $[r(J), z(J)]$, J is waiting in $S_1(I)$ if and only if $S_1(I)$ is busy. So the waiting time of J in $S_1(I)$ is the total length of the busy periods of $S_1(I)$ during $[r(J), z(J)]$. Denote these busy periods $\lambda_1, \lambda_2, \ldots, \lambda_h$.

Next, we consider the schedule $S_2(I)$. Suppose that J is waiting in $S_2(I)$ during a time interval ρ. Note that ρ is a busy period in $S_2(I)$; furthermore, J is also waiting in $S_1(I)$ during ρ, and ρ is a subinterval of some busy period λ of $S_1(I)$. Therefore, to upper bound the waiting time of J in $S_2(I)$, it suffices to consider each busy period λ of $S_1(I)$ separately. Below we show that the busy period of $S_2(I)$ within λ has a total length at most a fraction $\frac{1}{c}$ of λ (see the lemma below). Then we can conclude that the waiting time of J in $S_2(I)$, which is the total length of the busy periods within $\lambda_1, \lambda_2, \ldots, \lambda_h$, is at most $\frac{1}{c}(|\lambda_1| + |\lambda_2| + \cdots + |\lambda_h|)$ or, equivalently, $\frac{1}{c}$ times the waiting time of J in $S_1(I)$. Lemma 23 follows. □

It remains to prove the following lemma.

Lemma 24. *Let $\lambda = [t_1, t_2]$ be a busy period in $S_1(I)$. Denote the busy periods of $S_2(I)$ that are within $[t_1, t_2]$ as $\rho_1, \rho_2, \ldots, \rho_g$, and let $y = \sum_{1 \leq i \leq g} |\rho_i|$ be their total length. Then $y \leq \frac{1}{c}|\lambda|$.*

Proof. Let $\lambda = [t_1, t_2]$. The work scheduled by $S_1(I)$ during λ, denoted by W, has a total size exactly $\ell m \alpha |\lambda|$. Let R be the high-priority jobs remaining in $S_1(I)$ immediately after t_2. Note that R contains at most $\ell m - 1$ jobs as $S_1(I)$ is not busy immediately after t_2.

During the busy periods $\rho_1, \rho_2, \ldots, \rho_g$, the total amount of work processed in $S_2(I)$ is $2\ell m c \alpha y$; on the other hand, the high-priority work available to $S_2(I)$ is limited. In particular, $S_2(I)$ can process at most all the work W and some of the work of R (that are processed beyond t_2 in $S_1(I)$). The former has a total amount of $\ell m \alpha |\lambda|$, and the latter is bounded by $c\alpha y |R| < c\alpha y \ell m$. Thus, we have

$$2\ell m c \alpha y < \ell m \alpha |\lambda| + c \alpha y \ell m$$

or, equivalently, $y \leq |\lambda|/c$. □

Lemma 23 implies that when comparing $\mathrm{HDF}[2\ell, c\alpha]$ against $\mathrm{HDF}[\ell, \alpha]$, the weighted waiting time, flow time, and weighted flow time of each job decrease by c times. In particular, together with Lemma 22, Lemma 23 gives the main result of this section.

Corollary 25. SJF *is (4-machine, $8s$-speed) $\frac{1}{s}$-competitive for normalized waiting time, where $s \geq 1$ is any real.*

Proof. By Lemma 22, SJF is (2-machine, 4-speed) 2-competitive for normalized waiting time. By Lemma 23, if we double the number of machines and increase the speed $2s$ times, then the competitive ratio is reduced to $1/s$, where $s \geq 1$. □

Lemma 23 has another implication. It is known that HDF is 4-speed 2-competitive for weighted flow time [6]. Again, if we double the number of machines and increase the speed $2s$ times for any $s \geq 1$, then the competitive ratio is reduced to $1/s$.

LEMMA 26. *Let $s \geq 1$ be any real number.* HDF *is $(2, 8s)$-machine-speed $(1/s)$-competitive for weighted flow time.*

The above result also implies an algorithm that is $16s$-speed $(1/s)$-competitive for weighted flow time (by simulating HDF$(2, 8s)$ using time-sharing).

6. Conclusion. This paper serves as the first step in understanding how extra-machine analysis is related to extra-speed analysis, and how extra machines can provide 1-competitive scheduling for minimizing flow time and stretch. There are several interesting problems to be addressed. We do not have a similar result for weighted flow time. Unlike the algorithm IMD [13], our new algorithms incorporate SRPT or HDF, and they are migratory in nature and do not allow immediate dispatch. It is interesting to investigate nonmigratory algorithms with similar performances. Another important direction is to minimize the L_p norm of flow time and stretch [5, 13]. Note that Chekuri et al. [13] have extended $(1+\epsilon)$-speed (or $(1+\epsilon)$-machine) $O(1 + 1/\epsilon)$-competitive results for flow time and stretch to the L_p norm.

REFERENCES

[1] N. Avrahami and Y. Azar, *Minimizing total flow time and total completion time with immediate dispatching*, in Proceedings of SPAA, ACM, New York, 2003, pp. 11–18.

[2] B. Awerbuch, Y. Azar, S. Leonardi, and O. Regev, *Minimizing the flow time without migration*, in Proceedings of STOC, ACM, New York, 1999, pp. 198–205.

[3] N. Bansal, *On minimizing the total flow time on multiple machines*, in Proceedings of SODA, ACM, New York, SIAM, Philadelphia, 2004, pp. 572–574.

[4] N. Bansal and K. Dhamdhere, *Minimizing weighted flow time*, in Proceedings of SODA, ACM, New York, SIAM, Philadelphia, 2003, pp. 508–516.

[5] N. Bansal and K. Pruhs, *Server scheduling in the L_p norm: A rising tide lifts all boat*, in Proceedings of STOC, ACM, New York, 2003, pp. 242–250.

[6] L. Becchetti, S. Leonardi, A. Marchetti-Spaccamela, and K. Pruhs, *Online weighted flow time and deadline scheduling*, in Proceedings of RANDOM-APPROX, Springer-Verlag, Berlin, 2001, pp. 36–47.

[7] L. Becchetti, S. Leonardi, and S. Muthukrishnan, *Scheduling to minimize average stretch without migration*, in Proceedings of SODA, ACM, New York, SIAM, Philadelphia, 2000, pp. 548–557.

[8] M. A. Bender, S. Chakrabarti, and S. Muthukrishnan, *Flow and stretch metrics for scheduling continuous job streams*, in Proceedings of SODA, ACM, New York, SIAM, Philadelphia, 1998, pp. 270–279.

[9] M. A. Bender, S. Muthukrishnan, and R. Rajaraman, *Improved algorithms for stretch scheduling*, in Proceedings of SODA, ACM, New York, SIAM, Philadelphia, 2002, pp. 762–771.

[10] P. Berman and C. Coulston, *Speed is more powerful than clairvoyance*, in Proceedings of the 6th Annual SWAT, Stockholm, Sweden, 1998, pp. 255–263.

[11] H. L. Chan, T. W. Lam, and K. K. To, *Non-migratory online deadline scheduling on multiprocessors*, in Proceedings of SODA, ACM, New York, SIAM, Philadelphia, 2004, pp. 970–979.

[12] W. T. Chan, T. W. Lam, K. S. Liu, and P. Wong, *New resource augmentation analysis of the total stretch of SRPT and SJF in multiprocessor scheduling*, in Proceedings of MFCS, Springer-Verlag, Berlin, 2005, pp. 236–247.

[13] C. Chekuri, A. Goel, S. Khanna, and A. Kumar, *Multi-processor scheduling to minimize flow time with ϵ resource augmentation*, in Proceedings of STOC, ACM, New York, 2004, pp. 363–372.

[14] C. Chekuri, S. Khanna, and A. Zhu, *Algorithms for minimizing weighted flow time*, in Proceedings of STOC, ACM, New York, 2001, pp. 84–93.

[15] J. Du, J. Y. T. Leung, and G. H. Young, *Minimizing mean flow time with release time constraint*, Theoret. Comput. Sci., 75 (1990), pp. 347–355.

[16] J. Edmonds, *Scheduling in the dark*, in Proceedings of STOC, ACM, New York, 1999, pp. 179–188.

[17] B. Kalyanasundaram and K. Pruhs, *Speed is as powerful as clairvoyance*, J. ACM, 47 (2000), pp. 617–643.

[18] C. Y. Koo, T. W. Lam, T. W. Ngan, and K. K. To, *Extra processors versus future information in optimal deadline scheduling*, in Proceedings of SPAA, ACM, New York, 2002, pp. 133–142.

[19] K. R. Baker, *Introduction to Sequencing and Scheduling*, Wiley, New York, 1974.

[20] S. Leonardi and D. Raz, *Approximating total flow time on parallel machines*, in Proceedings of STOC, ACM, New York, 1997, pp. 110–119.

[21] J. McCullough and E. Torng, *SRPT optimally utilizes faster machines to minimize flow time*, in Proceedings of SODA, ACM, New York, SIAM, Philadelphia, 2004, pp. 343–351.

[22] S. Muthukrishnan, R. Rajaraman, A. Shaheen, and J. Gehrke, *Online scheduling to minimize average stretch*, in Proceedings of FOCS, IEEE Computer Society, Los Alamitos, CA, 1999, pp. 433–442.

[23] C. A. Phillips, C. Stein, E. Torng, and J. Wein, *Optimal time-critical scheduling via resource augmentation*, in Proceedings of STOC, ACM, New York, 1997, pp. 140–149.

[24] K. Pruhs, J. Sgall, and E. Torng, *Online scheduling*, in Handbook of Scheduling: Algorithms, Models and Performance Analysis, J. Leung, ed., CRC Press, Boca Raton, FL, 2004, pp. 15-1–15-41.

SIAM J. COMPUT.
Vol. 37, No. 5, pp. 1613–1636

RANDOMIZATION DOES NOT REDUCE THE AVERAGE DELAY IN PARALLEL PACKET SWITCHES*

HAGIT ATTIYA† AND DAVID HAY†

Abstract. Switching cells in parallel is a common approach to building switches with very high external line rates and a large number of ports. A prime example is the *parallel packet switch* (PPS) in which a demultiplexing algorithm sends cells, arriving at rate R on N input-ports, through one of K intermediate slower switches, operating at rate $r < R$. In order to utilize the parallelism of the PPS, a key issue is to balance the load among the planes; since *randomization* is known as a successful paradigm to solve load balancing problems, it is tempting to design randomized demultiplexing algorithms that balance the load on the average. This paper presents lower bounds on the *average* queuing delay introduced by the PPS relative to an optimal work-conserving first-come first-serve (FCFS) switch for *randomized* demultiplexing algorithms that do not have full and immediate information about the switch status. These lower bounds are shown to be asymptotically optimal through a methodology for analyzing the *maximal* relative queuing delay by measuring the imbalance between the middle stage switches; clearly, this also bounds (from above) the *average* relative queuing delay. The methodology is used to devise new algorithms that rely on slightly outdated global information on the switch status. It is also used to provide, for the first time, a complete proof of the maximum relative queuing delay provided by the *fractional traffic dispatch* algorithm [S. Iyer and N. McKeown, in *Proceedings of IEEE INFOCOM*, IEEE Communications Society, New York, NY, 2001, pp. 1680–1687; D. Khotimsky and S. Krishnan, in *Proceedings of the IEEE International Conference on Communications*, IEEE Communications Society, New York, NY, 2001, pp. 100–106]. These optimal algorithms are deterministic, proving that randomization does not reduce the relative queuing delay of the PPS.

Key words. load balancing, randomization, packet switching, Clos networks, queuing delay, inverse multiplexing

AMS subject classifications. 68W10, 68W15, 68W20

DOI. 10.1137/050648250

1. Introduction.

Parallelism often increases the throughput of a system by distributing tasks among several processing entities. Careful *load balancing* is required to ensure even distribution and guarantee small delay for each task. *Randomization* is an attractive paradigm for balancing the load on the average [4, 29]: even a very simple strategy ensures (with high probability) maximum load close to the optimal distribution [17].

Load balancing has recently been employed in packet switching architectures with high line-rates and large numbers of ports [6, 35, 34, 33, 22]. A successful example of such a switch is the *parallel packet switch (PPS)* [19], which is the core of several contemporary switches (e.g., [1, 11, 28, 31]). Like *inverse multiplexing* systems [2, 9, 14, 15], an $N \times N$ PPS demultiplexes cells, arriving at rate R, through K slower switches (*planes*) operating at rate $r < R$ (see Figure 1).

A PPS is evaluated by the *queuing delay* it introduces *relative* to an *optimal work-conserving shadow* switch that receives the same incoming traffic. A work-conserving switch guarantees that an output-port is never idle unnecessarily and, by

*Received by the editors December 22, 2005; accepted for publication (in revised form) July 11, 2007; published electronically February 14, 2008. Some of the results appeared, in a preliminary form, in *Proceedings of the 17th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2005)*, pp. 11–20.

http://www.siam.org/journals/sicomp/37-5/64825.html

†Department of Computer Science, Technion—Israel Institute of Technology, Haifa 32000, Israel (hagit@cs.technion.ac.il, hdavid@cs.technion.ac.il).

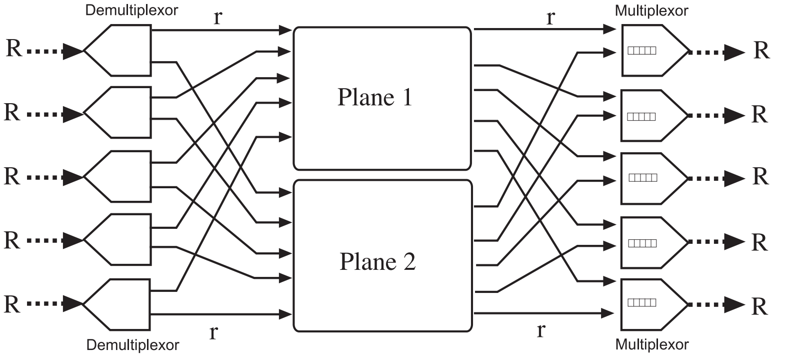


FIG. 1. *A 5×5 PPS with two planes in its center stage, without buffers in the input-ports.*

that, maximizes the switch throughput and minimizes its average cell delay [10, 26, 27]. This generalizes the common practice of comparing with an output-queued switch [8, 19, 20, 25, 27, 32, 36, 37]. This comparison is not burdened by unsubstantiated probabilistic assumptions on the incoming traffic [13] and reveals inherent strengths and weaknesses of the PPS architecture.

As we shall prove, the relative queuing delay is determined by the balancing of cells among the planes. Given the successful application of randomization in traditional load balancing settings [4, 17, 29] and in other high-bandwidth switches [16, 38], it is tempting to employ randomization to reduce the average imbalance between planes and by that reduce the average relative queuing delay.

This paper shows that randomization does not help to decrease the average relative queuing delay. This result holds due to the requirement that switches should not mis-sequence cells [23]. This property allows an adversary to exploit a transient increase of the relative queuing delay and perpetuate it sufficiently to increase the *average* relative queuing delay.

Specifically, we show that an adversary can devise traffic that exhibits with high probability a large average relative queuing delay. The exact bounds depend on the locality of information used for cell demultiplexing, the type of the adversary, and the exact restriction on the order of cells the switch should respect. The bounds are equal to the lower bounds known for maximum relative queuing delay [3]: If a PPS respects the arrival order of cells with the same input-port and the same output-port and the adversary is *adaptive* [30], the lower bound is $\Omega(\min\{u, \frac{R}{r}\} \cdot N)$ time-slots for algorithms that use global information older than u time-slots (namely, u real-time distributed algorithms [3]) and $\Omega(\frac{R}{r}N)$ time-slots for algorithms that use only local information. The latter lower bound also holds with an *oblivious* adversary [5] if a PPS obeys a *global* first-come first-serve (FCFS) policy (that is, all cells to the same destination should leave the switch according to their arrival order).

To prove that these bounds are tight, we devise a methodology for evaluating the relative queuing delay under global FCFS policies. We show a general upper bound that depends on the difference between the number of cells with the same destination that are sent through a specific plane and the total number of cells with this destination.

Our methodology is employed to prove that the maximal relative queuing delay of the *fractional traffic dispatch (FTD)* algorithm [20] is $O(N\frac{R}{r})$ time-slots. This matches the lower bound on the average relative queuing delay introduced by fully distributed demultiplexing algorithms (even when randomization can be used). This is the first formal and complete correctness proof for this algorithm. Iyer and McKeown [21, 20] outline an approach for bounding the relative queuing delay of FTD but leave a number of details missing [18]; a previous attempt [25] to complete the formal proof and precisely bound the relative queuing delay of FTD turned out to be flawed [24].

By precisely capturing the crucial factors affecting the relative queuing delay, our methodology leads to new algorithms that use global information that is u time-slot old. Their maximum relative queuing delay is $O(N)$ time-slots, asymptotically matching the lower bound on the average relative queuing delay for this class of demultiplexing algorithms (even when randomization can be used).

2. The bufferless PPS model.

A *switch* handles fixed-size *cells* that arrive and leave at rate R in discrete *time-slots*: Each cell c arrives at time $ta(c)$ to input-port $orig(c)$, and it is destined for output-port $dest(c)$. We assume the switch does not drop cells.

A *traffic* T is a finite collection of cells, such that no two cells arrive at the same input-port at the same time-slot. A *flow* (i, j) is the collection of cells sent from input-port i to output-port j. The *projection* of a traffic T on a set of input-ports I, denoted by $T|_I$, is $\{c \in T \mid orig(c) \in I\}$. Since for any input-port i and traffic T, there are no two cells $c_1, c_2 \in T|_i$ such that $ta(c_1) = ta(c_2)$, the arrival times of cells in $T|_i$ induce a total order on them.

For any cell c, $\text{shift}(c, t)$ is a cell with the same origin and destination such that $ta(\text{shift}(c, t)) = ta(c) + t$. The *shift* operation is used for concatenating two finite traffics, T_1 and T_2, so that T_2 starts after the last cell of traffic T_1. Formally, $T_1 \circ T_2$ is the traffic $T_1 \cup \{\text{shift}(c, t) \mid c \in T_2\}$, where $t = 1 + \max\{ta(c) \mid c \in T_1|_{I_1}\}$.

An $N \times N$ *PPS* is a three-stage Clos network [12] with $K < N$ planes. Each plane is an $N \times N$ switch operating at rate $r < R$ and is connected to all input-ports on one side and to all output-ports on the other side (see Figure 1). The *speedup* $S = \frac{Kr}{R}$ captures the switch over-capacity.

A *bufferless* PPS has no buffers at its input-ports but can store pending cells in its planes and in its output-ports. Each cell arriving at input-port i is immediately sent to one of the planes; the plane through which the cell is sent is determined by a randomized state machine with state set $\mathbb{S}_i$, following some algorithm.

Definition 2.1. *The demultiplexing algorithm of a bufferless input-port i is a function*

$$\mathtt{ALG}_i : \{1, \ldots, N\} \times \mathbb{S}_i \times \mathtt{COINSPACE} \to \{1, \ldots, K\} \times \mathbb{S}_i$$

which gives a plane number and the next state, according to the incoming cell destination, the current state, and the result of a coin-toss that is taken out of a finite and uniform coin-space COINSPACE. (For a deterministic algorithm, $|\mathtt{COINSPACE}| = 1$.)

$\mathcal{E}_{PPS}(\mathtt{ALG}, \sigma, T)$ is the *execution* of a PPS using demultiplexing algorithm ALG in response to incoming traffic T and coin-toss sequence σ; for all cells in T, the execution indicates the planes the cells are sent through: $\{\langle c, plane(c, T)\rangle \mid c \in T\}$.

A state $s \in \mathbb{S}_i$ is *reachable* if there is a sequence of coin tosses σ and a traffic T, such that the state-machine reaches state s in execution $\mathcal{E}_{PPS}(\mathtt{ALG}, \sigma, T)$. A switch *configuration* consists of the states of all state-machines and the contents of all the buffers in the switch at a given time. A configuration is *reachable* if it is reached in an execution of the switch. Since the switch does not have a predetermined initial configuration, we assume that, for every pair of reachable configurations C_1, C_2, there

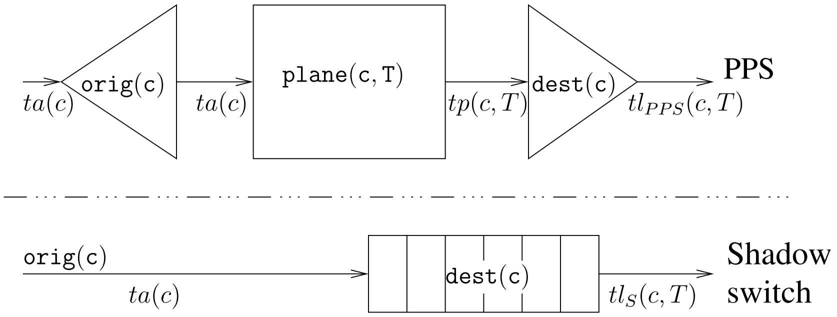


FIG. 2. *Time-points associated with a cell $c \in T$.*

is a finite incoming traffic that causes the switch a transition from C_1 to C_2.

The internal lines of the switch operate at rate $r < R$. For simplicity, we assume that $r' \triangleq \frac{R}{r} = \lceil \frac{R}{r} \rceil$. This lower rate r imposes an *input constraint* on the demultiplexing algorithm [19].

For any two cells c_1, c_2 in traffic T such that $orig(c_1) = orig(c_2)$ and $plane(c_1, T) = plane(c_2, T)$, $|ta(c_1) - ta(c_2)| > r'$.

Since a PPS has no buffers in its input-ports, cells are immediately sent to one of the planes; that is, a cell c traverses the internal link between $orig(c)$ and $plane(c, T)$ at time $ta(c)$ (see Figure 2).

We assume that both the planes and the output buffers are FCFS and work-conserving. Let $tp(c, T)$ be the time-slot in which a cell $c \in T$ leaves $plane(c, T)$, and denote $tl_{PPS}(c, T)$ the time-slot it leaves the PPS. The lower rate of the internal links between the planes to the output-ports imposes an *output-constraint* [19].

For every two cells $c_1, c_2 \in T$, if $dest(c_1) = dest(c_2)$ and $plane(c_1, T) = plane(c_2, T)$ then $|tp(c_1, T) - tp(c_2, T)| > r'$.

To neglect delays caused by the additional stage of the PPS, a cell can leave the PPS at the same time-slot it arrives at the output-port, provided that no other cell is leaving at this time-slot, i.e., $tl_{PPS}(c, T) \geq tp(c, T)$. Note, however, that $tp(c, T) \geq ta(c) + 1$.

A PPS is compared to a work-conserving shadow switch that receives the same traffic T and obeys the *per-flow FCFS* discipline; that is, cells with the same origin and the same destination should leave the switch in their arrival order. We denote the execution of the shadow switch in response to traffic T by $\mathcal{E}_S(T)$ and the time a cell $c \in T$ leaves the shadow switch by $tl_S(c, T)$. Note that $tl_S(c, T) \geq ta(c) + 1$.

The *relative queuing delay* of a cell $c \in T$ under a demultiplexing algorithm ALG and a coin-toss sequence σ is $\mathcal{R}(\texttt{ALG}, \sigma, c, T) = tl_{PPS}(c, T) - tl_S(c, T)$.

DEFINITION 2.2. *For traffic T, demultiplexing algorithm ALG and coin-toss sequence σ, the* maximum relative queuing delay $\mathcal{R}_{max}(\texttt{ALG}, \sigma, T)$ *is* $\max_{c \in T}\{\mathcal{R}(\texttt{ALG}, \sigma, c, T)\}$*, and the* average relative queuing delay $\mathcal{R}_{avg}(\texttt{ALG}, \sigma, T)$ *is* $\frac{1}{|T|} \sum_{c \in T} \mathcal{R}(\texttt{ALG}, \sigma, c, T)$.

The maximum and the average relative queuing delays of an algorithm ALG against an adversary A are denoted $\mathcal{R}^A_{max}(\texttt{ALG})$ and $\mathcal{R}^A_{avg}(\texttt{ALG})$, respectively.

When it is clear from the context, we omit the traffic T from the notation $plane(c, T)$, $tp(c, T)$, $tl_{PPS}(c, T)$, $tl_S(c, T)$, and $\mathcal{R}(\texttt{ALG}, \sigma, c, T)$.

3. Lower bounds on the average relative queuing delay. In this section, we prove lower bounds on the *average* relative queuing delay even when randomization is used. Our first lower bounds use an adaptive adversary that sends cells to the switch at each time-slot based on the algorithm actions at previous slots. Then, we show that, under reasonable assumptions, the lower bounds can be extended to hold with an oblivious adversary [5] that chooses the entire traffic in advance, knowing only the demultiplexing algorithm.

We prove yet stronger results and show that the lower bounds hold even when the traffic is restricted by the (R, B)-*leaky-bucket* model [7, 13]. This model restricts the traffic from flooding the switches by requiring that the combined rate of flows sharing the same input-port or the same output-port does not exceed the external rate R of that port by more than a fixed bound B, called the burstiness factor of the traffic, which is independent of time. Specifically, a traffic T is (R, B)-leaky-bucket if, for any two time-slots $t_1 \leq t_2$ and any output-port j, $|\{c \in T \mid t_1 \leq ta(c) \leq t_2 \text{ and } dest(c) = j\}| \leq (t_2 - t_1) + B$.

3.1. General techniques and observations. A key observation is that if the last cell of a traffic attains relative queuing delay $\mathcal{R}$, then this traffic can be continued so that every added cell attains *at least* relative queuing delay $\mathcal{R}$, regardless of the random choices made by the demultiplexing algorithm.

We first define how a traffic is continued. A cell $c_2 \in T$ is the *immediate successor* of cell $c_1 \in T$ in demultiplexing algorithm ALG, denoted $c_2 = succ(c_1, T)$, if $tl_S(c_2, T) = tl_S(c_1, T) + 1$, and for every coin tosses sequence σ, $tl_{PPS}(c_2, T) > tl_{PPS}(c_1, T)$ in the execution $\mathcal{E}_{PPS}(\texttt{ALG}, \sigma, T)$. Namely, a PPS cannot change the order in which c_1 and c_2 are delivered; this happens, for example, when a PPS follows a per-flow FCFS policy and c_1, c_2 share the same input-port and the same output-port.

Let c be the last cell in a traffic T; i.e., $tl_S(c, T) = \max_{c' \in T}\{tl_S(c', T)\}$. A traffic $T' = \{c_0, \ldots, c_n\}$ is a *proper continuation* of T if, in the execution of the shadow switch in response to traffic $T \circ T'$, all the cells of T' are delivered one time-slot after the other without any stalls, and the delivery times of the cells of T remain unchanged. Formally, T' is a proper continuation of T if, in execution $\mathcal{E}_S(T \circ T')$, $c_0 = succ(c, T \circ T')$, $c_i = succ(c_{i-1}, T \circ T')$ for every i, and for every $c' \in T$, $tl_S(c', T) = tl_S(c', T \circ T')$ and $tl_{PPS}(c', T) = tl_{PPS}(c', T \circ T')$.

We first examine proper continuations by a single cell.

LEMMA 3.1. *For any demultiplexing algorithm ALG, coin-toss sequence σ, and finite traffic T, if c_1 is the last cell of T, and $T' = \{c_2\}$ is a proper continuation of T, then $\mathcal{R}(\texttt{ALG}, \sigma\theta, c_2, T \circ T') \geq \mathcal{R}(\texttt{ALG}, \sigma, c_1, T)$ for any coin-toss θ.*

Proof. Since T' is a proper continuation of T, cell c_2 leaves the shadow switch exactly at time-slot $tl_S(c_1, T \circ T') + 1$, and in addition $tl_{PPS}(c_2, T \circ T') \geq tl_{PPS}(c_1, T \circ T') + 1$. Hence,

$$\begin{aligned}\mathcal{R}(\texttt{ALG}, \sigma\theta, c_2, T \circ T') &\geq tl_{PPS}(c_2, T \circ T') - tl_S(c_2, T \circ T') \\ &\geq (tl_{PPS}(c_1, T \circ T') + 1) - (tl_S(c_1, T \circ T') + 1) \\ &= tl_{PPS}(c_1, T) - tl_S(c_1, T) \ = \ \mathcal{R}(\texttt{ALG}, \sigma, c_1, T). \qquad \square\end{aligned}$$

If the adversary can construct, for every traffic, a proper continuation that is arbitrarily long, then it can construct a traffic that exhibits an average relative queuing delay that matches the maximum relative queuing delay. Intuitively, the adversary waits for a cell c that attains $\mathcal{R}_{max}$ and then sends many cells, which form a proper continuation (whose length depends on the number of cells that arrived before c).

LEMMA 3.2. *Fix an adversary A, demultiplexing algorithm ALG, a coin-toss sequence σ, and a finite traffic T whose last cell c has $\mathcal{R}(\texttt{ALG}, \sigma, c, T) = x$. If the adversary A can construct a proper continuation of traffic T, whose size is at least $\left\lceil |T| \frac{x - \varepsilon}{\varepsilon} \right\rceil$ (ε is an arbitrarily small constant), then $\mathcal{R}^A_{avg}(\texttt{ALG}) \geq x - \varepsilon$.*

Proof. Let ℓ be the number of cells in traffic T, and let T' be a proper continuation of T such that $|T'| = \lceil \ell \frac{x-\varepsilon}{\varepsilon} \rceil$. Applying Lemma 3.1 $|T'|$ times implies that for every cell b in T' and any coin-toss sequence σ_b, $\mathcal{R}(\texttt{ALG}, \sigma\sigma_b, b) \geq \mathcal{R}(\texttt{ALG}, \sigma, c) \geq x$. Hence,

$$\mathcal{R}^A_{avg}(\texttt{ALG}) \;\geq\; \frac{1}{\ell + \lceil \ell \frac{x-\varepsilon}{\varepsilon} \rceil} \left\lceil \ell \frac{x-\varepsilon}{\varepsilon} \right\rceil x \;\geq\; x - \varepsilon. \qquad \square$$

The specific construction of the proper continuation depends on the type of the adversary. Lemma 3.4 and Step 2 of Theorem 3.9 show such constructions.

High relative queuing delay is exhibited when cells that are supposed to leave the shadow switch one after the other are concentrated in a single plane. An execution $\mathcal{E}_{PPS}(\texttt{ALG}, \sigma, T)$ is (f, s)-*concentrating for output-port* j *and plane* k if there is a time-slot t such that

1. output-port j's buffer of the shadow switch is empty at time-slot t;
2. at least f cells destined for output-port j arrive to the switch during time-interval $[t, t+s)$, and f out of these cells are sent through the plane k; and
3. traffic T ends at time-slot $t + s$.

We call an execution an (f, s)-*concentrating* execution when the plane k and the output-port j are clear from the context.

The following lemma bounds the relative queuing delay exhibited in (f, s)-concentrating executions, extending [3, Lemma 4] for randomized demultiplexing algorithms.

Lemma 3.3. *For any* (R, B)*-leaky-bucket traffic* T*, coin-toss sequence* σ*, and* (f, s)*-concentrating execution* $\mathcal{E}_{PPS}(\mathit{ALG}, \sigma, T)$ *for output-port* j *and plane* k*, the last cell* c *that is sent from plane* k *to output-port* j *in* $\mathcal{E}_{PPS}(\mathit{ALG}, \sigma, T)$ *attains* $\mathcal{R}(\mathit{ALG}, \sigma, c, T) \geq f \cdot r' - (s + B)$.

Proof. We compare the queuing delay of the cells in the PPS and in the shadow switch. Since the shadow switch is work-conserving, all f cells leave the switch exactly f time-slots after the first cell is dispatched. On the other hand, a PPS completes this execution after at least fr' time-slots, because f cells are sent to the same plane, and only one cell can be sent from this plane to the output-port every r' time-slots. Let c be the last of these cells sent from the plane to the output-port. Hence, the relative queuing delay that c attains is at least $fr' - f$ time-slots. Since the incoming traffic is (R, B)-leaky-bucket, $f \leq s + B$, and therefore $\mathcal{R}(\texttt{ALG}, \sigma, c, T) \geq fr' - f \geq fr' - (s+B)$ time-slots. $\square$

We concentrate now on an adaptive adversary, denoted *adp*, which sends cells to the switch based on the algorithm actions.

For every traffic T, we examine the probability $\Pr_\sigma [\mathcal{E}_{PPS}(\texttt{ALG}, \sigma, T)$ is (f, s)-concentrating], taken over all coin-toss sequences σ, that the execution of ALG given T and σ is (f, s)-concentrating.

Another key observation is that if there is traffic T such that its execution is (f, s)-concentrating with small but nonnegligible probability, an adaptive adversary can construct another execution that is *almost always* (f, s)-concentrating.

Lemma 3.4. *If from every configuration* C *there is an* (R, B)*-leaky-bucket traffic* T *such that* $\Pr_\sigma [\mathcal{E}_{PPS}(\mathit{ALG}, \sigma, T)$ *is* (f, s)*-concentrating*$] \geq \widetilde{p} > 0$*, then an adaptive adversary can construct an* (R, B)*-leaky-bucket traffic* T' *from* C *such that* $\Pr_\sigma [\mathcal{E}_{PPS}(\mathit{ALG}, \sigma, T')$ *is* (f, s)*-concentrating*$] \geq 1 - \delta$*, where* δ *can be made arbitrarily small.*

Proof. Fix a configuration C; the adaptive adversary constructs the executions from C iteratively: Denote $C^0 \triangleq C$. Let C^i be the configuration just before iteration $i \geq 0$, and denote by T^i the traffic such that from configuration C^i,

$\Pr_\sigma\left[\mathcal{E}_{PPS}(\mathtt{ALG},\sigma,T^i) \text{ is } (f,s)\text{-concentrating}\right] \geq \widetilde{p}$. The adversary stops if the last execution is indeed (f,s)-concentrating. Otherwise, it concatenates an empty traffic of B time-slots (denoted T_e) and continues to the next iteration.

Since in each iteration the adversary stops with probability at least $\widetilde{p}$ independently of previous iterations, it stops with an (f,s)-concentrating execution at iteration $\ell \leq \lceil \log_{1-\widetilde{p}} \delta \rceil$ with probability $1-\delta$. Since there are B empty time-slots between the arrival of the last cell of traffic T^i and the arrival of the first cell in T^{i+1}, $T' = T^0 \circ T_e \circ \cdots \circ T_e \circ T^\ell$ has burstiness factor B, and its corresponding execution starting from C is (f,s)-concentrating with probability $1-\delta$. □

Since both the shadow switch and the PPS are per-flow FCFS, an adaptive adversary can always construct an arbitrarily long proper continuation of some traffic T. Therefore, we have the following lemma.

Lemma 3.5. *If from every configuration C there is an (R,B)-leaky-bucket traffic T such that $\Pr_\sigma\left[\mathcal{E}_{PPS}(\mathtt{ALG},\sigma,T) \text{ is } (f,s)\text{-concentrating}\right] > 0$, then with probability $1-\delta$, $\mathcal{R}^{adp}_{avg}(\mathtt{ALG}) \geq f \cdot r' - (s+B) - \varepsilon$, where $\varepsilon > 0$ and $\delta > 0$ can be made arbitrarily small.*

Proof. By Lemma 3.4, an adaptive adversary can construct a traffic T' from configuration C, such that $\Pr_\sigma\left[\mathcal{E}_{PPS}(\mathtt{ALG},\sigma,T') \text{ is } (f,s)\text{-concentrating}\right] \geq 1-\delta$.

Let c be the last cell of T'. Lemma 3.3 implies that with probability $1-\delta$, the relative queuing delay of c is at least $f \cdot r' - (s+B)$.

The adaptive adversary continues with traffic T'', which consists of $\lceil |T'| \frac{f \cdot r' - (s+B) - \varepsilon}{\varepsilon} \rceil$ cells from $orig(c)$ to $dest(c)$, one cell at each time-slot. T'' is a proper continuation of traffic T', because both the PPS and the shadow switch obey a per-flow FCFS policy and all cells in T'' share the same input-port and the same output-port.

Hence, Lemma 3.2 implies that $\mathcal{R}^{adp}_{avg}(\mathtt{ALG}) \geq f \cdot r' - (s+B) - \varepsilon$, with probability $1-\delta$. □

3.2. Lower bound for fully distributed algorithms with an adaptive adversary.

A *fully distributed demultiplexing algorithm* [3] demultiplexes a cell, arriving at time-slot t, according to the input-port's local information in time interval $[0,t]$. Since no information is shared between input-ports, we assume that the state $s_i \in \mathbb{S}_i$ of demultiplexor i does not change, unless a cell arrives at input-port i. Note that demultiplexing algorithms that change their state even without receiving a cell are not considered fully distributed, because a common clock-tick is shared among all input-ports. (Such algorithms are covered in section 3.3.)

The relative queuing delay of a PPS with a fully distributed demultiplexing algorithm strongly depends on the number of input-ports that can send a cell, destined for the same output-port, through the same plane. The following definition captures this switch characteristic.

Definition 3.6. *A demultiplexing algorithm is* d-partitioned *if there is a plane k, an output-port j, and a set of input-ports I, such that $|I| \geq d$ and the following property holds: For every input-port $i \in I$ and state $s_i \in \mathbb{S}_i$, if at least n_i cells destined for output-port j arrive at input-port i after it is in state s_i, then with probability $p_i > 0$, i sends at least one cell destined for output-port j through plane k.*

We later show that a static partition of the planes among the demultiplexors may reduce the relative queuing delay. However, since such partitioning is failure-prone, most existing fully distributed algorithms are N-partitioned, meaning that each demultiplexor may use each plane in order to send cells to each output-port. All our results hold for this class of algorithms by substituting $d = N$.

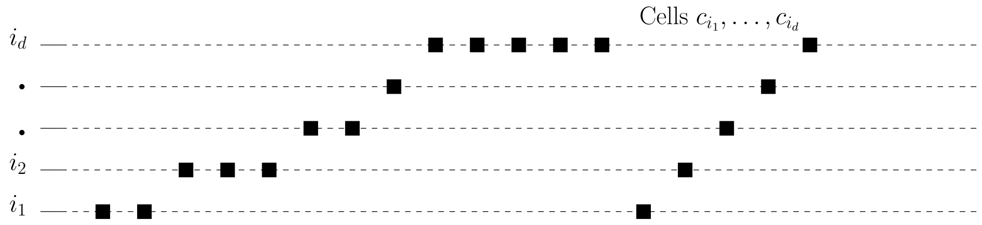


FIG. 3. *Illustration of traffic T in the proof of Theorem* 3.7.

We next prove a lower bound for d-partitioned fully distributed demultiplexing algorithms by showing that it is possible to construct a traffic with no bursts that causes, with nonnegligible probability, the algorithm to concentrate d cells in a single plane during a time interval of d time-slots; the proof follows the lower bound for the deterministic case [3, Theorem 6].

THEOREM 3.7. *Any randomized d-partitioned fully distributed demultiplexing algorithm* ALG *has, with probability $1-\delta$, $\mathcal{R}^{adp}_{max}(\texttt{ALG}) \geq d(r'-1)$ time-slots and $\mathcal{R}^{adp}_{avg}(\texttt{ALG}) \geq d(r'-1)-\varepsilon$ time-slots, where $\varepsilon > 0$ and $\delta > 0$ can be made arbitrarily small.*

Proof. Given ALG, the adversary computes the set $I = \{i_1, \ldots, i_d\}$ of d input-ports, the output-port j, and the plane k, and for each input-port $i \in I$ the values n_i and $p_i > 0$, for which the conditions presented in Definition 3.6 hold.

Fix a configuration C, and for every $i \in I$, let T'_i be a traffic consisting of n_i cells destined for output-port j that arrive one after the other to input-port i. By the definition of n_i, with probability at least p_i, there is at least one cell in T'_i that is sent through plane k. Let c_i be the first such cell; it follows that $\Pr_\sigma$ [in $\mathcal{E}_{PPS}(\texttt{ALG}, \sigma, T'_i)$ cell c_i is sent through plane k] $\geq p_i$. Let T_i be the prefix of T'_i that ends with cell c_i; that is, $T_i = \{c \in T'_i | ta(c) \leq ta(c_i)\}$. Since the probability to send c_i through plane k in execution $\mathcal{E}_{PPS}(\texttt{ALG}, \sigma, T'_i)$ depends only on cells that arrive at the switch before cell c_i, it follows that for prefix T_i, $\Pr_\sigma$ [in $\mathcal{E}_{PPS}(\texttt{ALG}, \sigma, T_i)$ cell c_i is sent through plane k] $\geq p_i$.

Traffic T is defined as follows: $T = (T_{i_1} \setminus \{c_{i_1}\}) \circ \cdots \circ (T_{i_d} \setminus \{c_{i_d}\}) \circ \{c_{i_1}\} \cdots \circ \{c_{i_d}\}$ (see Figure 3). We next show that, with nonnegligible probability, taken over all coin-toss sequences σ, all cells $c_{i_1} \ldots c_{i_d}$ are sent through plane k in the execution of ALG on traffic T.

In traffic T, for each input-port $i \in I$, no cells arrive to input-port i between $T_i \setminus \{c_i\}$ and c_i. Thus, for each input-port $i \in I$ and coin-toss sequence σ, $plane(c_i, T) = plane(c_i, T_{i_1} \circ \cdots \circ T_{i_d}\}$. Since the demultiplexors are independent, the probability, taken over all coin-toss sequences σ that the last d cells are sent through plane k in execution $\mathcal{E}_{PPS}(\texttt{ALG}, \sigma, T)$ is at least $\prod_{a=1}^{d} p_{i_a} > 0$.

This implies that execution $\mathcal{E}_{PPS}(\texttt{ALG}, \sigma, T)$ is (d,d)-concentrating with nonnegligible probability. Since T has no bursts, the claim follows immediately from Lemma 3.5. □

3.3. Lower bound for u-RT algorithms with an adaptive adversary.

Another interesting class includes *u real-time distributed (u-RT)* demultiplexing algorithms [3], which demultiplex a cell arriving at time-slot t, according to the input-port's local information in time interval $[0,t]$, and to the switch's global information in time interval $[0, t-u]$. In this manner, an input-port state transition may depend on other input-ports' state transitions, and on incoming flows to other input-ports, as long as they occurred more than u time-slots earlier. A prominent example of 1-RT demultiplexing algorithms (that is, with $u = 1$) are demultiplexing algorithms that share only a common clock-tick among input-ports. Note that a 1-RT demultiplexing algorithm may change its state even if no cell arrives at its input-port.

Let $\overline{u} = \min\{u, \frac{r'}{2}\}$, that is, the minimum between the lag in gathering global information and half the external rate relative to the rate of the planes. The next theorem, which is based on Lemma 3.5 and some observations from [3], gives a lower bound on the average relative queuing delay of u-RT demultiplexing algorithms.

THEOREM 3.8. *Any randomized u-RT demultiplexing algorithm* $\mathtt{ALG}$ *has, with probability* $1-\delta$, $\mathcal{R}^{adp}_{max}(\mathtt{ALG}) \geq \frac{\overline{u}N}{S}(1-\frac{\overline{u}}{r'})$ *time-slots and* $\mathcal{R}^{adp}_{avg}(\mathtt{ALG}) \geq \frac{\overline{u}N}{S}(1-\frac{\overline{u}}{r'})-\varepsilon$ *time-slots, where* $\varepsilon > 0$ *and* $\delta > 0$ *can be made arbitrarily small.*

Proof. Consider an arbitrary configuration C. Denote by t_0 the time-slot in which the PPS is in configuration C, by x_0 the number of cells that arrived at the PPS until time-slot t_0, and by n_0 the number of cells stored in one of the PPS's buffers at time-slot t_0.

Consider now the empty traffic T_e, in which no cells arrive at the switch at all. We first argue that if T_e is long enough, all the buffers of the switch become empty. Specifically, denote by C_1 the switch configuration at time-slot $t_1 = t_0 + n_0 + \frac{\overline{u}Nx_0}{S} + 1$. If there are still cells stored in one of the buffers at time-slot t_1, then these cells have relative queuing delay of at least $\frac{\overline{u}Nx_0}{S} + 1$ time-slots; therefore, the average relative queuing delay is more than $\frac{\overline{u}N}{S}$ time-slots, and the theorem follows.

Assume now that all the buffers are empty in configuration C_1. Fix an output-port j, and consider the traffic $\overline{T}$ in which cells destined for j arrive simultaneously to all input-ports at each time-slot in the interval $[t_1, t_1 + \overline{u})$. Note that $\overline{T}$ is an $(R, \overline{u}N - \overline{u})$-leaky-bucket traffic, since for any $\tau \geq 1$ and time interval $[t, t+\tau)$, the total number of cells arriving at the switch is bounded by $\tau + (\overline{u}N - \overline{u})$.

Since $\overline{u} \leq \frac{1}{2}\frac{R}{r} < \frac{R}{r}$, the input constraint implies that two cells arriving at the same input-port are not sent through the same plane. Hence, for every coin-toss sequence σ, there is a plane k used by a set I of at least $\frac{\overline{u}}{K}N$ input-ports in the execution $\mathcal{E}_{PPS}(\mathtt{ALG}, \sigma, \overline{T})$; note that since a PPS speedup is at least 1, $\frac{\overline{u}}{K}N < \frac{R}{rK}N \leq N$.

For every input-port $i \in I$, let $c_i \in \overline{T}|_i$ be a cell such that $plane(c_i, \overline{T}|_i) = k$. Consider the traffic $T|_i = \{c | c \in \overline{T}|_i \text{ and } ta(c) \leq ta(c_i)\}$; that is, $T|_i$ consists of the cells in $\overline{T}|_i$ that arrive at the switch before cell c_i.

Consider the parallel composition of traffics $T|_i$, $T|_I = \bigcup_{i \in I} T|_i$ (see Figure 4). Note that both $T|_I$ and $T_e \circ T|_I$ are $(R, \overline{u}^2\frac{N}{K} - \overline{u})$-leaky-bucket traffics.

For every input-port $i \in I$, $ta(c_i) < t_1 + \overline{u} \leq t_1 + u$, which implies that input-port i does not have global information on the switch status after time-slot t_1. Hence,

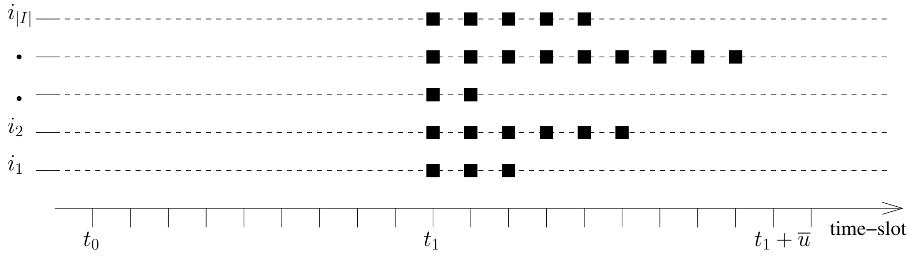


FIG. 4. *Illustration of traffic* $T_e \circ T|_I$ *in the proof of Theorem* 3.8.

the executions $\mathcal{E}_{PPS}(\texttt{ALG}, \sigma, \overline{T})$ and $\mathcal{E}_{PPS}(\texttt{ALG}, \sigma, T|_I)$ are equivalent. Therefore, with probability of at least

$$\prod_{i\in I}\left(\frac{1}{|\texttt{COINSPACE}|}^{|T|_i|}\right) \geq \left(\frac{1}{|\texttt{COINSPACE}|}^{\overline{u}}\right)^{|I|} \geq \left(\frac{1}{|\texttt{COINSPACE}|}^{\overline{u}}\right)^{\frac{\overline{u}N}{K}} > 0,$$

taken over the coin-toss sequences σ, all the input-ports $i \in I$ send their last cell to plane k in $\mathcal{E}_{PPS}(\texttt{ALG}, \sigma, T_e \circ T|_I)$, starting at configuration C. Hence, configuration C satisfies the conditions of Lemma 3.5, and the claim follows. □

3.4. Lower bound for fully distributed algorithms with an oblivious adversary. We now consider oblivious adversaries, *obl*, that choose the entire traffic in advance, knowing only the demultiplexing algorithm ALG [5]. $\mathcal{R}^{obl}_{max}(\texttt{ALG})$ and $\mathcal{R}^{obl}_{avg}(\texttt{ALG})$ denote the maximum and average queuing delay of algorithm ALG against such an adversary. We assume that the PPS and the shadow switch obey a *global FCFS* policy; i.e., cells that share the same output-port should leave the switch in the order of their arrival (with ties broken arbitrarily). Unlike per-flow FCFS policy, global FCFS policy requires cells to leave in order even if they do not share the same origin.

We next extend Theorem 3.7 to hold with an oblivious adversary, under a global FCFS discipline.

THEOREM 3.9. *Any randomized d-partitioned fully distributed demultiplexing algorithm ALG has $\mathcal{R}^{obl}_{max}(\texttt{ALG}) \geq d(r'-1)$ time-slots and $\mathcal{R}^{obl}_{avg}(\texttt{ALG}) \geq d(r'-1) - \varepsilon$ time-slots, with probability $1-\delta$, where $\varepsilon > 0$ and $\delta > 0$ can be made arbitrarily small.*

Proof. Given ALG, the adversary precomputes the set $I = \{i_1, \ldots, i_d\}$ of d input-ports, the output-port j, the plane k, and for each input-port $i \in I$ the values n_i and p_i, for which the conditions of Definition 3.6 hold. Let $\widetilde{p} \triangleq \prod_{a=1}^{d} \frac{p_{i_a}}{n_{i_a}} > 0$.

For any input-port $i \in I$, let x_i be a value chosen uniformly at random from $\{1, \ldots, n_i\}$. Let T_i be a traffic consisting of x_i cells from input-port i to output-port j, and let c_i be the last cell of T_i. Traffic T' is defined as follows: $T' = (T_{i_1} \setminus \{c_{i_1}\}) \circ \cdots \circ (T_{i_d} \setminus \{c_{i_d}\}) \circ \{c_{i_1}\} \cdots \circ \{c_{i_d}\}$. Note that traffic T' is similar to traffic T in the proof of Theorem 3.7 (illustrated in Figure 3).

Using traffic T', the adversary constructs a traffic T (illustrated in Figure 5) whose average relative queuing delay is at least $d(r'-1) - \varepsilon$ time-slots with probability $1-\delta$ (the constants $\delta, \varepsilon > 0$ can be made arbitrarily small). The construction has two steps.

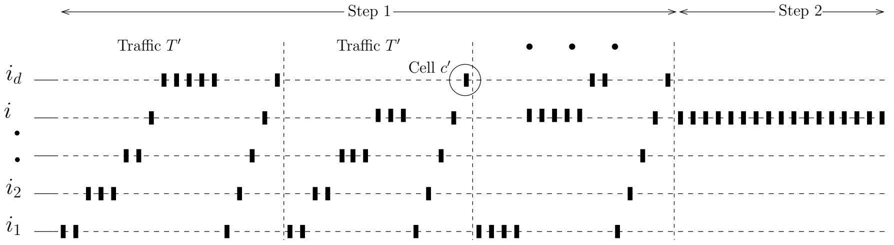


FIG. 5. *Illustration of traffic T in the proof of Theorem* 3.9.

Step 1. Concatenate $\lceil \log_{1-\widetilde{p}} \delta \rceil$ instances of traffic T'. For each instance, choose independently and uniformly at random the values for x_{i_m}, $1 \leq m \leq d$, from $\{1, \dots n_{i_m}\}$. Let ℓ be the total size of these instances.

Step 2. Concatenate a traffic of size $\ell \lceil \frac{d(r'-1)-\varepsilon}{\varepsilon} \rceil$ cells, such that each cell is sent from an arbitrary input-port i to output-port j.

We first prove that with nonnegligible probability, taken over all coin-toss sequences σ, the execution of ALG on any instance of traffic $T' = (T_{i_1} \setminus \{c_{i_1}\}) \circ \cdots \circ (T_{i_d} \setminus \{c_{i_d}\}) \circ \{c_{i_1}\} \cdots \circ \{c_{i_d}\}$ is (d,d)-concentrating, regardless of the initial configuration.

CLAIM 3.10. $\Pr_\sigma[\mathcal{E}_{PPS}(ALG, \sigma, T')$ *sends the last* d *cells through plane* $k] \geq \prod_{a=1}^{d} \frac{p_{i_a}}{n_{i_a}} \triangleq \widetilde{p} > 0$.

Proof of claim. For any input-port i, denote by $\widetilde{T_i}$ the traffic consisting of n_i cells from input-port i to output-port j. By the definition of n_i, with probability p_i, at least one cell in $\widetilde{T_i}$ is sent through plane k. Since x_i is chosen uniformly at random from the values $\{1, \dots, n_i\}$, this cell is the x_ith cell (that is, the cell c_i) with probability at least $\frac{1}{n_i}$. Note that traffic T_i is a prefix of traffic $\widetilde{T_i}$; since the demultiplexor is bufferless, the decision of through which plane to send the cell c_i is based only on cells arriving at the switch prior to c_i, which implies that cell c_i is sent through k with probability of at least $\frac{1}{n_i} \cdot p_i$.

In traffic T', for each input-port $i \in I$, no cells arrive at input-port i between $T_i \setminus \{c_i\}$ and c_i. Thus, for each input-port $i \in I$ and coin-toss sequence σ, $plane(c_i, T') = plane(c_i, T_{i_1} \circ \cdots \circ T_{i_d})$ Since the demultiplexors are independent, the probability, taken over all coin-toss sequences σ, that execution $\mathcal{E}_{PPS}(\mathtt{ALG}, \sigma, T')$ sends the last d cells through plane k is at least $\prod_{a=1}^{d} \frac{p_{i_a}}{n_{i_a}} \triangleq \widetilde{p} > 0$. □

In Step 1, the random choices of the T' instances are independent. Therefore, (d,d)-concentration occurs at least once in Step 1 with probability at least $1 - \delta$. Let c' be last cell of the first instance in which (d,d)-concentration occurs and T_1 be the traffic $\{c \in T \mid ta(c) \leq ta(c')\}$. Since $\mathcal{E}_{PPS}(\mathtt{ALG}, \sigma, T_1)$ is (d,d)-concentrating, Lemma 3.3 implies that $\mathcal{R}(\mathtt{ALG}, \sigma, c', T_1) \geq d(r'-1)$.

Let $T_2 = T \setminus T_1$. We next show that T_2 is a proper continuation of T_1. Intuitively, this is due to the fact that the switches are work-conserving with FCFS policy and during each interval of size τ, exactly τ cells arrive at the switch and are destined for the same output-port j (i.e., there are no stalls between cells in traffic $T = T_1 \circ T_2$).

Formally, consider two cells $c_1, c_2 \in T$ such that $ta(c_2) = ta(c_1) + 1$. The FCFS policy implies that $tl_S(c_2, T) > tl_S(c_1, T)$ and $tl_{PPS}(c_2, T) > tl_{PPS}(c_1, T)$. In addition, by the construction of traffic T, there is no cell $c_3 \in T$ such that $ta(c_1) \leq ta(c_3) \leq ta(c_2)$. Therefore, the FCFS policy and the work-conservation of the shadow switch imply that $tl_S(c_2, T) = tl_S(c_1, T) + 1$. Hence, for every two cells $c_1, c_2 \in T$, if $ta(c_2) = ta(c_1) + 1$, then $c_2 = succ(c_1, T)$; in particular, the first cell of T_2 is the successor of cell c'. Moreover, since the switches follow an FCFS policy, cells of traffic T_2 do not prohibit cells of traffic T_1 from being delivered on time; namely, for any cell $c \in T_1$, $tl_S(c, T_1) = tl_S(c, T_1 \circ T_2)$ and $tl_{PPS}(c, T_1) = tl_{PPS}(c, T_2)$.

Since $\mathcal{R}(\mathtt{ALG}, \sigma, c', T_1) \geq d(r'-1)$ and $|T_2| \geq \lceil |T_1| \frac{d(r'-1)-\varepsilon}{\varepsilon} \rceil$, Lemma 3.2 implies that $\mathcal{R}^{obl}_{avg}(\mathtt{ALG}) \geq d(r'-1) - \varepsilon$ and $\mathcal{R}^{obl}_{max}(\mathtt{ALG}) \geq d(r'-1)$, with probability $1-\delta$. □

The question of whether the lower bound for the u-RT demultiplexing algorithm (described in Theorem 3.8) can be extended to hold with an oblivious adversary is left open. The proof technique described in this section will most likely fail to provide such an extension, since the worst-case traffics that are used in order to prove lower bounds for u-RT algorithms have bursts. Unfortunately, the burstiness accumulates when concatenating bursty traffics, unless there is a gap of a certain number of time-slots in which no cells arrive at the overloaded output-port. Large bursts may justify a high queuing delay of cells and hence result in low *relative* queuing delay. On the other hand, a gap in which no cells arrive at the overloaded output reduces the relative queuing delay of cells that arrive immediately after it. This implies that the adversary should identify the concentration and then choose to continue the traffic without a gap (as in Lemma 3.2).

4. Bounding the relative queuing delay. This section presents a methodology for bounding $\mathcal{R}_{max}(\texttt{ALG}, \sigma, T)$ for an arbitrary traffic T and coin-toss sequence σ. We fix some traffic T and omit the notation $\texttt{ALG}, \sigma$, and T. For simplicity assume the execution begins after time-slot 0, and that at time-slot 0 (i.e., at "the beginning of time"), no cells arrive at the switch, and therefore all the queues are empty. Our analysis depends on the realistic assumption that the PPS obeys the *global* FCFS policy.

Cells are queued in a bufferless PPS either within the planes or within the multiplexors residing at the output-ports. A simple situation in which queuing in a multiplexor happens is when the output-port is flooded, but in this case, cells also suffer from high queuing delay in the shadow switch, and the relative queuing delay is small. A more complicated situation is when a cell arrives at the multiplexor out of order and should wait for previous cells to arrive from their planes. In this case, the relative queuing delay is an indirect result of queuing within the other planes (of some preceding cell)—the relative queuing delay of the waiting cell is at most the relative queuing delay of some preceding cell that was queued only in the planes. This observation is captured in the following lemma.

LEMMA 4.1. *There is a cell c such that $tl_{PPS}(c) = tp(c)$ and $\mathcal{R}(c) = \mathcal{R}_{max}$.*

Proof. Let c be the first cell to leave the PPS such that $\mathcal{R}(c) = \mathcal{R}_{max}$. Assume that $tl_{PPS}(c) > tp(c)$; since the multiplexor buffer is work-conserving, in time-slot $tl_{PPS}(c) - 1$ another cell c' leaves the PPS from output-port $dest(c)$. Hence $tl_{PPS}(c') = tl_{PPS}(c) - 1$, and therefore $\mathcal{R}(c') = tl_{PPS}(c') - tl_S(c') = tl_{PPS}(c) - 1 - tl_S(c')$. Since c' leaves the PPS before c and the shadow switch is FCFS, $tl_S(c') \leq tl_S(c) - 1$. Hence the relative queuing delay of c' is $\mathcal{R}(c') \geq tl_{PPS}(c) - tl_S(c) = \mathcal{R}(c) = \mathcal{R}_{max}$, contradicting the minimality of c. □

Consider a single cell c, and focus on the queuing within $plane(c)$, caused by the lower rate on the link from $plane(c)$ to $dest(c)$. Since both the PPS and the shadow switch are FCFS, cells arriving at the switch after cell c cannot prohibit c from being transmitted on time. We present an upper bound that depends only on the disproportion of the number of cells sent through $plane(c)$ to $dest(c)$. Relating this quantity and the queue lengths at time-slot $ta(c)$ is not immediate, since it is possible that the shadow switch is busy when the plane is idle, and vice versa.

Let $A_j(t_1, t_2)$ be the number of cells destined for output-port j that arrive at the switch during time interval $[t_1, t_2]$, and let $A_j^k(t_1, t_2)$ be the number of these cells that are sent through plane k. The following definition captures the imbalance between planes.

DEFINITION 4.2. *For a plane k, output-port j and time-slots $0 \leq t_1 \leq t_2$ the following are defined.*

1. *The* imbalance of time interval $[t_1, t_2]$ *is* $\Delta_j^k(t_1, t_2) = A_j^k(t_1, t_2) - \frac{1}{r'} A_j(t_1, t_2)$.
2. *The* imbalance by time-slot t_2 *is* $\Delta_j^k(t_2) = \max_{t_1 \leq t_2}\{\Delta_j^k(t_1, t_2)\}$.
3. *The* maximum imbalance *is* $\Delta_j^k = \max_{t_2}\{\Delta_j^k(t_2)\}$.

Clearly, $\Delta_j^k \geq \Delta_j^k(t_2) \geq \Delta_j^k(t_1, t_2)$ for every output-port j, plane k, and time-slots $t_1 > t_2$. In addition, the imbalance is superadditive.

PROPERTY 4.3. *For every output-port j, plane k, and time-slots $t_1 > t_2$,*

$$\Delta_j^k(t_2) \geq \Delta_j^k(t_1 - 1) + \Delta_j^k(t_1, t_2).$$

Proof. By Definition 4.2, there is a time-slot $t_1' \leq t_1$ such that $\Delta_j^k(t_1 - 1) = \Delta_j^k(t_1', t_1 - 1) = A_j^k(t_1', t_1 - 1) - \frac{1}{r'} A_j(t_1', t_1 - 1)$. Since $\Delta_j^k(t_1, t_2) = A_j^k(t_1, t_2) - \frac{1}{r'} A_j(t_1, t_2)$, we have

$$\begin{aligned}\Delta_j^k(t_1, t_2) + \Delta_j^k(t_1 - 1) &= A_j^k(t_1', t_1 - 1) + A_j^k(t_1, t_2) - \tfrac{1}{r'}(A_j(t_1', t_1 - 1) + A_j(t_1, t_2)) \\ &= \Delta_j^k(t_1', t_2) \ \leq \ \Delta_j^k(t_2). \qquad \square\end{aligned}$$

Let $Q_j(t)$ be the size of the jth buffer in the shadow switch after time-slot t; similarly, $Q_j^k(t)$ is the size of jth buffer of plane k of the PPS after time-slot t. Let $L_j^k(t_1, t_2)$ be the number of cells destined for output-port j that leave plane k during time interval $[t_1, t_2]$. Note that $Q_j^k(t) = A_j^k(0, t) - L_j^k(0, t)$.

Time-slot t_1 is the *beginning of a (k, j) busy period* for time-slot $t_2 \geq t_1$ if it is the last time-slot before t_2, such that $Q_j^k(t_1 - 1) = 0$. Note that this expression is well defined because in time-slot 0 all the queues are empty. Since $Q_j^k(t_1) > Q_j^k(t_1 - 1)$, a cell c arrives at the switch at time-slot t_1, and therefore exactly one cell destined for j leaves plane k in time interval $(t_1 + 1 - r', t_1 + 1]$. This is either cell c itself or another cell that prohibits c from using the link and therefore is sent at most r' time-slots before time-slot $t_1 + 1$. Since the queue is never empty until time-slot t_2, one cell is sent to j exactly every r' time-slots after the first cell. This implies that the number of cells sent from k, $L_j^k(t_1, t_2) \geq \lfloor \frac{(t_2 - t_1) + 1}{r'} \rfloor$.

Remark 4.1. Khotimsky and Krishnan [25] defined busy periods only with respect to an output-port j. This points to a flaw in their proof, which ignores situations when the optimal shadow switch is busy sending cells to output-port j, while a specific plane in the PPS is idle part of the time [24]. These situations are the main source of difficulty in our proof.

The following lemma, which is illustrated in Figure 6, bounds how badly a plane can perform relative to the shadow switch, by comparing their busy periods.

LEMMA 4.4. *If $Q_j(t - 1) = 0$, then for every plane k and for every $\delta \in \{0, \ldots, \Delta_j^k(t - 1)r'\}$,*

$$L_j^k\left(0, (t - 1) + \delta\right) \geq A_j^k(0, t - 1) - \left\lceil \frac{\Delta_j^k(t - 1)r' - \delta}{r'} \right\rceil.$$

Proof. If there is a time-slot $t_1 \in [t - 1, t - 1 + \delta]$ such that $Q_j^k(t_1) = 0$, then by time-slot t_1 no cells destined for j are waiting in plane k. That is, $L_j^k(0, (t - 1) + \delta) \geq L_j^k(0, t_1) = A_j^k(0, t_1) \geq A_j^k(0, t - 1)$, and the lemma follows.

Otherwise, let t_2 be the beginning of a (k, j) busy period for time-slot $(t - 1) + \delta$. During time interval $[t_2, (t - 1) + \delta]$ plane k sends a cell to output j every r' time-slots; therefore,

$$L_j^k(t_2, (t - 1) + \delta) \geq \left\lfloor \frac{t + \delta - t_2}{r'} \right\rfloor. \tag{1}$$

On the other hand, $Q_j(t - 1) = 0$ implies that for every time-slot $t_3 \leq t - 1$, $A_j(t_3, t - 1) \leq t - t_3$; otherwise, the jth buffer of the shadow switch is not empty after time-slot $t - 1$. In particular,

$$A_j(t_2, t - 1) \leq t - t_2. \tag{2}$$

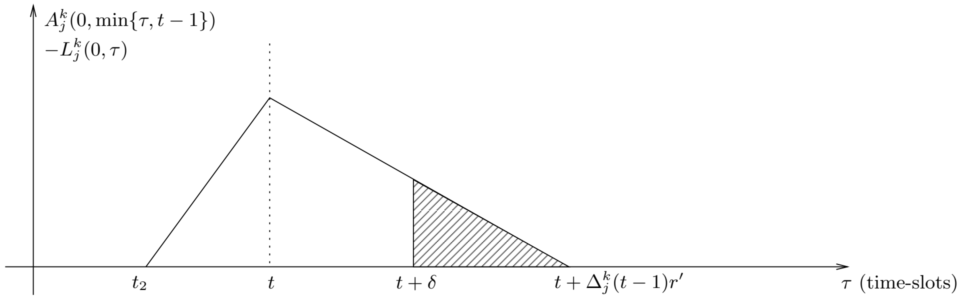


FIG. 6. *Number of cells arriving until time-slot $t-1$ and still queued in plane k by time-slot τ.*

Using these inequalities, we bound $L_j^k(0,(t-1)+\delta)$:

$$\begin{aligned}
L_j^k(0,(t-1)+\delta) &= L_j^k(0,t_2-1) + L_j^k(t_2,(t-1)+\delta) \\
&\geq L_j^k(0,t_2-1) + \left\lfloor \tfrac{t+\delta-t_2}{r'} \right\rfloor && \text{by (1)} \\
&\geq A_j^k(0,t_2-1) + \left\lfloor \tfrac{t+\delta-t_2}{r'} \right\rfloor && \text{since } Q_j^k(t_2-1)=0 \\
&\geq A_j^k(0,t_2-1) + \left\lfloor \tfrac{\delta}{r'} + \tfrac{A_j(t_2,t-1)}{r'} \right\rfloor && \text{by (2)} \\
&= A_j^k(0,t_2-1) + A_j^k(t_2,t-1) + \left\lfloor \tfrac{\delta}{r'} - \Delta_j^k(t_2,t-1) \right\rfloor \\
& && \text{by Definition 4.2} \\
&\geq A_j^k(0,t-1) - \left\lceil \frac{r'\Delta_j^k(t-1)+\delta}{r'} \right\rceil . \qquad \square
\end{aligned}$$

By substituting $\delta = 0$ in Lemma 4.4, we get the following corollary, demonstrating the relation between the imbalance and the queue size in the beginning of a busy period.

COROLLARY 4.5. *If $Q_j(t-1) = 0$, then for every plane k,*

$$Q_j^k(t-1) \leq \max\left\{0, \left\lceil \Delta_j^k(t-1) \right\rceil \right\}.$$

We complete the proof by bounding the lag between the time a cell leaves the plane it is sent through and the time it should leave the shadow switch.

THEOREM 4.6. *The maximum relative queuing delay of cells destined for output-port j and sent through plane k is bounded by $\max\{0, r'(\Delta_j^k + 1) + B_j\}$, where B_j is the maximum number of cells destined for output-port j that arrive at the switch in the same time-slot.*

Proof. By Lemma 4.1, it suffices to bound $tp(c) - tl_S(c)$ for every cell c. Since $tp(c) - tl_S(c) = (tp(c) - ta(c)) - (tl_S(c) - ta(c))$, it suffices to bound only the difference between the time a cell spends in the plane, $tp(c) - ta(c)$, and the time it spends in the shadow switch, $tl_S(c) - ta(c)$. Since both switches operate under the FCFS policy, these values depend solely on the corresponding queues' lengths when cell c arrives.

Let t_1 be the earliest time-slot, such that the buffer of output-port j in the shadow switch is never empty during time interval $[t_1, ta(c)]$; if no such time-slot exists, let $t_1 = ta(c)$.

First, we bound $tl_S(c) - ta(c)$ from below. The buffer in the shadow switch is empty at time-slot t_1–1, and then the switch is continuously busy during time interval $[t_1, ta(c)-1]$, transmitting exactly one cell at each time-slot to output-port j. This

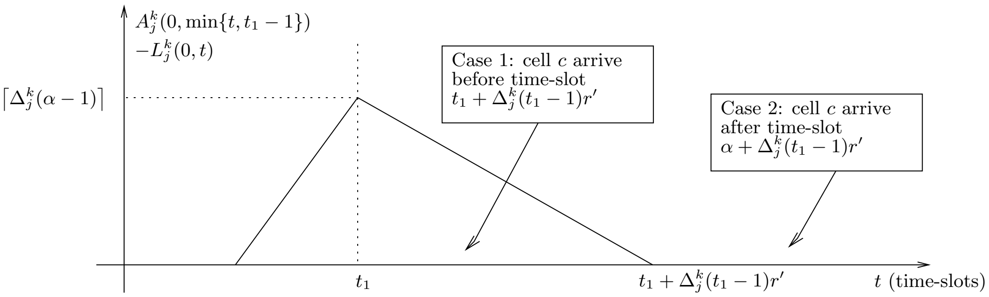

$A_j^k(0, \min\{t, t_1-1\})$
$-L_j^k(0,t)$
$\lceil \Delta_j^k(\alpha-1) \rceil$
Case 1: cell c arrive before time-slot $t_1 + \Delta_j^k(t_1-1)r'$
Case 2: cell c arrive after time-slot $\alpha + \Delta_j^k(t_1-1)r'$
t_1
$t_1 + \Delta_j^k(t_1-1)r'$
t (time-slots)

FIG. 7. *Illustration for the different cases in the proof of Theorem* 4.6.

implies that $Q_j(ta(c)-1) = A_j(t_1, ta(c)-1) - (ta(c) - t_1)$. All the cells in the queue should leave the switch after time-slot $ta(c)$ and before $tl_S(c)$; therefore,

$$tl_S(c) - ta(c) > A_j(t_1, ta(c)-1) - (ta(c) - t_1).$$

Since $A_j(ta(c), ta(c)) \leq B_j$, and $tl_S(c) - ta(c)$ is an integer, it follows that

$$tl_S(c) - ta(c) \geq A_j(t_1, ta(c)) - B_j + t_1 - ta(c) + 1. \tag{3}$$

Recall that by Corollary 4.5, $Q_j^k(t_1-1) \leq \max\{0, \lceil \Delta_j^k(t_1-1) \rceil\}$. There are two cases to consider, depending on whether all the cells that were queued in plane k at time-slot t_1 left the plane before the arrival of cell c (see Figure 7).

Case 1. $ta(c) \leq t_1 + \Delta_j^k(t_1-1)r'$. Since plane k is FCFS and work-conserving, it transfers every cell in its queue in exactly r' time-slots, except cell c, which is considered as transferred in the first time-slot of its transmission:

$$\begin{aligned}
tp(c) - ta(c) &\leq r' Q_j^k(ta(c)) + 1 \\
&= r'(A_j^k(0, ta(c)) - L_j^k(0, ta(c))) + 1 \qquad \text{by the definition of } Q_j^k(ta(c)) \\
&\leq r' \left(A_j^k(0, ta(c)) - A_j^k(0, t_1 - 1) + \left\lceil \frac{r'\Delta_j^k(t_1-1) + t_1 - ta(c)}{r'} \right\rceil \right) + 1 \\
&\qquad \text{by Lemma 4.4, since } ta(c) \in [t_1, t_1 + \Delta_j^k(t_1-1)r'] \\
&\qquad \text{and } L_j^k(0, ta(c)) \geq L_j^k(0, ta(c)-1) \\
&\leq r' \left(A_j^k(0, ta(c)) - A_j^k(0, t_1 - 1) + \frac{r'\Delta_j^k(t_1-1) + t_1 - ta(c)}{r'} + 1 \right) + 1 \\
&\leq r' A_j^k(t_1, ta(c)) + r'\Delta_j^k(t_1-1) + t_1 - ta(c) + r' + 1 \\
&= A_j(t_1, ta(c)) + r'\Delta_j^k(t_1, ta(c)) + r'\Delta_j^k(t_1-1) - ta(c) + t_1 + r' + 1 \\
&\qquad \text{by Definition 4.2} \\
&\leq A_j(t_1, ta(c)) + r'(\Delta_j^k(ta(c)) + 1) - ta(c) + t_1 + 1 \qquad \text{by Property 4.3.}
\end{aligned}$$

Together with (3), this implies that $tp(c) - tl_S(c) \leq r'(\Delta_j^k(ta(c)) + 1) + B_j$.

Case 2. $ta(c) > t_1 + \Delta_j^k(t_1-1)r'$. If $Q_j^k(ta(c)) = 0$, then cell c is immediately delivered to the output-port, i.e., $tp(c) = ta(c) + 1 \leq tl_S(c)$, and the claim holds since $tp(c) - tl_S(c) \leq 0$.

If $Q_j^k(ta(c)) > 0$, let t_2 be the beginning of a (k, j) busy period for $ta(c)$. Note that by the choice of t_2, $L_j^k(t_2, ta(c)) \geq \lfloor \frac{ta(c)-t_2+1}{r'} \rfloor$. Hence, we have

$$\begin{aligned}
tp(c) - ta(c) &\leq r' Q_j^k(ta(c)) + 1 \\
&= r'\left(A_j^k(t_2, ta(c)) - L_j^k(t_2, ta(c))\right) + 1 \qquad\qquad \text{since } Q_j^k(t_2 - 1) = 0 \\
&\leq r'\left(A_j^k(t_2, ta(c)) - \left\lceil \frac{ta(c) - (t_2 - 1)}{r'} \right\rceil\right) + 1 \\
&\qquad\qquad\qquad\qquad \text{since plane } k \text{ is continuously busy} \\
&\leq A_j(t_2, ta(c)) + r'\Delta_j^k(t_2, ta(c)) - r'\left\lceil \frac{ta(c) - (t_2 - 1)}{r'} \right\rceil + 1 \\
&\leq A_j(t_1, ta(c)) + r'\left(\Delta_j^k(t_2, ta(c)) + 1\right) + t_1 - ta(c) + (t_2 - t_1) \\
&\qquad - A_j(t_1, t_2 - 1) + 1.
\end{aligned}$$

By the choice of t_1, the output-buffer of the shadow switch is empty at time-slot $t_1 - 1$ and not empty during time interval $[t_1, t_2 - 1]$. This implies that $(t_2 - t_1) \leq A_j(t_1, t_2 - 1)$, and therefore

$$tp(c) - ta(c) \leq A_j(t_1, ta(c)) + r'(\Delta_j^k(ta(c)) + 1) + t_1 - ta(c) + 1.$$

Together with (3), this implies that $tp(c) - tl_S(c) \leq r'(\Delta_j^k(ta(c)) + 1) + B_j$. □

5. Demultiplexing algorithms with optimal relative queuing delay.

This section presents several demultiplexing algorithms and uses the methodology described in section 4 in order to bound their relative queuing delay.

First, we revisit the *fractional traffic dispatch (FTD) algorithm* [20] and show that its relative queuing delay is $(N+1)r'$ time-slots. For a PPS with speedup $S > 2$, we introduce a variant of the FTD algorithm that is $2N/S$-partitioned; its relative queuing delay is at most $(2N/S + 1)\, r' + N(1 - 2/S)$ time-slots, matching the lower bound for fully distributed demultiplexing algorithms (Theorem 3.7).

Then, we present novel 1-RT and u-RT demultiplexing algorithms with relative queuing delays of $3N + r'$ time-slots (sections 5.2 and 5.3). Both algorithms have optimal relative queuing delays when the speedup of the PPS is constant.

5.1. Optimal fully distributed demultiplexing algorithm.

The *fractional traffic dispatch (FTD) algorithm* [20] is the best-known example of a fully distributed demultiplexing algorithm. In the FTD algorithm, there is a *window* of size r' time-slots that slides over the sequence of cells in each flow (i, j). The algorithm maintains a window constraint that ensures that two cells in the same window are not sent through the same plane. An equivalent variation of the algorithm statically divides each flow to blocks of size r' [20, 25].

The demultiplexing algorithm chooses the plane through which a cell is sent arbitrarily from the set of planes that do not violate the window constraint and the input constraint described at section 2. A speedup of $S \geq 2$ suffices for the algorithm to work correctly [20].

A simple application of Theorem 4.6 and the fact that $B_j \leq N$ shows the following.

THEOREM 5.1. $\mathcal{R}_{avg}(FTD) \leq \mathcal{R}_{max}(FTD) \leq (N+1)r'$.

Proof. Let $A_{i\to j}(t_1, t_2)$ be the number of cells in flow (i, j) that arrive at the switch during time interval $[t_1, t_2]$, and let $A_{i\to j}^k(t_1, t_2)$ be the number of these cells that are sent through plane k.

$$\begin{aligned}
\Delta_j^k(t_1,t_2) &= A_j^k(t_1,t_2) - \frac{A_j(t_1,t_2)}{r'} && \text{by Definition 4.2}\\
&= \sum_{i=1}^{N} A_{i\to j}^k(t_1,t_2) - \frac{A_j(t_1,t_2)}{r'}\\
&\le \sum_{i=1}^{N} \left\lceil \frac{A_{i\to j}(t_1,t_2)}{r'} \right\rceil - \frac{A_j(t_1,t_2)}{r'} && \text{due to the window constraint}\\
&\le \sum_{i=1}^{N} \left(\frac{A_{i\to j}(t_1,t_2)}{r'} + \frac{r'-1}{r'} \right) - \frac{A_j(t_1,t_2)}{r'} && \text{since } A_{i\to j}, r' \text{ are integers}\\
&= N\frac{r'-1}{r'}.
\end{aligned}$$

By Theorem 4.6, $\mathcal{R}_{max}(FTD) \le (N+1)r'$, since $B_j \le N$. □

For a PPS with speedup $S > 2$, a $\frac{2N}{S}$-partitioned variant of the FTD algorithm yields a better relative queuing delay, matching the lower bounds described in Theorems 3.7 and 3.9. In this algorithm, denoted PART-FTD, demultiplexor i uses only planes from the set $\{2r'\lfloor\frac{i}{2N/S}\rfloor, \ldots, 2r'(\lfloor\frac{i}{2N/S}\rfloor + 1) - 1\}$. This implies that each demultiplexor uses exactly $2r'$ planes, as required for the correctness of the FTD algorithm, but each plane is used only by at most $\frac{2N}{S}$ demultiplexors.

THEOREM 5.2. *$\mathcal{R}_{avg}$(PART-FTD) $\le$ $\mathcal{R}_{max}$(PART-FTD) $\le$ $\left(\frac{2N}{S}+1\right)r' + N\left(1-\frac{2}{S}\right)$.*

Proof. We use the same notation as in the proof of Theorem 5.1. The only difference is that

$$\Delta_j^k(t_1,t_2) = A_j^k(t_1,t_2) - \frac{A_j(t_1,t_2)}{r'} \le \sum_{i=1}^{N} \left\lceil \frac{A_{i\to j}(t_1,t_2)}{r'} \right\rceil - \frac{A_j(t_1,t_2)}{r'} \le \frac{2N}{S}\frac{r'-1}{r'}$$

since at most $\frac{2N}{S}$ demultiplexors can send cells through plane k. Therefore, by Theorem 4.6, $\mathcal{R}_{max}$(PART-FTD) $\le \left(\frac{2N}{S}+1\right)r' + N\left(1-\frac{2}{S}\right)$. □

5.2. Optimal 1-RT demultiplexing algorithm. We describe a 1-RT demultiplexing algorithm that matches the lower bound presented in Theorem 3.8. Informally, Algorithm 1 divides the set of planes into two equal-size sets, V_0 and V_1, and its operations with respect to cells destined for a specific output-port into two phases. In each phase, the algorithm sends cells destined for a specific output-port through different set of planes (i.e., V_0 or V_1). After every time-slot, each input-port collects global information about the switch and uses it to calculate the imbalance for each plane k and each output-port j. In the next time-slot, each input-port sends a cell to output-port j only through planes with low (or zero) imbalance. Intuitively, a phase i ends when there are no balanced planes in V_i to use. In the next phase, the demultiplexors use the planes of the set V_{1-i}.

To avoid situations in which all the input-ports send cells through the same plane, we divide the input-ports into $\frac{N}{r'}$ sets of size r' and ensure that under no circumstances can two input-ports in the same set send a cell destined for the same output-port through the same plane. This is done by calculating the actions of other input-ports in the same set as if they indeed get a cell destined for the same output-port.

Algorithm 1 1-RT Algorithm

```
Constants:
    $V_0 = \{1, \ldots, \frac{K}{2}\}$; $V_1 = \{\frac{K}{2} + 1, \ldots, K\}$
Shared:
    $F[N]$: $N$ sets of planes, initially all $V_0$                ▷ cells for $j$ can be sent only through $F[j]$
    $R[N][r']$: matrix of values in $\{1, \ldots, K, \perp\}$, initially all $\perp$      ▷ holding input-constraints
    $t$: value in $\{0, \ldots, r' - 1\}$, initially 0                    ▷ cyclic pointer to matrix R
    $Q[N], L[N]$: $N$ sets of planes, initially all $\emptyset$
    $M[N]$: $N$ sets of planes, initially all $\{1, \ldots, K\}$
    $phase[N]$: vector of values in $\{0, 1\}$, initially all 0

void procedure ADVANCE-CLOCK( )                    ▷ invoked at the beginning of each time-slot
    For every $j \in \{1, \ldots, N\}$: CALCULATE($j$)
    For every $j \in \{1, \ldots, N\}$: $F[j] \leftarrow$ UPDATE($j$)
    Update the matrix $R[N][r']$ according to global information
    $t \leftarrow (t + 1) \bmod r'$
end procedure

int procedure DISPATCH(cell $c$) at demultiplexor $i$
    $j \leftarrow dest(c)$
    $p \leftarrow \lfloor \frac{i}{r'} \rfloor$
    set $B \leftarrow \emptyset$
    for $x \leftarrow r'p$ to $i$ do
        $E \leftarrow \{k \in \{1, \ldots, K\} \mid \exists a \in \{0, \ldots, r' - 1\}, R[x][a] = k\}$
        $k \leftarrow \min(F[j] \setminus (B \cup E))$
        $B \leftarrow B \cup \{k\}$
    end for
    $R[i][t] \leftarrow k$                  ▷ can be read by other input-ports only in the next time-slot
    return $k$
end procedure

set procedure UPDATE(int $j$)
    set $S \leftarrow F[j]$
    $Q[j] \leftarrow Q[j] \setminus M[j]$
    if $Q[j] = \emptyset$ then                                             ▷ change phase
        $Q[j] \leftarrow \{1, \ldots, K\} \setminus M[j]$
        $phase[j] \leftarrow (1 - phase[j])$
        $S \leftarrow V_{phase[j]}$
    else
        $S \leftarrow S \setminus (L[j])$
    end if
    return $S$
end procedure

void procedure CALCULATE(int $j$)
    set $A \leftarrow \{k \mid \Delta_j^k(t) > \frac{N}{r'}\}$                         ▷ using global information
    $M[j] \leftarrow \{k \mid \Delta_j^k(t) \leq 0\}$                             ▷ using global information
    $L[j] \leftarrow (L[j] \cup A) \setminus M[j]$
end procedure
```

With respect to each output-port j, planes are divided into three levels according to their imbalance (see Definition 4.2): *balanced* planes with imbalance $\Delta_j^k(t) \leq 0$, *extremely imbalanced* planes with imbalance $\Delta_j^k(t) \geq \frac{N}{r'}$, and *slightly imbalanced* planes whose imbalance satisfies $0 < \Delta_j^k(t) < \frac{N}{r'}$. At the beginning of each time-slot, a set of *eligible* planes, denoted by $F[j]$, is calculated for every destination j: A plane is eligible for output-port j if it is balanced with respect to output-port j or if it was never extremely imbalanced with respect to output-port j since the last phase change. Phase i is changed to phase $1 - i$ when all planes $k \in V_{1-i}$ become balanced, as maintained by the set $Q[j]$.

Table 5.1

Illustration of Example 5.1. *The plane number through which each demultiplexor would have sent a cell destined for output-port* 0, *if such a cell arrived at the switch. Actual arrivals are marked in framed boxes. No cells arrive at different output-ports in this time interval.*

	Time-slot								
	0	1	2	3	4	5	6	7	8
Demultiplexor 0		$\boxed{1}$	2	1	$\boxed{2}$	3	$\boxed{9}$	10	1
Demultiplexor 1		2	$\boxed{1}$	2	$\boxed{3}$	2	$\boxed{10}$	9	2
Demultiplexor 2		$\boxed{1}$	2	1	2	2	$\boxed{9}$	10	1
Demultiplexor 3		2	$\boxed{1}$	2	3	3	$\boxed{10}$	9	2
Demultiplexor 4		$\boxed{1}$	2	$\boxed{1}$	2	$\boxed{2}$	9	9	1
Demultiplexor 5		2	$\boxed{1}$	$\boxed{2}$	3	3	10	10	2
Demultiplexor 6		1	$\boxed{1}$	2	2	2	9	$\boxed{9}$	1
Demultiplexor 7		2	2	$\boxed{1}$	3	3	10	10	2
$\Delta_0^1(t)$		1.5	3.5	4	3	2.5	0.5	0	0
$\Delta_0^9(t)$	6.5	5	3	1.5	0.5	0	0	0.5	0.5

We demonstrate the operations of the algorithm with the following example for an 8×8 PPS with speedup $S = 8$ and $r' = \frac{R}{r} = 2$.

Example 5.1. Suppose that at time-slot $t = 0$, *phase*[0] changed from 1 to 0, $\Delta_0^9(0) = 6.5$, and all other planes in V_1 have imbalance at most 6.5. In addition, we assume that planes 1 and 2 did not receive any cells before time-slot 0.

The demultiplexors are divided into 4 sets: $\{0, 1\}, \{2, 3\}, \{4, 5\}, \{6, 7\}$. Upon receiving a cell, each demultiplexor calculates the behavior of all demultiplexors in its set that have smaller index and ensures that it will not send the cell through the same plane as them. Table 5.1 shows the plane number through which each demultiplexor would have sent a cell destined for output-port 0 if such a cell arrives at the switch.

The actual arrivals are marked in framed boxes and are taken into account in the following time-slots.

At time-slot 1, demultiplexor 0 will send a cell through the first plane in V_0 (that is, plane 1). On the other hand, demultiplexor 1 must avoid sending its cell through plane 1, and therefore it will use plane 2. Similarly, demultiplexors $2, 4$, and 6 will use plane 1, and demultiplexors $3, 5$, and 7 will use plane 2.

At time-slot 2, demultiplexor 0 cannot use plane 1 due to the input-constraint. Therefore, it will use plane 2 and demultiplexor 1 will use plane 1. Plane 1 becomes extremely imbalanced after time-slot 2, and therefore it is not eligible to receive cells for output-port 0 in the following time-slots. Although plane 1 becomes slightly imbalanced after time-slot 3, Algorithm 1 dictates that it is still not eligible for output-port 0, since the phase has not changed yet.

The phase changes after time-slot 5 because, for every plane $k \in V_1$, $\Delta_0^k(5) \leq 0$. This implies that planes from the set V_1 are used for sending cells destined for output-port 0 in the following time-slots. At this time, $Q[0] = \{1, 2\}$ since $\Delta_0^1(5) = 2.5$ and $\Delta_0^2(5) = 0.5$. The phase changes again after time-slot 7, since both $\Delta_0^1(7)$ and $\Delta_0^2(7)$ are not positive.

To prove the correctness of Algorithm 1, we start with two lemmas.

The first lemma shows that the imbalance between each plane and each output-port is bounded under this algorithm.

Lemma 5.3. *In Algorithm* 1, *for every plane $k \in V_0 \cup V_1$ and output-port j, $\Delta_j^k < \frac{2N}{r'}$.*

Proof. Clearly, if $\Delta_j^k(t_3) > \frac{N}{r'}$, then $k \in L[j]$ in the beginning of time-slot t_3+1 (procedure CALCULATE, line 4). Therefore, $k \notin F[j]$ in the beginning of time-slot t_3+1 (procedure UPDATE, line 9), and cells are not sent through plane k until a time-slot $t_3{}' > t_3+1$ in which $\Delta_j^k(t_3{}'-1) \leq 0$. This observation also holds if the phase changes in the beginning of time-slot t_3+1, since the condition $Q[j] = \emptyset$ in line 4 of procedure UPDATE implies that $V_{phase} \subseteq M[j]$ in line 7.

For every two input-ports i_1 and i_2, if $\lfloor \frac{i_1}{r'} \rfloor = \lfloor \frac{i_2}{r'} \rfloor$, then i_1 and i_2 do not send cells destined for the same output-port through the same plane in the same time-slot (procedure DISPATCH). This implies that the maximum number of cells destined for the same output-port and sent through the same plane in a single time-slot is $\frac{N}{r'}$.

By Definition 4.2, if a plane does not receive cells destined for output-port j in time-slot t_1, then $\Delta_j^k(t_1) \leq \Delta_j^k(t_1-1)$. This implies that there is a time-slot t_1, in which plane k receives cells destined for j, and $\Delta_j^k(t_1) = \Delta_j^k$. In the worst case, $\Delta_j^k(t_1-1) = \frac{N}{r'}$ and k receives $\frac{N}{r'}$ cells destined for j.

Assume, toward a contradiction, that $\Delta_j^k(t_1) \geq \frac{2N}{r'}$. Then there is a time-slot t_2 such that $\Delta_j^k(t_2,t_1) \geq \frac{2N}{r'}$. Note that $\Delta_j^k(t_1,t_1) < \frac{N}{r'}$, since $A_j^k(t_1,t_1) \leq \frac{N}{r'}$ and $A_j(t_1,t_1) \geq A_j^k(t_1,t_1)$. This implies that $t_2 < t_1$, and therefore by Definition 4.2

$$\begin{aligned}\Delta_j^k(t_2,t_1) &= A_j^k(t_2,t_1) - \frac{1}{r'}A_j(t_2,t_1)\\ &= A_j^k(t_2,t_1-1) + A_j^k(t_1,t_1) - \frac{1}{r'}A_j(t_2,t_1-1) - \frac{1}{r'}A_j(t_1,t_1)\\ &= \Delta_j^k(t_2,t_1-1) + \Delta_j^k(t_1,t_1) \ < \ \frac{2N}{r'}.\end{aligned}$$

This contradicts the choice of t_2, and the claim follows. □

The second property is a simple conclusion from Lemma 5.3.

LEMMA 5.4. *If* $2N$ *cells destined for output* j *arrive at a PPS operating under Algorithm* 1 *during time interval* $[t_1,t_2]$ *and none of them is sent through plane* k*, then* $\Delta_j^k(t_2) \leq 0$.

Proof. By Definition 4.2, there is a time-slot t_3 such that $\Delta_j^k(t_2) = \Delta_j^k(t_3,t_2)$.

If $t_3 \geq t_1$, then $\Delta_j^k(t_3,t_2) \leq 0$, since $A_j^k(t_3,t_2) = 0$. Otherwise,

$$\begin{aligned}\Delta_j^k(t_3,t_2) &= \Delta_j^k(t_3,t_1-1) + \Delta_j^k(t_1,t_2)\\ &= \Delta_j^k(t_3,t_1-1) + A_j^k(t_1,t_2) - \frac{1}{r'}A_j(t_1,t_2).\end{aligned}$$

By Lemma 5.3 and Definition 4.2, $\Delta_j^k(t_3,t_1-1) \leq \Delta_j^k \leq \frac{2N}{r'}$. Since $A_j^k(t_1,t_2) = 0$ and $A_j(t_1,t_2) \geq 2N$, it follows that $\Delta_j^k(t_3,t_2) \leq 0$ also in this case. □

The final theorem shows that a speedup of 8 suffices for this demultiplexing algorithm to achieve optimal relative queuing delay. Note that such a high speedup is considered impractical for real switches; nevertheless, Algorithm 1 demonstrates that the lower bound presented in Theorem 3.8 is tight for $u = 1$.

THEOREM 5.5. *Algorithm* 1 *works correctly with speedup* $S = 8$ *and maximum relative queuing delay of* $3N + r'$ *time-slots.*

Proof. It suffices to show that every time line 7 of procedure DISPATCH is executed, $F[j] \setminus (B \cup E) \neq \emptyset$, and a plane can be chosen. Clearly, at each step, $|B| \leq r'$ and $|E| < r'$; therefore, the claim follows if $|F[j]| > 2r'$. Since $F[j]$ is changed only by procedure UPDATE(j), it suffices to show that $|F[j]| > 2r'$ after any execution of UPDATE(j).

Assume, without loss of generality, that *phase* $= 0$ after an execution of procedure UPDATE(j) at time-slot t_1. Assume, by way of contradiction, that $|F[j]| \leq 2r'$ at time-slot t_1. Clearly, from line 7 and the fact that $|V_0| = |V_1| = \frac{K}{2} = \frac{Sr'}{2} = 4r' > 2r'$, it follows that *phase* $= 0$ after time-slot $t_1 - 1$. This implies that $|V_0 \cap L[j]| \geq 2r'$.

Denote by t_2 the last time-slot in which *phase* was changed from 1 to 0 ($t_2 = 0$ if no such time-slot exists). At time-slot t_2, when executing line 4, $Q[j]$ is empty, and therefore all planes $k \in V_0$ are at $M[j]$ at time-slot t_2. This implies that for every $k \in V_0$, $\Delta_j^k(t_2) \leq 0$.

Let k be a plane in $V_0 \cap L[j]$. By the definition of $L[j]$, there is a time-slot $t_3 \in [t_2, t_1]$ such that $\Delta_j^k(t_3) > \frac{N}{r'}$. Let t_4 be the last time-slot such that $\Delta_j^k(t_3) = \Delta_j^k(t_4, t_3)$. If $t_4 < t_2$, then

$$\begin{aligned}\Delta_j^k(t_4, t_3) &= \Delta_j^k(t_4, t_2) + \Delta_j^k(t_2 + 1, t_3)\\ &\leq \Delta_j^k(t_2) + \Delta_j^k(t_2 + 1, t_3) \ \leq \ \Delta_j^k(t_2 + 1, t_3),\end{aligned}$$

and therefore t_4 is not minimal. Hence $t_4 \geq t_2$ and $[t_4, t_3] \subseteq [t_2, t_1]$. Since $\Delta_j^k(t_4, t_3) = A_j^k(t_4, t_3) - \frac{1}{r'} A_j(t_4, t_3) > \frac{N}{r'}$, and $A_j(t_4, t_3) \geq A_j^k(t_4, t_3)$, it follows that $A_j^k(t_4, t_3) > \frac{N}{r'-1}$.

Because $|V_0 \cap L[j]| \geq 2r'$, the number of cells arriving at the switch, destined for output-port j, during time interval $[t_2, t_1]$, is at least $(2r')\frac{N}{r'-1} > 2N$. During time interval $[t_2, t_1]$, no cells are sent to any plane in V_1, and Lemma 5.4 implies that for every plane $k \in V_1$, $\Delta_j^k(t_1) \leq 0$; in particular, this holds for all planes in $Q[j]$. This implies that $Q[j]$ becomes empty, and the phase changes at least once during time interval $[t_2, t_1]$, contradicting the choice of t_1 and t_2.

By Lemma 5.3 and Theorem 4.6, the relative queuing delay of the algorithm is at most $3N + r'$. ☐

5.3. Optimal u-RT demultiplexing algorithm.

Algorithm 1 can be used as a building block for u-RT algorithms with $u > 1$. Algorithm 2 runs u instances of Algorithm 1 in cycles, such that in each time-slot only one instance is active (that is, the ith instance is active on time-slots $i, i + u, i + 2u, i + 3u$ etc.). Since there are u time-slots between two consecutive times in which the same instance is active, global information on the previous time the instance was active can be shared among the demultiplexors. In addition, each instance of Algorithm 1 has its own set of $8r'$ planes; hence Algorithm 2 needs a speedup $S = 8u$.

Algorithm 2 u-RT Algorithm

```
Shared:
    $ALG[u]$: $u$ instances of Algorithm 1   ▷ Each instance with its own planes and shared
        variables
    $x$: value in $\{0, \ldots, u-1\}$, initially $u-1$              ▷ cyclic pointer to array ALG

void procedure ADVANCE-CLOCK( )            ▷ invoked at the beginning of each time-slot
    $x \leftarrow (x + 1) \bmod u$
    $ALG[x]$.ADVANCE-CLOCK() ▷ invoke procedure ADVANCE-CLOCK on the $x$th instance
end procedure

int procedure DISPATCH(cell $c$) at demultiplexor $i$
    return $ALG[x]$.DISPATCH($c$)          ▷ invoke procedure DISPATCH on the $x$th instance
end procedure
```

We next bound the imbalance under Algorithm 2.

LEMMA 5.6. *In Algorithm* 2*, for every plane k and output-port j, $\Delta_j^k < \frac{2N}{r'}$.*

Proof. Assume, toward a contradiction, that there is a traffic T, a plane k, and an output-port j such that $\Delta_j^k \geq \frac{2N}{r'}$. Let t be the first time-slot in which $\Delta_j^k(t) \geq \frac{2N}{r'}$, and let $x = t \bmod u$. The choice of t and Algorithm 2 imply that a cell is sent through plane k at time-slot t by instance x.

Let T' be the traffic consisting of the cells of traffic T handled by the instance x; that is, $T' = \{c \mid c \in T, ta(c) - x \bmod u = 0\}$. Let $round(c) = \frac{ta(c)-x}{u}$ be the number of times instance x was active until cell c arrived to the switch.

Consider traffic $\widetilde{T}$ in which each cell c of traffic T' arrives at the switch at time-slot $round(c)$; that is, $\widetilde{T} = \{\text{shift}(c, round(c) - ta(c)) \mid c \in T'\}$. Let $\widetilde{A}_j(t_1, t_2)$ be the number of cells in traffic $\widetilde{T}$ destined for output-port j that arrive at the switch during time interval $[t_1, t_2]$, and let $\widetilde{A}_j^k(t_1, t_2)$ be the number of these cells that are sent through plane k by Algorithm 1. Similarly, following Definition 4.2, $\widetilde{\Delta}_j^k(t_1, t_2) = \widetilde{A}_j^k(t_1, t_2) - \frac{1}{r'}\widetilde{A}_j(t_1, t_2)$, and $\widetilde{\Delta}_j^k(t_2) = \max_{t_1 \leq t_2} \widetilde{\Delta}_j^k(t_2)$.

Since only instance x sends cells to plane k, and the dispatching decisions of instance x in response to traffic T are the same as the decisions of Algorithm 1 in response to traffic $\widetilde{T}$, it follows that for every time $t' < t$, $A_j^k(t', t) = \widetilde{A}_j^k(\lceil \frac{t'-x}{u} \rceil, \frac{t-x}{u})$. On the other hand, in $\widetilde{T}$ there is a subset of T's cells destined for output-port j, and therefore $A_j(t', t) \geq \widetilde{A}_j(\lceil \frac{t'-x}{u} \rceil, \frac{t-x}{u})$. This implies that $\widetilde{\Delta}_j^k(\frac{t-x}{u}) \geq \Delta_j^k(t) \geq \frac{2N}{r'}$, contradicting Lemma 5.3. □

Lemma 5.6, Theorem 5.5, and Theorem 4.6 immediately imply the following theorem.

THEOREM 5.7. *For any $u \geq 1$ and a PPS with speedup $S = 8u$, there is a u-RT demultiplexing algorithm* `ALG` *such that $\mathcal{R}_{max}($*`ALG`*$) \leq 3N + r'$.*

6. Discussion.

This paper presents lower bounds on the average relative queuing delay which hold with high probability even if randomization is used. This generally implies that, unlike other load balancing problems, randomization does not reduce the relative queuing delay. Our lower bounds use the fact that switches are FCFS but can be generalized to rely on other priority-based policies.

We also present a methodology for bounding the relative queuing delay and use it to show that the lower bounds are tight. In particular, we prove that the relative queuing delay of the FTD algorithm [20] is $(N+1)r'$ time-slots, exactly matching the lower bound for fully distributed algorithms. We also design u-RT demultiplexing algorithms with relative queuing delay of $3N + r' < 4N$ time-slots; for the common case of constant speedup—independent of the size of the switch and its rates—this asymptotically matches the lower bound of $\frac{uN}{S}\left(1 - \frac{u}{r'}\right)$ time-slots for u-RT algorithms, when $1 \leq u \leq \frac{r'}{2}$.

A natural design choice is to transmit global control information on the internal links of the PPS, at rate r. Substituting $u \approx r'$ in our lower bounds for u-RT algorithms yields a relative queuing delay approximately equal to the delay when no information is gathered at all. This implies that without dedicated control links working at a rate much larger than r, it is probably unprofitable to share information between the input-ports.

Acknowledgments. We would like to thank Ilan Newman and Fabian Kuhn for helpful discussions about randomized demultiplexing algorithms. We would also like to thank the anonymous referees for their comments and suggestions.

REFERENCES

[1] 3Com Corporation, *Inverse Multiplexing over ATM (IMA): A Breakthrough WAN Technology for Corporate Networks*, Marlborough, MA, 1997.

[2] The ATM Forum, *Inverse Multiplexing for ATM (IMA) Specification, Version* 1.1, 1999, AF-PHY-0086.001; available online from http://www.ipmplsforum.org/tech/atm_specs.shtml.

[3] H. Attiya and D. Hay, *The inherent queuing delay of parallel packet switches*, IEEE Trans. Parallel Distrib. Systems, 17 (2006), pp. 1048–1056.

[4] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal, *Balanced allocations*, SIAM J. Comput., 29 (1999), pp. 180–200.

[5] S. Ben-David, A. Borodin, R. Karp, G. Tardos, and A. Wigderson, *On the power of randomization in online algorithms*, Algorithmica, 11 (1994), pp. 2–14.

[6] C.S. Chang, D.S. Lee, and Y.S. Jou, *Load balanced Birkhoff-von Neumann switches, part* I*: One-stage buffering*, Computer Communications, 25 (2002), pp. 611–622.

[7] A. Charny, *Providing QoS Guarantees in Input Buffered Crossbar Switches with Speedup*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, 1998.

[8] A. Charny, P. Krishna, N.S. Patel, and R.J. Simcoe, *Algorithms for providing bandwidth and delay guarantees in input-buffered crossbars with speedup*, in Proceedings of the 6th IEEE/IFIP International Workshop on Quality of Service, IEEE Communications Society, New York, NY, 1998, pp. 235–244.

[9] F.M. Chiussi, D.A. Khotimsky, and S. Krishnan, *Generalized inverse multiplexing of switched ATM connections*, in Proceedings of IEEE Globecom, IEEE Communications Society, New York, NY, 1998, pp. 3134–3140.

[10] S. Chuang, A. Goel, N. McKeown, and B. Prabhakar, *Matching output queueing with a combined input output queued switch*, in Proceedings of IEEE INFOCOM, IEEE Communications Society, New York, NY, 1999, pp. 1169–1178.

[11] Cisco Systems, Inc., *ATM Switch Router Software Configuration Guide*, San Jose, CA, 2001, 12.1(6)EY.

[12] C. Clos, *A study of non-blocking switching networks*, Bell System Tech. J., 32 (1953), pp. 406–424.

[13] R.L. Cruz, *A calculus for network delay, part* I*: Network elements in isolation*, IEEE Trans. Inform. Theory, 37 (1991), pp. 114–131.

[14] J. Duncanson, *Inverse multiplexing*, IEEE Communications Magazine, 32 (1994), pp. 34–41.

[15] P. Fredette, *The past, present, and future of inverse multiplexing*, IEEE Communications Magazine, 32 (1994), pp. 42–46.

[16] P. Giaccone, B. Prabhakar, and D. Shah, *Randomized scheduling algorithms for high-aggregate bandwidth switches*, IEEE J. Selected Areas in Communications, 21 (2003), pp. 546–559.

[17] G. Gonnet, *Expected length of the longest probe sequence in hash coding searching*, J. ACM, 28 (1981), pp. 289–304.

[18] S. Iyer, *Personal communication*, 2006.

[19] S. Iyer, A.A. Awadallah, and N. McKeown, *Analysis of a packet switch with memories running slower than the line rate*, in Proceedings of IEEE INFOCOM, IEEE Communications Society, New York, NY, 2000, pp. 529–537.

[20] S. Iyer and N. McKeown, *Making parallel packet switches practical*, in Proceedings of IEEE INFOCOM, IEEE Communications Society, New York, NY, 2001, pp. 1680–1687.

[21] S. Iyer and N. McKeown, *Analysis of the parallel packet switch architecture*, IEEE/ACM Trans. Networking, 11 (2003), pp. 314–324.

[22] I. Keslassy, *Load Balanced Router*, Ph.D. thesis, Stanford University, Palo Alto, CA, 2004.

[23] I. Keslassy and N. McKeown, *Maintaining packet order in two-stage switches*, in Proceedings of IEEE INFOCOM, IEEE Communications Society, New York, NY, 2002.

[24] D. Khotimsky, *Personal communication*, 2004.

[25] D. Khotimsky and S. Krishnan, *Stability analysis of a parallel packet switch with bufferless input demultiplexors*, in Proceedings of the IEEE International Conference on Communications (ICC), IEEE Communications Society, New York, NY, 2001, pp. 100–106.

[26] L. Kleinrock, *Queuing Systems, Volume* II, John Wiley & Sons, New York, 1975.

[27] P. Krishna, N.S. Patel, A. Charny, and R.J. Simcoe, *On the speedup required for work-conserving crossbar switches*, IEEE J. Selected Areas in Communications, 17 (1999), pp. 1057–1066.

[28] Lucent Technologies, *Inverse Multiplexing for ATM, Expanding the Revenue Opportunities for Converged Services over ATM*, Murray Hill, NJ, 2001.

[29] M.D. Mitzenmacher, *On the analysis of randomized load balancing schemes*, Theory Comput. Syst., 32 (1999), pp. 361–386.

[30] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, Cambridge, UK, 1995.

[31] PMC-Sierra, Inc., *Inverse Multiplexing Over ATM Works Today*, available online at http://www.electronicstalk.com/news/pmc/pmc121.html, 2002.

[32] B. Prabhakar and N. McKeown, *On the speedup required for combined input and output queued switching*, Automatica J. IFAC, 35 (1999), pp. 1909–1920.

[33] A. Prakash, A. Aziz, and V. Ramachandran, *A near optimal schedule for switch-memory-switch routers*, in Proceedings of the ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), ACM, New York, 2003, pp. 343–352.

[34] A. Prakash, A. Aziz, and V. Ramachandran, *Randomized parallel schedulers for switch-memory-switch routers: Analysis and numerical studies*, in Proceedings of IEEE INFOCOM, IEEE Communications Society, New York, NY, 2004.

[35] S. Sharif, A. Aziz, and A. Prakash, *An $O(\log^2 N)$ parallel algorithm for output queuing*, in Proceedings of IEEE INFOCOM, IEEE Communications Society, New York, NY, 2002.

[36] D.C. Sthephens and H. Zhang, *Implementing distributed packet fair queueing in a scalable architecture*, in Proceedings of IEEE INFOCOM, IEEE Communications Society, New York, NY, 1998, pp. 282–290.

[37] I. Stoica and H. Zhang, *Exact emulation of an output queueing switch by a combined input output queueing switch*, in Proceedings of the 6th IEEE/IFIP International Workshop on Quality of Service, IEEE Communications Society, New York, NY, 1998, pp. 218–224.

[38] L. Tassiulas, *Linear complexity algorithms for maximum throughput in radio networks and input queued switches*, in Proceedings of IEEE INFOCOM, IEEE Communications Society, New York, NY, 1998, pp. 533–539.

SIAM J. COMPUT.
Vol. 37, No. 5, pp. 1637–1655

A PRACTICAL SHORTEST PATH ALGORITHM WITH LINEAR EXPECTED TIME*

ANDREW V. GOLDBERG†

Abstract. We present an improvement of the multilevel bucket shortest path algorithm of Denardo and Fox [*Oper. Res.*, 27 (1979), pp. 161–186] and justify this improvement both theoretically and experimentally. We prove that if the input arc lengths come from a natural probability distribution, the new algorithm runs in linear average time while the original algorithm does not. We also describe an implementation of the new algorithm. Our experimental data suggests that the new algorithm is preferable to the original one in practice. Furthermore, for integral arc lengths that fit into a word of today's computers, the performance is close to that of breadth-first search, suggesting limitations on further practical improvements.

Key words. algorithms, data structures, shortest paths, experimental evaluation

AMS subject classifications. 68W40, 68Q25

DOI. 10.1137/070698774

1. Introduction.

The shortest path problem with nonnegative arc lengths (*the NSP problem*) is very common in practice, and algorithms for this problem have been extensively studied both from theoretical, e.g., [2, 6, 9, 10, 12, 13, 17, 25, 29, 30, 34, 37, 38, 39, 40, 42, 43, 44, 45], and computational, e.g., [4, 5, 11, 19, 20, 24, 27, 28, 31, 35, 36, 46], viewpoints. Efficient implementations of Dijkstra's algorithm [12], in particular, have received a lot of attention.

Suppose that the input graph has n vertices and m arcs. To state some of the previous results, we assume that the input arc lengths are integral. Let U denote the biggest arc length. We define C to be the ratio between U and the smallest nonzero arc length, δ. Note that if the lengths are integral, then $C \leq U$. Modulo precision problems and arithmetic operation complexity, our results apply to real-valued arc lengths as well. To simplify comparing time bounds with and without U (or C), one can make the following similarity assumption [18]: $\log U = O(\log n)$.

Several algorithms for the problem have near-linear worst-case running times, although no algorithm has a linear running time if the graph is directed and the computational model is well established. In the pointer model of computation, the Fibonacci heap data structure of Fredman and Tarjan [16] leads to an $O(m + n \log n)$ implementation of Dijkstra's algorithm. In a RAM model with unit-time word operations, the fastest currently known algorithms achieve the following bounds: $O(m + n(\log U \log\log U)^{1/3})$ [39], $O(m + n(\sqrt{\log n}))$ [38], $O(m \log\log U)$ [25], and $O(m \log\log n)$ [43].

For undirected graphs, Thorup's algorithm [42] has a linear running time in a word RAM model. A constant-time priority queue of [3] yields a linear-time algorithm for directed graphs, but only in a nonstandard computation model that is not supported by any existing computer.

Our work has been motivated by a recent paper of Meyer [30]. The paper gives an NSP algorithm with a linear average time for input arc lengths drawn independently

*Received by the editors November 21, 2001; accepted for publication (in revised form) July 29, 2007; published electronically February 14, 2008. Preliminary versions of theoretical and experimental results presented in this paper appeared in [21, 22] and [23], respectively.

http://www.siam.org/journals/sicomp/37-5/69877.html

†Microsoft Research, 1075 la Avenida, Mountain View, CA 94043 (goldberg@microsoft.com).

from a uniform distribution on $[1, \ldots, M]$. It also proves that, under the same assumptions, the running time is linear with high probability (w.h.p.). Meyer's algorithm may scan some vertices more than once, and its worst-case time bound, $O(nm \log n)$, is far from linear. Both the algorithm and its analysis are complicated.

In this paper we show that a natural improvement of the *multilevel bucket (MLB)* shortest path algorithm of [9] has an average running time that is linear and a worst-case time of $O(m + n \log C)$. Our average-time bound holds for arc lengths distributed uniformly on $[1, \ldots, M]$; the lengths do not need to be independent. We also show that if the lengths are independent, the algorithm running time is linear w.h.p. We refer to the new algorithm as the *smart queue algorithm*. Unlike the MLB algorithm, our algorithm is not an implementation of Dijkstra's algorithm: a vertex selected for scanning is not necessarily a minimum labeled vertex. However, the selected vertex distance label is equal to the correct distance, and each vertex is scanned at most once. A similar relaxation of Dijkstra's algorithm was originally introduced by Dinic [13] and used in its full strength by Thorup [42].

A recent paper of Hagerup [26] gives a different NSP algorithm with linear expected time. This algorithm uses the power of word operations to a greater extent than our algorithm, which allows the use of a simpler bucket structure.

The practical importance of the NSP problem motivated extensive computational work. Implementations of Dijkstra's algorithm based on the classical binary heap data structure [45] were often used as a benchmark to compare other work against. This algorithm is easy to code and has reasonably constant factors.[1] In practice binary heaps usually outperform Fibonacci heaps [16] and often outperform pairing heaps [40]. However, the running time of the algorithm based on binary heaps grows superlinearly with the number of elements on the heap, and the algorithm is not hard to beat on bigger problems.

In many practical problems, label-correcting algorithms of Pape [36], Pallottino [35], and Glover and Klingman [20] perform extremely well. However, these algorithms have bad worst-case time bounds and sometimes perform poorly. Furthermore, unlike implementations of Dijkstra's algorithm and the smart queue algorithm, these algorithms cannot be terminated early if one desires a shortest path between a pair of vertices.

Dail's bucket-based algorithm [10] works well if U is small. Cherkassky, Goldberg, and Radzik [5] show that the two-level bucket algorithm has a reasonable performance for moderately large U. Further theoretical and experimental work [6, 24] produced more robust two- and three-level implementations. Although they have much better worst-case performance than the label-correcting algorithms mentioned above, these MLB implementations are somewhat slower on many practical problems. For example, based on real-life road networks, Zhan and Noon [46] recommend Pallottino's algorithm for the NSP problem and bucket-based algorithms for the single pair problem.

We describe a practical implementation of the smart queue algorithm. As a step toward this implementation, we develop a new implementation of the MLB algorithm that is more efficient than the previous implementations if the number of bucket levels is large. A simple modification of this implementation yields an efficient implementation of the smart queue algorithm. Our experimental results show that the smart queue algorithm with a large number of levels is practical. In particular, an implementation with the number of levels optimized for the worst-case theoretical performance

[1] Implementations based on 4-heaps have better constants.

works well on both typical and bad-case inputs. For 32-bit arc lengths, the code runs in time less than 2.5 times that of breadth-first search for all inputs we tried. These inputs included ones designed to be hard for our implementation. Our results lead to a better understanding of NSP algorithm implementations and show how close their performance is to the lower bound provided by breadth-first search.

Our algorithm has been motivated by a theoretical average-case analysis. The fact that the algorithm is also practical is an interesting example of a situation where probabilistic analysis leads to improved practical performance.

2. Preliminaries. The input to the NSP problem is a directed graph $G = (V, A)$ with n vertices, m arcs, a source vertex s, and nonnegative arc lengths $\ell(a)$. The goal is to find shortest paths from the source to all vertices of the graph. Unless mentioned otherwise, we assume that arc lengths are integers in the interval $[1, \dots, U]$, where U denotes the biggest arc length. Let δ be the smallest nonzero arc length, and let C be the ratio of the biggest arc length to δ. If all arc lengths are zero or if $C < 2$, then the problem can be solved in linear time [13] (for arc lengths in the range $[L, U]$ with $U < 2L$, the linear-time algorithm uses width L single-level buckets). Without loss of generality, we assume that $C \geq 2$ and $\log C \geq 1$, which is the only technical reason for making the assumption, as it simplifies the bounds. This implies that $\log U \geq 1$. We say that a statement holds *with high probability (w.h.p.)* if the probability that the statement is true approaches one as $m \to \infty$.

We assume the *word RAM* model of computation (see, e.g., [1]). To efficiently implement the MLB data structure [9], we need array addressing and the following unit-time word operations: addition, subtraction, comparison, and arbitrary shifts. To allow a higher-level description of our algorithm, we use a *strong RAM* computation model that also allows word operations including bitwise logical operations and the operation of finding the index of the most significant bit in which two words differ. The latter operation is in AC0; see [8] for a discussion of a closely related operation. The use of this more powerful model does not improve the amortized operation bounds but simplifies the description.

3. Labeling method and related results. The labeling method for the shortest path problem [14, 15] works as follows (see, e.g., [41]). The method maintains for every vertex v its distance label $d(v)$, parent $p(v)$, and status $S(v) \in \{\texttt{unreached}, \texttt{labeled}, \texttt{scanned}\}$. Initially $d(v) = \infty$, $p(v) = nil$, and $S(v) = \texttt{unreached}$. The method starts by setting $d(s) = 0$ and $S(s) = \texttt{labeled}$. While there are labeled vertices, the method picks such a vertex v, scans all arcs out of v, and sets $S(v) = \texttt{scanned}$. To scan an arc (v, w), one checks if $d(w) > d(v) + \ell(v, w)$ and, if true, sets $d(w) = d(v) + \ell(v, w)$, $p(w) = v$, and $S(w) = \texttt{labeled}$.

If the length function is nonnegative, the labeling method always terminates with correct shortest path distances and a shortest path tree. The efficiency of the method depends on the rule to choose a vertex to scan next. We say that $d(v)$ is *exact* if the distance from s to v is equal to $d(v)$. It is easy to see that if the method always selects a vertex v such that, at the selection time, $d(v)$ is exact, then each vertex is scanned at most once.

Dijkstra [12] observed that if ℓ is nonnegative and v is a labeled vertex with the smallest distance label, than $d(v)$ is exact. However, a linear-time implementation of Dijkstra's algorithm in the strong RAM model is at least as hard as linear-time sorting. Dinic [13] and Thorup [42] use a relaxation of Dijkstra's selection rule to get linear-time algorithms for special cases of the NSP problem. To describe a related relaxation that we use, define the *caliber* of a vertex v, $c(v)$, to be the minimum length of an arc entering v, or infinity if no arc enters v.

LEMMA 3.1 (caliber lemma). *Suppose ℓ is nonnegative, and let μ be a lower bound on distance labels of labeled vertices. Let v be a vertex such that $\mu + c(v) \geq d(v)$. Then $d(v)$ is exact.*

The lemma follows from the observation that for any labeled vertex u, such that $(u, v) \in A$, $d(u) + \ell(u, v) \geq \mu + c(v) \geq d(v)$.

4. Algorithm description and correctness. Our algorithm is based on the MLB implementation of Dijkstra's algorithm modified to use Lemma 3.1 to detect and scan vertices with exact (but not necessarily minimum) distance labels. Our algorithm is a labeling algorithm. During the initialization, the algorithm also computes $c(v)$ for every vertex v. The algorithm keeps labeled vertices in one of two places: a set F and a priority queue B. The former is implemented to allow constant time additions and deletions, for example, as a doubly linked list. The latter is implemented using multilevel buckets as described below. The priority queue supports operations `insert`, `delete`, `decrease-key`, and `extract-min`. However, the `insert` operation inserts vertices into either B or F, and the `decrease-key` operation may move vertices from B to F.

At a high level, the algorithm works as follows. Vertices in F have exact distance labels and if F is nonempty, we remove and scan a vertex from F. If F is empty, we remove and scan a vertex from B with the minimum distance label. Suppose a distance label of a vertex u decreases. Note that u cannot belong to F. If u belongs to B, then we apply the `decrease-key` operation to u. This operation either relocates u within B or discovers that u's distance label is exact and moves u to F. If u is in neither B nor F, we apply the `insert` operation to u, and u is inserted either into B or, if $d(u)$ is determined to be exact, into F.

Next we describe the bucket structure B. For a given integer parameter $\Delta \geq 2$, B contains $k + 1$ levels of buckets, where $k = \lceil \log_\Delta U \rceil$. Except for the top level, a level contains Δ buckets. Conceptually, the top level contains infinitely many buckets. However, at any given time all buckets on that level are empty except at most three consecutive ones (see Lemma 4.1 below), and one can maintain only these buckets by wrapping around modulo three at the top level.[2] We denote bucket j at level i by $B(i, j)$; i ranges from 0 (bottom level) to k (top), and j ranges from 0 to $\Delta - 1$, except at the top level discussed above. A bucket contains a set of vertices maintained in a way that allows constant-time insertion and deletion, e.g., in a doubly linked list. At each level i, we maintain the number of vertices at this level.

We maintain μ such that μ is a lower bound on the distance labels of labeled vertices. Initially $\mu = 0$. Every time an `extract-min` operation removes a vertex v from B, we set $\mu = d(v)$. Consider the base Δ representation of the distance labels and number digit positions starting from 0 for the least significant digit. Let $\mu_{i,j}$ denote the ith through jth least significant digit of μ, and let μ_i denote the ith least significant digit. Similarly, $d_i(u)$ denotes the ith least significant digit of $d(u)$, and likewise for the other definitions. Note that μ and the $k + 1$ least significant digits of the base Δ representation of $d(u)$ uniquely determine $d(u)$: $d(u) = \mu + (d_{0,k} - \mu_{0,k})$ if $d_{0,k} > \mu_{0,k}$ and $d(u) = \mu + \Delta^k + (d_{0,k} - \mu_{0,k})$ otherwise.

For a given μ, let $\underline{\mu_i}$ and $\overline{\mu_i}$ be μ with the i least significant digits replaced by 0 or $\Delta - 1$, respectively. Each level $i < k$ corresponds to the range of values $[\underline{\mu_{i+1}}, \overline{\mu_{i+1}}]$.

[2] For low-level efficiency, one may want Δ to be a power of two and wrap around modulo four.

Each bucket $B(i,j)$ corresponds to the subrange containing all integers in the range with the ith digit equal to j. At the top level, a bucket $B(k,j)$ corresponds to the range $[j \cdot \Delta^k, (j+1) \cdot \Delta^k)$. The *width* of a bucket at level i is equal to Δ^i: the bucket contains Δ^i distinct values. We say that a vertex u *is in the range of* $B(i,j)$ if $d(u)$ belongs to the range corresponding to the bucket.

The position of a vertex u in B depends on μ: u belongs to the lowest-level bucket containing $d(u)$. More formally, let i be the index of the most significant digit in which $d(u)$ and $\mu_{0,k}$ differ, or 0 if they match. Note that $\underline{\mu_i} \leq d(u) \leq \overline{\mu_i}$. Given μ and u with $d(u) \geq \mu$, we define the *position of* u by $(i, d_i(u))$ if $i < k$ and $B(k, \lfloor (d(u) - \mu)/\Delta^k \rfloor)$ otherwise. If u is inserted into B, it is inserted into $B(i,j)$, where (i,j) is the position of u. For each vertex in B, we store its position.

Figure 1 gives an example of the bucket structure. In this example, $\Delta = 2$, $k = 3$, and $\mu = 10$. For instance, to find the position of a vertex v with $d(v) = 14$, we note that the binary representations of 10 and 14 differ in bit 2 (remember that we start counting from 0) and the bit value is 1. Thus v belongs to bucket 1 at level 2.

Our modification of the MLB algorithm uses Lemma 3.1 during the `insert` operation to put vertices into F whenever the lemma allows it. The details are as follows.

`insert`. Insert a vertex u into $B \cup F$ as follows. If $\mu + c(u) \geq d(u)$, put u into F. Otherwise compute u's position (i,j) in B and add u to $B(i,j)$.

`decrease-key`. Decrease the key of an element u in position (i,j) as follows. Remove u from $B(i,j)$. Set $d(u)$ to the new value and insert u as described above.

`extract-min`. Find the lowest nonempty level i. Find j, the first nonempty bucket at level i, by setting j to the index of the bucket at level i that contains μ, and increment j until the bucket $B(i,j)$ is nonempty. If $i = 0$, delete a vertex u from $B(i,j)$. (In this case $\mu = d(u)$.) Return u. If $i > 0$, examine all elements of $B(i,j)$ and delete a minimum element u from $B(i,j)$. Note that in this case $\mu < d(u)$; set $\mu = d(u)$. Since μ increased, some vertex positions in B may have changed. We do *bucket expansion* of $B(i,j)$ and return u.

To understand bucket expansion, note that the vertices with changed positions are exactly those in $B(i,j)$. To see this, let μ' be the old value of μ and consider a vertex v in B. Let (i',j') be v's position with respect to μ'. By the choice of $B(i,j)$, if $(i,j) \neq (i',j')$, then either $i < i'$ or $i = i'$ and $j < j'$. In both cases, the common prefix of μ' and $d(v)$ is the same as the common prefix of $d(u)$ and $d(v)$, and the position of v does not change.

On the other hand, vertices in $B(i,j)$ have a longer common prefix with $d(u)$ than they have with μ' and these vertices need to move to a *lower* level. Bucket expansion

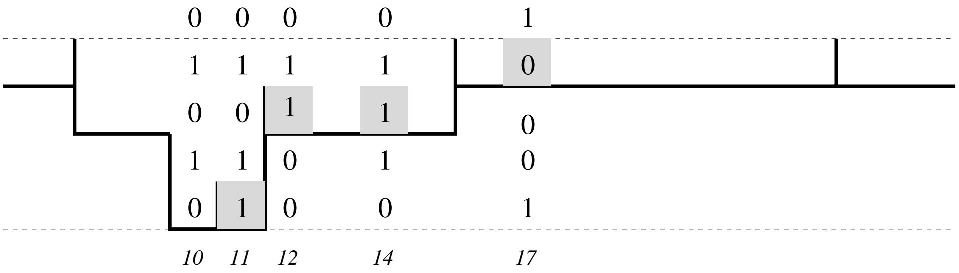


FIG. 1. *MLB example.* $\Delta = 2$, $k = 3$, $\mu = 10$. *Values on the bottom are in decimal. Values on top are in binary, with the least significant bit on the bottom. Shaded bits determine positions of the corresponding elements.*

deletes these vertices from $B(i)$ and uses the `insert` operation to add the vertices back into B or into F, as appropriate. The vertex u in $B(i,j)$ with the minimum distance label is not inserted into B but is returned and scanned. Note that we do bucket expansions only when F is empty and the expanded bucket contains a labeled vertex with the minimum distance. Thus μ is updated correctly.

Although the formal description of the algorithm is nontrivial, the algorithm itself is simple: At each step, remove a vertex from F or, if F is empty, then remove the minimum-labeled vertex from B. In the latter case, expand the bucket from which the vertex has been removed, if necessary. Scan the vertex and update its neighbors if necessary. Terminate when both F and B are empty.

The width of a top-level bucket is at least U. In the original MLB algorithm, at any point of the execution all labeled vertices are contained in at most two consecutive top-level buckets. A slightly weaker result holds for our algorithm.

Lemma 4.1. *At any point of the execution, all labeled vertices are in the range of at most three consecutive top-level buckets.*

Proof. Let μ' be the current value of μ and let $B(k,j)$ be the top-level bucket containing μ'. Except for s (for which the result holds trivially), a vertex v becomes labeled during a scan of another vertex u removed from either B or F. In the former case, at the time of the scan $d(u) = \mu \leq \mu'$, $d(v) = \mu + \ell(u,v) \leq \mu' + U$, and therefore v is contained in either $B(k,j)$ or $B(k,j+1)$. In the latter case, when u has been added to F, the difference between $d(u)$ and μ was at most $c(u) \leq U$, and thus $d(u) \leq \mu' + U$, $d(v) \leq d(u) + U \leq \mu' + 2 \cdot U$, and thus v belongs to $B(k,j)$, $B(k,j+1)$, or $B(k,j+2)$. □

Algorithm correctness follows from Lemmas 3.1 and 4.1 and the observations that μ is always set to the minimum distance label of a labeled vertex, μ remains a lower bound on the labeled vertex labels (and therefore is monotonically nondecreasing), and F always contains vertices with exact distance labels.

Remark. An alternative interpretation of the algorithm is as follows. For each vertex v added to F, we apply a potential transformation as follows. Let $\delta = d(v) - \mu$, where the value of μ is taken at the time v is added to F. We reduce the length of all arcs into v by δ and increase the length of all arcs out of v by δ. Note that the transformed lengths are nonnegative as $d(v) - \mu \leq c(v)$. With respect to the transformed lengths, our algorithm follows Dijkstra's rule of always scanning a vertex with the minimum distance label. This interpretation gives an alternative proof of correctness of the algorithm.

5. Worst-case analysis. In this section we prove a worst-case bound on the running time of the algorithm. Our analysis is similar to that for the MLB algorithm given in [6], except the resulting bound has a C instead of a U. Some definitions and lemmas introduced in this section will also be used in the next section.

We start the analysis with the following lemmas.

Lemma 5.1 (see [6]).

- *Given μ and u, we can compute the position of u with respect to μ in constant time.*
- *We can find the lowest nonempty level of B in constant time.*

Lemma 5.2. *The algorithm runs in $O(m + n + \Phi_1 + \Phi_2)$ time, where Φ_1 is the total number of times a vertex moves from a bucket of B to a lower-level bucket and Φ_2 is the number of empty buckets examined by the algorithm.*

Proof. Since each vertex is scanned at most once, the total scan time is $O(m+n)$. A vertex is added to and deleted from F at most once, so the total time devoted to maintaining F is $O(n)$. An `insert` operation takes constant time, and these operations are caused by inserting vertices into B for the first time by `decrease-key` operations and by `extract-min` operations. The former take $O(n)$ time; we account for the remaining ones jointly with the other operations. A `decrease-key` operation takes constant time and is caused by a decrease of $d(v)$ due to a scan of an arc (u, v). Since an arc is scanned at most once, these operations take $O(m)$ total time. The work we accounted for so far is linear.

Next we consider the `extract-min` operations. Consider an `extract-min` operation that returns u. The operation takes $O(1)$ time, plus the time proportional to the number of empty buckets examined, plus the time proportional to the number of vertices in the expanded bucket, excluding u. Each of these vertices moves to a lower level in B. Thus we get the desired time bound. □

Note that $\Phi_1 = O(nk)$ and $\Phi_2 = O(n\Delta)$ since after examining less than Δ empty buckets we discover a nonempty one and scan a vertex from it. Setting Δ to $\lceil \frac{\log U}{\log\log U} \rceil$ balances Φ_1 and Φ_2 and yields the $O(m + n\frac{\log U}{\log\log U})$ worst-case time bound. To get a better bound, we define $k' = \lfloor \log_\Delta \delta \rfloor$.

Lemma 5.3. *Buckets at level k' and below are never used.*

Proof. Let (i, j) be the position of a vertex v of caliber $c(v) \geq \delta$. If $i \leq k'$, then $d(v) - \mu < \Delta^i \leq \Delta^{k'} \leq \delta \leq c(v)$ and the algorithm adds v to F, not B. □

The lemma implies that the algorithm uses $O(\log_\Delta U - \log_\Delta \delta) = O(\log_\Delta C)$ bucket levels.

Setting Δ to $\lceil \frac{\log C}{\log\log C} \rceil$ and applying Lemma 5.3, we get the following result.

Theorem 5.4. *The worst-case running time of the algorithm is $O(m+n\frac{\log C}{\log\log C})$.*

Note that setting $\Delta = 2$ yields an $O(m + n \log C)$ bound.

Our optimization can also be used to improve other data structures based on multilevel buckets, such as radix heaps [2] and hot queues [6]. For these data structures, the equivalent of Lemma 5.3 allows one to replace time bound parameter U by C. In particular, the bound of the hot queue implementation of Raman [39] improves to $O(m + n(\log C \log\log C)^{1/3})$. The modification of Raman's algorithm to obtain this bound is straightforward given the results of this section.

6. Average-case analysis. In this section we prove that for $\Delta = 2$ (or any other constant), the smart queue algorithm runs in linear average time under the assumption that the input arc lengths are uniformly distributed on $[1, \ldots, M]$.[3] We also show that the running time is linear w.h.p. with the additional assumption that the lengths are independent.

Since $\Delta = 2$, $\Phi_2 = O(n)$; it remains to bound Φ_1.

A key lemma for our analysis is as follows.

Lemma 6.1. *The algorithm never inserts a vertex v into a bucket at a level less than or equal to $\log c(v) - 1$.*

Proof. Suppose during an `insert` operation, v's position in B is (i, j) with $i \leq \log c(v) - 1$. Then the most significant digit in which $d(v)$ and μ differ is digit i and $d(v) - \mu < \Delta^{i+1} \leq c(v)$. Therefore `insert` puts v into F, not B. □

The above lemma motivates the following definitions. The *weight of an arc a*, $w(a)$, is defined by $w(a) = k - \lfloor \log \ell(a) \rfloor$. The *weight of a vertex v*, $w(v)$, is defined to be the maximum weight of an incoming arc or zero if v has no incoming arcs.

[3] As we shall see, if M is large enough, then the result also applies to the range $[0, \ldots, M]$.

Lemma 6.1 implies that the number of times v can move to a lower level of B is at most $w(v)+1$, and therefore $\Phi_1 \leq m + \sum_V w(v)$. Note that k depends on the input, and thus the weights are defined with respect to a given input.

For the probability distribution of arc weights defined above, we have $\mathbf{Pr}[\lfloor \log \ell(a) \rfloor = i] = 2^i/M$ for $i = 0, \dots, k-1$. The definition of w yields

$$\mathbf{Pr}[w(a) = t] = 2^{k-t}/M \quad \text{for } t = 1, \dots, k. \tag{1}$$

Since $M \geq U$, we have $M \geq 2^{k-1}$, and therefore

$$\mathbf{Pr}[w(a) = t] \leq 2^{-t+1} \quad \text{for } t = 1, \dots, k. \tag{2}$$

Theorem 6.2. *If arc lengths are uniformly distributed on* $[1, \dots, M]$*, then the average running time of the algorithm is linear.*

Proof. Since $\Phi \leq m + \sum_V w(v)$, it is enough to show that $\mathbf{E}[\sum_V w(v)] = O(m)$. By the linearity of expectation and the definition of $w(v)$, we have $\mathbf{E}[\sum_V w(v)] \leq \sum_A \mathbf{E}[w(a)]$. The expected value of $w(a)$ is

$$\mathbf{E}[w(a)] = \sum_{i=1}^{k} i \mathbf{Pr}[w(a) = i] \leq \sum_{i=1}^{\infty} i 2^{-i+1} = 2 \sum_{i=1}^{\infty} i 2^{-i} = O(1).$$

Note that this bound holds for any k. Thus $\sum_A \mathbf{E}[w(a)] = O(m)$. □

Remark. Note that Theorem 6.2 does not require arc lengths to be independent. Our proof of its high-probability variant, Theorem 6.7, requires this independence.

Remark. The proof of the theorem works for any arc length distribution such that $\mathbf{E}[w(a)] = O(1)$. In particular, the theorem holds for real-valued arc lengths selected uniformly from $(0, 1]$. (We exclude zero so that w is well defined.) In fact, for this distribution the high-probability analysis below is simpler (given the independence assumption). However, the integer distribution is somewhat more interesting, because some test problem generators use this distribution and most practical problems have integral lengths.

Next we show that the algorithm running time is linear w.h.p. by showing that $\sum_A w(a) = O(m)$ w.h.p. First, we show that w.h.p. U is not much smaller than M and δ is close to Mm^{-1} (Lemmas 6.3 and 6.4). Let S_t be the set of all arcs of weight t and note that $\sum_A w(a) = \sum_t t|S_t|$. We show that as t increases, the expected value of $|S_t|$ goes down exponentially. For small values of t, the value of $|S_t|$ is also bounded by an exponentially decreasing function w.h.p. To deal with large values of t, we show that the total number of arcs with large weights is small, and so is the contribution of these arcs to the sum of arc weights.

Proofs of the following two lemmas are fairly standard; we include them to make the paper self-contained.

Lemma 6.3. *W.h.p.,* $U \geq M/2$.

Proof. For an arc a, $\mathbf{Pr}[\ell(a) < M/2] < 1/2$, and by the independence of arc lengths,

$$\mathbf{Pr}[U < M/2] \leq 2^{-m} \to 0 \quad \text{as } m \to \infty.$$

Thus $\mathbf{Pr}[U \geq M/2] \to 1$ as $m \to \infty$. □

LEMMA 6.4. *W.h.p., $\delta \geq Mm^{-4/3}$. If $M \geq m^{2/3}$, then w.h.p. $\delta \leq Mm^{-2/3}$.*

Proof. For an arc a, we have

$$\mathbf{Pr}\Big[\ell(a) \geq Mm^{-4/3}\Big] = (M - Mm^{-4/3})/M \geq 1 - m^{-4/3}.$$

Since the arc lengths are independent, we have

$$\mathbf{Pr}\Big[\delta \geq Mm^{-4/3}\Big] \geq (1 - m^{-4/3})^m \to 1 \quad \text{as } m \to \infty.$$

Similarly,

$$\mathbf{Pr}\Big[\ell(a) > Mm^{-2/3}\Big] = (M - \lfloor Mm^{-2/3} \rfloor)/M \leq 1 - \frac{m^{-2/3}}{2},$$

and by the independence

$$\mathbf{Pr}\Big[\delta > Mm^{-2/3}\Big] \geq \left(1 - \frac{1}{2m^{2/3}}\right)^m \to 0 \quad \text{as } m \to \infty. \qquad \square$$

From (1), we have $\mathbf{E}[|S_t|] = m2^{k-t}/M$. Since the weights are independent, we can apply the Chernoff bound (see, e.g., [7, 33]) to conclude that $\mathbf{Pr}\big[|S_t| \geq 2m2^{k-t}/M\big] < (\frac{e}{4})^{m2^{k-t}/M}$. Since $M \geq 2^{k-1}$, we have

$$\mathbf{Pr}\big[|S_t| \geq 4m2^{-t}\big] < \left(\frac{e}{4}\right)^{2m2^{-t}}.$$

As mentioned above, we bound the contributions of arcs with large and small weights to $\sum_A w(a)$ differently. We define $\beta = \log(m^{2/3})$ and partition A into two sets—A_1, containing the arcs with $w(a) \leq \beta$, and A_2, containing the arcs with $w(a) > \beta$.

LEMMA 6.5. *$\sum_{A_1} w(a) = O(m)$ w.h.p.*

Proof. Assume that $\delta \geq Mm^{-4/3}$ and $U \geq M/2$; by Lemmas 6.3 and 6.4 this happens w.h.p. This assumption implies $C \leq m^{4/3}$. The probability that for some t, $1 \leq t \leq \beta$, $|S_t| \geq 4m2^{-t}$ is, by the union bound and the fact that the probability is maximized for $t = \beta$, less than

$$\beta \left(\frac{e}{4}\right)^{m2^{-\beta}} \leq \log(m^{2/3}) \left(\frac{e}{4}\right)^{mm^{-2/3}} \leq \log m \left(\frac{e}{4}\right)^{m^{1/3}} \to 0 \quad \text{as } m \to \infty.$$

Thus w.h.p., for all t, $1 \leq t \leq \beta$, we have $|S_t| < 4m2^{-t}$ and

$$\sum_{A_1} w(a) = \sum_{t=1}^{t \leq \beta} t|S_t| \leq 4m \sum_{t=1}^{\infty} t2^{-t} = O(m). \qquad \square$$

LEMMA 6.6. *$\sum_{A_2} w(a) = O(m)$ w.h.p.*

Proof. If $M < m^{2/3}$, then $k \leq \beta$ and A_2 is empty, so the lemma holds trivially.

Now consider the case $M \geq m^{2/3}$. By Lemmas 6.3 and 6.4, w.h.p. $Mm^{-4/3} \leq \delta \leq Mm^{-2/3}$ and $U \geq M/2$; assume that this is the case. The assumption implies $m^{2/3}/2 \leq C \leq m^{4/3}$. Under this assumption, we also have $2^{k-1} \leq M \leq 2^{k+1}$. Combining this with (1) we get $2^{-2-t} \leq \mathbf{Pr}[w(a) = t] \leq 2^{1-t}$. This implies that

$$2^{-2-\beta} \leq \mathbf{Pr}[w(a) > \beta] \leq 2^{2-\beta};$$

therefore,

$$\frac{m^{-2/3}}{8} \le \mathbf{Pr}[w(a) > \beta] \le 4m^{-1/3}$$

and

$$\frac{m^{1/3}}{8} \le \mathbf{E}[|A_2|] \le 4m^{2/3}.$$

Using the independence and applying the Chernoff bound, we get

$$\mathbf{Pr}[|A_2| > 2\mathbf{E}[|A_2|]] < \left(\frac{e}{4}\right)^{\mathbf{E}[|A_2|]}.$$

Replacing the first occurrence of $\mathbf{E}[|A_2|]$ by the upper bound on its value and the second occurrence by the lower bound (since $e/4 < 1$), we get

$$\mathbf{Pr}\left[|A_2| > 8m^{2/3}\right] < \left(\frac{e}{4}\right)^{m^{1/3}/8} \to 0 \quad \text{as } m \to \infty.$$

For all arcs a, $\ell(a) \ge \delta$, and thus

$$w(a) = k - \lfloor \ell(a) \rfloor \le 1 + \log U + 1 - \log \delta = 2 + \log C \le 2 + (4/3)\log m.$$

Therefore, w.h.p.,

$$\sum_{A_2} w(a) \le 8m^{2/3}(2 + (4/3)\log m) = o(m). \qquad \square$$

Thus we have the following theorem.

THEOREM 6.7. *If arc lengths are independent and uniformly distributed on* $[1, \ldots, M]$*, then w.h.p., the algorithm runs in linear time.*

Remark. The expected and high-probability bounds also apply if the arc lengths come from $[0, \ldots, U]$ and $U = \omega(m)$, as in this case w.h.p. no arc has zero length.

7. Algorithm implementation. In this section we describe efficient implementations of the MLB algorithm and smart queue algorithms, including algorithm engineering considerations involved in their development. We also discuss how the MLB implementation differs from the previous implementations [6, 24]. The MB code implements the algorithm MLB algorithm, and the SQ code implements the smart queue algorithm.

Previous implementations of the MLB algorithm, as well as those described below, use the following *wide bucket* heuristic. Pick w such that $0 < w \le \delta$. Then the MLB algorithm remains correct if one multiplies the bucket width on every level by w. Both MB and SQ codes use the wide bucket heuristic. We refer to the modifications needed to convert the MLB algorithm to the smart queue algorithm as the *caliber heuristic.* This heuristic is the only difference between MB and SQ.

Our implementation of MB is very similar to that of [24], except in the details of the `insert` operation. The previous implementation maintained a range of distance values for each level, updating the ranges when the value of μ changed. To insert a vertex, one looks for the lowest level to which the vertex belongs and then computes the offset of the bucket to which the vertex belongs. In contrast, MB and SQ compute the vertex position with respect to μ as described in section 4. This is slightly more efficient when the number of levels is large. The efficiency gain is bigger for SQ because it does not necessarily examine all levels for a given value of μ. The new implementation is also simpler than the old one.

We always set Δ to a power of two. This allows us to use bit shifts instead of divisions. Our codes set w to the biggest power of two not exceeding δ, or to one if $\delta \leq 1$. We use an array to represent each level of buckets.

One can give MB either k or Δ as a parameter. Then MB sets the other parameter based on the input arc lengths. We refer to the code with the number of levels k set to two by MB2L, and to the code with Δ set to two by MB2D. These are the two extreme cases that we study. (We do not study the single-level case because it often would have needed too much memory and time.) Alternatively, one can let MB choose the values of both k and Δ based on the input. We refer to this adaptive variant as MB-A. The adaptive variant of the algorithm uses the relationship $\Delta = \Theta(k)$ suggested by the worst-case analysis. To choose the constant hidden by the Θ location, we observe the following. Examining empty buckets involves looking at a single pointer and has good locality properties as we access the buckets sequentially. Moving vertices to lower levels, on the other hand, requires changing several pointers, and has poor locality. This suggests that Δ should be substantially greater than k, and experiments confirm this.

In more detail, MB-A sets k and Δ as follows. First we find the smallest value of k such that k is a power of two and $(16k)^k \geq U/w$. Then we set Δ to $16k$. At this point, however, both Δ and k may be larger than they need to be. While $(\Delta/2)^k \geq U/w$ we reduce Δ. Finally, while $(\Delta)^{k-1} \geq U/w$ we reduce k. This typically leads to $16k \leq \Delta \leq 128k$ and works well in our tests.

We obtain our SQ code by adding the caliber heuristic to MB. The modification of MB is relatively straightforward. We use a stack to implement the set F needed by the caliber heuristic. The adaptive variant of the code, SQ-A, uses the same procedure to set k and Δ as MB-A does.

8. Experimental methodology and setup. Following Moret and Shapiro [32], we use a baseline code—breadth-first search (BFS) in our case—and measure running times of our shortest path codes on an input relative to the BFS running time on this input. Our BFS code computes distances and a shortest path tree for the unit length function. The breadth-first search problem is a special case of NSP and, modulo low-level details, the BFS running time is a lower bound on the NSP codes. Baseline running times give a good indication of how close to optimal the running times are and reduces dependency on low-level implementation and architecture details.

However, some of the dependencies, in particular, cache dependencies, remain. Our codes put arc and vertex records in consecutive locations. Input IDs of the vertices determine their ordering in memory. In general, breadth-first search examines vertices in a different order than an NSP algorithm. This may—and in some cases does—lead to very different caching behavior of the two codes for certain vertex orderings. To deal with this dependency on input IDs, our generators permute the IDs at random. Thus all our problem generators are randomized.

For every input problem type and any set of parameter values, we run the corresponding generator five times and report the averages. We report the baseline BFS time in seconds and all other times in units of the BFS time. In addition, we count operations that determine Φ_1 and Φ_2 in Lemma 5.2. For each of these operations, we give the number of the operations divided by the number of vertices so that the amortized operation cost is immediate. The two kinds of operations we count are examinations of empty buckets and the number of vertices processed during bucket expansion operations.

We use 64-bit integers for internal representation of arc lengths and distances. If the graph contains simple paths longer than 2^{64}, our codes may get overflows. Note that for 32-bit input arc lengths, no overflow can happen unless the number of vertices exceeds 2^{32}, which is too many to fit into the memory of a modern computer.

Our experiments have been conducted on a 933 MHz Pentium III machine with 512M of memory, 256K cache, and running RedHat Linux 7.1. All our shortest path codes and the baseline code are written in C++, in the same style, and compiled with the `gcc` compiler using the `-O6` optimization option. Our BFS code uses the same data structures as the MB code.

9. Problem families. We report data on seven problem families produced by three problem generators. These problem families have been selected as the most interesting from many more families we experimented with. Since we are interested in the efficiency of the data structures, we restrict our study to sparse graphs, for which the data structure manipulation time is most apparent.

Our first generator, SPRAND, builds a Hamiltonian cycle and then adds arcs at random. The generator may produce parallel arcs but not self-loops. Arc lengths are chosen independently and uniformly from $[\ell, u]$. Vertex 1 is the source. If the number of arcs is large enough, SPRAND graphs are expanders and the average number of vertices in the priority queue during a shortest path computation is large.

We use SPRAND to generate two problem families, RAND-I and RAND-C. For both families, $\ell = 1$ and $m = 4n$. For RAND-I, $u = n$, and n increases by a factor of two from one set of parameter values to the next one. We chose the initial value of n large enough so that the running time is nonnegligable and the final value is as large as possible subject to the constraint that all our codes run without paging. For RAND-C, $n = 2^{20}$ and $u = 2^i$; i starts at 1 and then takes on integer multiples of four from 4 to 32. Up to $i = 20$, the minimal arc length δ in all test inputs is one. For $i = 24$, δ is greater than one for some inputs. For $i = 28$ and 32, δ is always greater than one. Note that the expected value of C does not change for $i \geq 28$, and therefore the results for $i > 32$ would have been very similar to those for $i = 28$ and 32.

Our second generator, SPGRID, produces grid-like graphs. An x, y grid graph contains $x \cdot y$ vertices, $[i, j]$, for $0 \leq i < x$ and $0 \leq j < y$. A vertex $[i, j]$ is connected to the adjacent vertices in the same layer, $[i, j + 1 \bmod y]$ and $[i, j - 1 \bmod y]$. In addition, for $i < x-1$, each vertex $[i, j]$ is connected to the vertex $[i+1, j]$. Arc lengths are chosen independently and uniformly from $[\ell, u]$. Vertex $[0, 0]$ is the source. We use SPGRID to generate two problem families, LONG-I and LONG-C. Both families contain *long* grid graphs with $y = 8$ and x a parameter. For these graphs, the average number of vertices in the priority queue is small.

The LONG-I and LONG-C problem families are similar to the RAND-I and RAND-C families. For LONG-I, $u = n$, and n increases by a factor of two from the value that yields a reasonable running time to the maximum value that does not cause paging. The LONG-C problem family uses the same values of u as the RAND-C problem family.

Our last problem generator is SPHARD. This generator produces problems aimed to be hard for MLB algorithms for certain values of k and Δ. Graphs produced by this generator consist of $2k + 1$ vertex-disjoint paths, with the source connecting to the beginning of each path. (See Figure 2 for an example.) These paths have the same

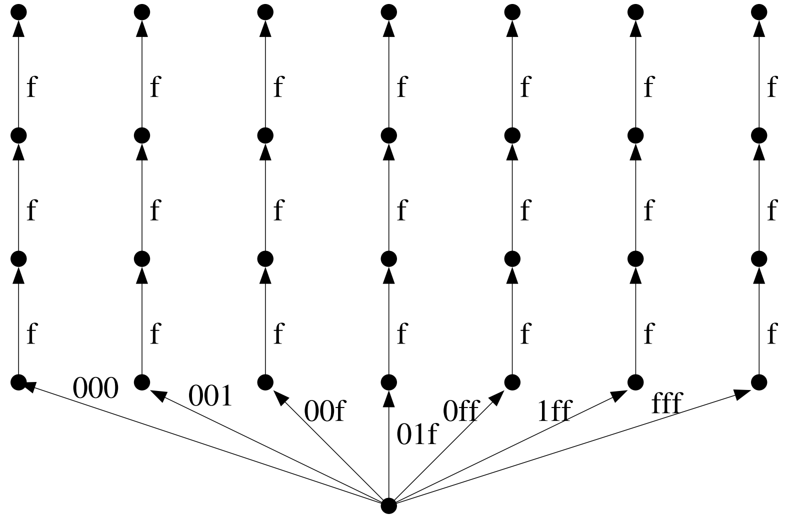


FIG. 2. *An example of a hard problem instance; $k = 3$ and $\Delta = 16$. Arc lengths are given in hexadecimal. We omit the extra vertex with arcs designed to manipulate vertex calibers.*

number of arcs, which can be adjusted to get a graph of the desired size. Path arcs have a length of Δ. The lengths of the source arcs are as follows. One arc has zero length. Out of the remaining arcs, k arcs have the following base-Δ representation. For $1 \leq i \leq k$, the first i digits are $\Delta - 1$ and the remaining digits are 0. The last k arcs, for $1 \leq j \leq k$, have the first $j-1$ digits $\Delta - 1$, the jth digit 1, and the remaining digits 0. The graph also contains an extra vertex with no incoming arcs connected to every other vertex of the graph. The length of the arc connecting the vertex to the source is zero to make sure that the minimum arc length is zero. The lengths of the other arcs are all the same. These lengths can be zero (to force every vertex caliber to zero) or large (so that the calibers are determined by the other arcs).

Note that if the SPHARD generator with parameters k and Δ produces an input, our adaptive codes may select different parameter values. For $D = \log \Delta$, a problem produced by SPRAND has $(k \cdot D)$-bit lengths. These lengths determine parameter values selected by the adaptive codes.

The three SPHARD problem families we study are HARD1, HARD0, and HARDEST-SQ. The first two problem families differ only in the length of the arcs which determine vertex calibers: the length is large for the first family and zero for the second. All problems in this family have approximately 2^{20} vertices, and the number of arcs is approximately the same in all problems. To create a problem in this family, we choose k and D such that $k \cdot D = 36$ and generate a problem which is hard for MB with k levels and $\Delta = 2^D$. Each HARDEST-SQ problem also has approximately 2^{20} vertices. Problems in this family differ by the k and Δ values. These values are selected so that both the generator and the adaptive codes use the same k and Δ parameters.

10. Experimental results.

10. Experimental results. This section discusses our experimental results.

Caliber heuristic effectiveness. Our analysis shows that work of the caliber heuristic is amortized over other work performed by the algorithm and therefore the heuristic cannot hurt the performance by much. The heuristic can, however, significantly improve performance. Experimental data confirms this fact. In particular, data for the HARD1 family in Table 1 shows how drastic performance improvement can be. The HARD0 family data in Table 1 shows that if all vertex calibers are forced to zero and the caliber heuristic never helps, its cost is just a few percent of the running time.

Making more levels practical. Our previous work [5, 6] showed that 2- and 3-level MLB implementations and their variants perform well except on certain types of graphs with very large lengths. Increasing the number of levels improved performance on bad examples but hurt performance on "typical" problems somewhat. Comparing MB and SQ code performance on graphs with random arc weights (Tables 2 and 3), we observe that, as the theory would suggest, the caliber heuristic helps more if the number of bucket levels is higher. This is especially apparent if one compares data for MB2D and SQ2D on RAND-C problems (Table 2). This makes the bucket structures with the higher number of levels, in particular, the adaptively selected number of levels, practical. While the random arc length data illustrates how the caliber heuristic helps in "typical" cases, Table 1 shows the effectiveness of adaptive parameter selection on hard problems: Note that for problems with 36-bit lengths, our adaptive codes set $k = 6$.

Operation counts and code tuning. As the analysis suggests, poor performance of the MLB codes is caused by either a large number of the empty bucket examinations or a high cost of bucket expansion operations. See, for example, Tables 3 and 1. The data shows that if the number of empty bucket examinations per vertex is moderate (e.g., ten), they are well amortized by other operations on vertices and do not have a noticeable effect on the running time. When the number of these operations reaches one hundred per vertex, they do have an effect. See, e.g., Table 2. Table 1 shows that processing vertices during bucket expansion is more expensive. Processing one vertex

TABLE 1
HARD1 (left) and HARD0 (right) family data.

k		BFS	MB	SQ
2	time	0.63	1189.20	1.38
	emp.	sec.	52428.94	0.40
	exp.		0.60	0.30
3	time	0.63	10.66	1.39
	emp.	sec.	1170.14	0.15
	exp.		1.14	0.57
4	time	0.62	2.68	1.44
	emp.	sec.	170.44	0.11
	exp.		1.56	0.67
6	time	0.63	2.25	1.53
	emp.	sec.	24.31	0.08
	exp.		2.46	0.77
9	time	0.62	2.87	1.60
	emp.	sec.	6.37	0.05
	exp.		3.89	0.85
12	time	0.63	3.70	1.66
	emp.	sec.	3.12	0.04
	exp.		5.36	0.88
18	time	0.63	5.86	1.82
	emp.	sec.	1.41	0.03
	exp.		8.32	0.93
36	time	0.62	16.32	2.32
	emp.	sec.	0.49	0.01
	exp.		17.75	0.97

k		BFS	MB	SQ
2	time	0.62	1199.67	1201.13
	emp.	sec.	52428.94	52428.94
	exp.		0.60	0.60
3	time	0.62	9.48	9.39
	emp.	sec.	1170.14	1170.14
	exp.		1.14	1.14
4	time	0.63	2.67	2.82
	emp.	sec.	170.44	170.44
	exp.		1.56	1.56
6	time	0.62	2.25	2.45
	emp.	sec.	24.31	24.31
	exp.		2.46	2.46
9	time	0.62	2.87	3.08
	emp.	sec.	6.37	6.37
	exp.		3.89	3.89
12	time	0.62	3.72	3.93
	emp.	sec.	3.12	3.12
	exp.		5.36	5.36
18	time	0.63	5.86	6.05
	emp.	sec.	1.41	1.41
	exp.		8.32	8.32
36	time	0.63	16.24	16.43
	emp.	sec.	0.49	0.49
	exp.		17.75	17.75

Table 2
RAND-C (top) and RAND-I (bottom) family data.

u		BFS	MB2L	SQ2L	MB2D	SQ2D	MB-A	SQ-A
1	time	2.97	1.39	1.35	1.39	1.36	1.38	1.35
	emp.	sec.	0.00	0.00	0.00	0.00	0.00	0.00
	exp.		0.48	0.48	0.48	0.48	0.48	0.48
4	time	2.99	1.74	1.59	1.99	1.73	1.47	1.42
	emp.	sec.	0.00	0.00	0.00	0.00	0.00	0.00
	exp.		1.04	0.88	1.64	1.15	0.42	0.39
8	time	3.03	1.81	1.69	2.80	1.96	1.79	1.70
	emp.	sec.	0.00	0.00	0.00	0.00	0.00	0.00
	exp.		1.26	1.05	3.51	1.49	1.26	1.05
12	time	3.00	1.86	1.73	3.41	2.07	1.85	1.74
	emp.	sec.	0.01	0.01	0.01	0.00	0.01	0.01
	exp.		1.35	1.13	5.49	1.56	1.35	1.13
16	time	3.00	1.84	1.74	3.91	2.16	1.95	1.71
	emp.	sec.	0.14	0.07	0.08	0.00	0.12	0.04
	exp.		1.37	1.16	7.44	1.56	2.04	1.08
20	time	2.99	1.82	1.75	4.52	2.27	1.89	1.71
	emp.	sec.	1.84	0.56	0.57	0.00	1.35	0.02
	exp.		1.37	1.16	8.97	1.56	2.11	1.04
24	time	2.99	2.08	1.80	4.95	2.30	2.11	1.76
	emp.	sec.	18.43	1.32	0.93	0.00	6.12	0.03
	exp.		1.33	1.13	9.23	1.56	2.99	1.12
28	time	2.92	2.59	1.85	5.22	2.37	2.13	1.75
	emp.	sec.	55.33	1.35	0.95	0.00	11.72	0.04
	exp.		1.33	1.13	9.23	1.56	2.79	1.06
32	time	2.98	2.49	1.85	5.11	2.36	2.13	1.74
	emp.	sec.	55.33	1.36	0.95	0.00	11.72	0.04
	exp.		1.33	1.13	9.23	1.56	2.79	1.06

n		BFS	MB2L	SQ2L	MB2D	SQ2D	MB-A	SQ-A
2^{17}	time	0.15	1.55	1.56	3.86	2.26	1.88	1.63
	emp./n	sec.	3.01	1.09	0.74	0.01	2.05	0.06
	exp./n		1.09	1.04	7.14	1.56	2.00	1.05
2^{18}	time	0.30	1.73	1.64	4.15	2.26	1.93	1.73
	emp./n	sec.	3.03	0.79	0.74	0.01	2.04	0.04
	exp./n		1.45	1.19	7.65	1.56	2.36	1.19
2^{19}	time	0.62	1.68	1.63	4.41	2.31	1.89	1.71
	emp./n	sec.	3.34	1.12	0.74	0.00	2.36	0.05
	exp./n		1.06	1.01	8.15	1.58	1.96	1.01
2^{20}	time	1.30	1.83	1.79	4.64	2.35	1.94	1.79
	emp./n	sec.	3.34	0.79	0.74	0.00	2.37	0.03
	exp./n		1.41	1.20	8.65	1.58	2.09	1.06
2^{21}	time	2.90	1.83	1.77	4.73	2.29	2.00	1.78
	emp./n	sec.	3.67	1.13	0.74	0.00	2.36	0.03
	exp./n		1.16	1.03	9.13	1.56	2.33	1.16

influences the running time roughly as much as scanning a hundred empty buckets. These observations justify the choice of k and Δ in our adaptive algorithms.

Most robust code. Our data also suggests that SQ-A is a very robust code. Often it is the fastest code, and its running time is always within 10% the fastest code. When designing the HARDEST-SQ problem family, our goal was to produce problems which are hard for the SQ-A code. If one believes that these problems are close to the worst case, then Table 4 shows that even for large lengths, SQ-A performs very well. For example, for 49-bit lengths, its running time exceeds that of BFS by less than a factor

Table 3

LONG-C (top) and LONG-I (bottom) family data.

u		BFS	MB2L	SQ2L	MB2D	SQ2D	MB-A	SQ-A
1	time	1.62	1.35	1.35	1.34	1.34	1.34	1.34
	emp.	sec.	0.13	0.06	0.13	0.06	0.13	0.06
	exp.		0.44	0.44	0.44	0.44	0.44	0.44
4	time	1.63	1.54	1.54	1.72	1.60	1.42	1.49
	emp.	sec.	0.61	0.34	0.55	0.23	0.69	0.66
	exp.		0.88	0.71	1.29	0.80	0.39	0.38
8	time	1.63	1.60	1.56	2.06	1.76	1.60	1.56
	emp.	sec.	2.64	0.77	0.96	0.29	2.64	0.77
	exp.		0.77	0.54	1.51	0.83	0.77	0.54
12	time	1.63	1.56	1.55	2.28	1.87	1.56	1.55
	emp.	sec.	5.82	2.44	1.00	0.29	5.82	2.44
	exp.		0.52	0.42	1.51	0.83	0.52	0.42
16	time	1.63	1.56	1.57	2.52	1.99	1.61	1.54
	emp.	sec.	13.68	9.45	1.00	0.29	8.87	1.40
	exp.		0.42	0.38	1.51	0.83	0.52	0.34
20	time	1.63	1.67	1.69	2.75	2.10	1.57	1.54
	emp.	sec.	43.27	37.60	1.00	0.29	9.30	2.64
	exp.		0.39	0.37	1.51	0.83	0.34	0.25
24	time	1.62	2.22	2.15	2.99	2.23	1.64	1.61
	emp.	sec.	146.96	136.51	1.00	0.30	6.10	2.30
	exp.		0.35	0.33	1.51	0.84	0.51	0.40
28	time	1.62	3.08	3.02	3.05	2.33	1.65	1.64
	emp.	sec.	212.05	205.08	1.00	0.35	10.96	2.72
	exp.		0.31	0.30	1.51	0.91	0.43	0.38
32	time	1.63	3.18	3.08	3.06	2.38	1.66	1.67
	emp.	sec.	212.00	206.77	1.00	0.36	10.96	2.91
	exp.		0.31	0.31	1.51	0.92	0.43	0.40

n		BFS	MB2L	SQ2L	MB2D	SQ2D	MB-A	SQ-A
2^{17}	time	0.08	1.71	1.71	2.71	2.14	1.86	1.71
	emp./n	sec.	8.59	4.77	1.00	0.29	4.88	1.00
	exp./n		0.46	0.39	1.51	0.83	0.61	0.41
2^{18}	time	0.17	1.66	1.61	2.80	2.13	1.81	1.68
	emp./n	sec.	12.45	5.10	1.00	0.29	4.17	1.31
	exp./n		0.27	0.22	1.51	0.83	0.63	0.46
2^{19}	time	0.35	1.65	1.65	2.74	2.17	1.73	1.61
	emp./n	sec.	13.65	9.43	1.00	0.29	8.89	1.40
	exp./n		0.42	0.38	1.51	0.83	0.52	0.34
2^{20}	time	0.75	1.59	1.60	2.72	2.10	1.64	1.60
	emp./n	sec.	17.89	10.09	1.00	0.29	6.98	1.47
	exp./n		0.23	0.21	1.51	0.83	0.45	0.31
2^{21}	time	1.61	1.60	1.63	2.65	2.06	1.62	1.59
	emp./n	sec.	23.59	18.84	1.00	0.29	5.88	2.44
	exp./n		0.40	0.37	1.51	0.83	0.52	0.42

of three. We estimate that for 32-bit lengths, SQ-A running time is always within a factor of 2.5 of the BFS time.

11. Concluding remarks. The worst-case bound for the smart queue algorithm is achieved for $\Delta = \theta(\frac{\log C}{\log\log C})$, when the work of moving vertices to lower levels balances the work of scanning empty buckets during bucket expansion. Our average-case analysis reduces the former but not the latter. We get a linear running time when Δ is constant and the empty bucket scans can be charged to vertices in nonempty

Table 4
HARDEST-SQ family data.

bits	$\log\Delta$	k		BFS	SQ-A
4	4	1	time	0.62	1.37
			emp.	sec.	0.33
			exp.		0.04
6	3	2	time	0.63	1.50
			emp.	sec.	1.60
			exp.		0.80
8	4	2	time	0.62	1.51
			emp.	sec.	3.20
			exp.		0.80
15	5	3	time	0.62	1.73
			emp.	sec.	9.00
			exp.		1.14
18	6	3	time	0.62	1.78
			emp.	sec.	18.14
			exp.		1.14
24	6	4	time	0.62	1.98
			emp.	sec.	21.11
			exp.		1.56
30	6	5	time	0.62	2.18
			emp.	sec.	23.00
			exp.		2.00
35	7	5	time	0.62	2.32
			emp.	sec.	46.27
			exp.		2.00
42	7	6	time	0.62	2.55
			emp.	sec.	48.92
			exp.		2.46
49	7	7	time	0.62	2.80
			emp.	sec.	50.87
			exp.		2.93

buckets. An interesting open question is if one can get a linear average running time and a better worst-case running time, for example, using techniques from [2, 6, 9], without running several algorithms "in parallel."

Our optimization is to detect vertices with exact distance labels before these vertices reach the bottom level of buckets and place them into F. This technique can be used not only in the context of multilevel buckets but also in the context of radix heaps [2] and hot queues [6].

The fact that SQ-A performance is close to that of BFS limits potential improvements one would consider. For example, a search of a graph to determine better parameter values would not pay for itself, unless it can be amortized over many shortest path computations, e.g., in the context of the all-pairs shortest path problem.

We would like to note that on the machine used in the experiments, fetching a value from the main memory takes on the order of a hundred processor clock cycles. Our experimental results imply that data structure manipulation times are comparable to the data fetch times. As the gap between the processor and memory speeds widens every year, the relative time spent on data structure operation will decrease, and our shortest path algorithm performance will be getting even closer to that of BFS.

An alternative to comparing a shortest path algorithm to BFS is as follows. Run the algorithm and save the sequence of vertices it scanned. Then rerun the algorithm using the saved sequence instead of the data structures to select the next vertex to scan. The difference between running times of the latter and the former algorithms is a measure of the data structure overhead. It can be compared to the running time of the original algorithm.

Informal experiments show that for problems that are easy for the label-correcting algorithms, the smart queue algorithm works almost as well. It would be interesting to have a more formal comparison of these implementations on real-life problems, such as those in [46]. We also implemented an algorithm that combines the ideas behind smart queues and hot queues [6]. Informal experiments show that the resulting code performs a little better on the hard instances but slightly worse on "typical" instances.

Our results suggest that the smart queue algorithm should be considered in practice when arc lengths are nonnegative integers. The shortest path codes and generators used in this study are available via http://www.avglab.com/andrew/soft.html.

Acknowledgments. Part of this work was done at STAR Lab., InterTrust Technologies Corp., 4750 Patrick Henry Dr., Santa Clara, CA 95054. The author would like to thank Jim Horning, Rajeev Raman, Bob Tarjan, and Eva Tardos for useful discussion and comments on a draft of this paper. We are also grateful to an anonymous referee of a conference version of the paper [22] for pointing out that Theorem 6.2 does not need arc lengths to be independent.

REFERENCES

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*. Addison–Wesley, Reading, MA, 1974.

[2] R. K. Ahuja, K. Mehlhorn, J. B. Orlin, and R. E. Tarjan, *Faster algorithms for the shortest path problem*, J. ACM, 37 (1990), pp. 213–223.

[3] A. Brodnik, S. Carlsson, J. Karlsson, and J. I. Munro, *Worst case constant time priority queues*, in Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, SIAM, Philadelphia, 2001, pp. 523–528.

[4] B. V. Cherkassky and A. V. Goldberg, *Negative-cycle detection algorithms*, Math. Program., 85 (1999), pp. 277–311.

[5] B. V. Cherkassky, A. V. Goldberg, and T. Radzik, *Shortest paths algorithms: Theory and experimental evaluation*, Math. Programming, 73 (1996), pp. 129–174.

[6] B. V. Cherkassky, A. V. Goldberg, and C. Silverstein, *Buckets, heaps, lists, and monotone priority queues*, SIAM J. Comput., 28 (1999), pp. 1326–1346.

[7] H. Chernoff, *A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations*, Ann. Math. Statistics, 23 (1952), pp. 493–507.

[8] R. Cole and U. Vishkin, *Deterministic coin tossing with applications to optimal parallel list ranking*, Inform. and Control, 70 (1986), pp. 32–53.

[9] E. V. Denardo and B. L. Fox, *Shortest–route methods:* 1. *Reaching, pruning, and buckets*, Oper. Res., 27 (1979), pp. 161–186.

[10] R. B. Dial, *Algorithm* 360*: Shortest path forest with topological ordering*, Comm. ACM, 12 (1969), pp. 632–633.

[11] R. B. Dial, F. Glover, D. Karney, and D. Klingman, *A computational analysis of alternative algorithms and labeling techniques for finding shortest path trees*, Networks, 9 (1979), pp. 215–248.

[12] E. W. Dijkstra, *A note on two problems in connection with graphs*, Numer. Math., 1 (1959), pp. 269–271.

[13] E. A. Dinic, *Economical algorithms for finding shortest paths in a network*, in Transportation Modeling Systems, Yu. S. Popkov and B. L. Shmulyian, eds., Institute for System Studies, Moscow, 1978, pp. 36–44 (in Russian).

[14] L. R. Ford, Jr., *Network Flow Theory*, Technical report P-932, Rand Corporation, Santa Monica, CA, 1956.

[15] L. R. Ford, Jr., and D. R. Fulkerson, *Flows in Networks*. Princeton University Press, Princeton, NJ, 1962.

[16] M. L. Fredman and R. E. Tarjan, *Fibonacci heaps and their uses in improved network optimization algorithms*, J. ACM, 34 (1987), pp. 596–615.

[17] M. L. Fredman and D. E. Willard, *Trans-dichotomous algorithms for minimum spanning trees and shortest paths*, J. Comput. System Sci., 48 (1994), pp. 533–551.

[18] H. N. Gabow, *Scaling algorithms for network problems*, J. Comput. System Sci., 31 (1985), pp. 148–168.

[19] G. Gallo and S. Pallottino, *Shortest paths algorithms*, Ann. Oper. Res., 13 (1988), pp. 3–79.

[20] F. Glover, R. Glover, and D. Klingman, *Computational study of an improved shortest path algorithm*, Networks, 14 (1984), pp. 25–37.

[21] A. V. Goldberg, *A Simple Shortest Path Algorithm with Linear Average Time*, Technical report STAR-TR-01-03, STAR Lab., InterTrust Tech., Inc., Santa Clara, CA, 2001.

[22] A. V. Goldberg, *A simple shortest path algorithm with linear average time*, in Proceedings of the 9th ESA, Lecture Notes in Comput. Sci. 2161, Springer-Verlag, Berlin, 2001, pp. 230–241.

[23] A. V. Goldberg, *Shortest path algorithms: Engineering aspects*, in Proceedings of the ISAAC '01, Lecture Notes in Comput. Sci., Springer-Verlag, Berlin, 2001, pp. 502–513.

[24] A. V. Goldberg and C. Silverstein, *Implementations of Dijkstra's algorithm based on multi-level buckets*, in Lecture Notes in Econom. and Math. Systems 450 (Refereed Proceedings), P. M. Pardalos, D. W. Hearn, and W. W. Hages, eds., Springer-Verlag, Berlin, 1997, pp. 292–327.

[25] T. Hagerup, *Improved shortest paths in the word RAM*, in 27th International Colloq. on Automata, Languages and Programming, Geneva, Switzerland, 2000, pp. 61–72.

[26] T. Hagerup, *Simpler computation of single-source shortest paths in linear average time*, in Proceedings of STACS 2004, ACM, New York, 2004, pp. 362–369.

[27] M. S. Hung and J. J. Divoky, *A computational study of efficient shortest path algorithms*, Comput. Oper. Res., 15 (1988), pp. 567–576.

[28] H. Imai and M. Iri, *Practical efficiencies of existing shortest-path algorithms and a new bucket algorithm*, J. Oper. Res. Soc. Japan, 27 (1984), pp. 43–58.

[29] D. B. Johnson, *Efficient algorithms for shortest paths in sparse networks*, J. ACM, 24 (1977), pp. 1–13.

[30] U. Meyer, *Single-source shortest paths on arbitrary directed graphs in linear average time*, in Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, SIAM, Philadelphia, 2001, pp. 797–806.

[31] J.-F. Mondou, T. G. Crainic, and S. Nguyen, *Shortest path algorithms: A computational study with the C progremming language*, Comput. Oper. Res., 18 (1991), pp. 767–786.

[32] B. M. E. Moret and H. D. Shapiro, *An empirical analysis of algorithms for constructing a minimum spanning tree*, in Proceedings of the 2nd Workshop on Algorithms and Data Structures, Springer-Verlag, Berlin, 1991, pp. 99–117.

[33] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, Cambridge, UK, 1995.

[34] K. Noshita, *A theorem on the expected complexity of Dijkstra's shortest path algorithm*, J. Algorithms, 6 (1985), pp. 400–408.

[35] S. Pallottino, *Shortest-path methods: Complexity, interrelations and new propositions*, Networks, 14 (1984), pp. 257–267.

[36] U. Pape, *Implementation and efficiency of Moore-algorithms for the shortest route problem*, Math. Programming, 7 (1974), pp. 212–222.

[37] R. Raman, *Fast Algorithms for Shortest Paths and Sorting*, Technical report TR 96-06, King's Colledge, London, 1996.

[38] R. Raman, *Priority queues: Small, monotone and trans-dichotomous*, in Proceedings of the 4th Annual European Symposium on Algorithms, Lecture Notes in Comput. Sci. 1136, Springer-Verlag, Berlin, 1996, pp. 121–137.

[39] R. Raman, *Recent results on single-source shortest paths problem*, SIGACT News, 28 (1997), pp. 81–87.

[40] J. T. Stasko and J. S. Vitter, *Pairing heaps: Experiments and analysis*, Comm. ACM, 30 (1987), pp. 234–249.

[41] R. E. Tarjan, *Data Structures and Network Algorithms*, CBMS-NSF Regional Conf. Ser. in Appl. Math. 44, SIAM, Philadelphia, 1983.

[42] M. Thorup, *Undirected single-source shortest paths with positive integer weights in linear time*, J. ACM, 46 (1999), pp. 362–394.

[43] M. Thorup, *On RAM priority queues*, SIAM J. Comput., 30 (2000), pp. 86–109.

[44] R. A. Wagner, *A shortest path algorithm for edge-sparse graphs*, J. ACM, 23 (1976), pp. 50–57.

[45] J. W. J. Williams, *Algorithm* 232 *(Heapsort)*, Comm. ACM, 7 (1964), pp. 347–348.

[46] F. B. Zhan and C. E. Noon, *Shortest path algorithms: An evaluation using real road networks*, Transp. Sci., 32 (1998), pp. 65–73.

SIAM J. COMPUT.
Vol. 37, No. 5, pp. 1656–1673

STABILITY OF LOAD BALANCING ALGORITHMS IN DYNAMIC ADVERSARIAL SYSTEMS[*]

ELLIOT ANSHELEVICH[†], DAVID KEMPE[‡], AND JON KLEINBERG[§]

Abstract. In the dynamic load balancing problem, we seek to keep the job load roughly evenly distributed among the processors of a given network. The arrival and departure of jobs is modeled by an adversary restricted in its power. Muthukrishnan and Rajaraman [*An adversarial model for distributed dynamic load balancing*, in Proceedings of the 10th ACM Symposium on Parallel Algorithms and Architectures, ACM, New York, 1998] gave a clean characterization of a restriction on the adversary that can be considered the natural analogue of a cut condition. They proved that a simple local balancing algorithm proposed by Aiello et al. [*Approximate load balancing on dynamic and asynchronous networks*, in Proceedings of the 25th ACM Symposium on Theory of Computing, ACM, New York, 1993] is stable against such an adversary if the insertion rate is restricted to a $(1 - \varepsilon)$ fraction of the cut size. They left as an open question whether the algorithm is stable at rate 1. In this paper, we resolve this question positively, by proving stability of the local algorithm at rate 1. Our proof techniques are very different from the ones used by Muthukrishnan and Rajaraman and yield a simpler proof and tighter bounds on the difference in loads. In addition, we introduce a multicommodity version of this load balancing model and show how to extend the result to the case of balancing two different kinds of loads at once (obtaining as a corollary a new proof of the 2-commodity Max-Flow Min-Cut Theorem). We also show how to apply the proof techniques to the problem of routing packets in adversarial systems. Awerbuch et al. [*Simple routing strategies for adversarial systems*, in Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 2001] showed that the same load balancing algorithm is stable against an adversary, inserting packets at rate 1 with a single destination in dynamically changing networks. Our techniques give a much simpler proof for a different model of adversarially changing networks.

Key words. adversarial load balancing, packet routing, multicommodity flow

AMS subject classifications. 68W15, 68W40

DOI. 10.1137/050639272

1. Introduction.

Load balancing. In a distributed network of computing hosts, the performance of the system can depend crucially on dividing up work effectively across the participating nodes. This type of *load balancing problem* has been studied in many different models, centered around the idea that an algorithm should avoid creating "hot spots" that degrade system performance [26].

We consider a basic model of load balancing in a distributed network, which has formed the basis of a number of earlier studies [1, 18, 6, 23, 24]. A network of identical processors is represented by an undirected graph $G = (V, E)$. There are a number of

[*] Received by the editors August 31, 2005; accepted for publication (in revised form) August 14, 2007; published electronically February 14, 2008. A preliminary version of this paper appeared in *Proceedings of the* 34*th Annual ACM Symposium on Theory of Computing*, ACM, New York, 2002.
http://www.siam.org/journals/sicomp/37-5/63927.html

[†] Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180 (eanshel@cs.rpi.edu). This author's research was supported in part by an NSF graduate fellowship.

[‡] Department of Computer Science, University of Southern California, Los Angeles, CA 90089 (dkempe@usc.edu). This author's research was supported by an NSF graduate fellowship.

[§] Department of Computer Science, Cornell University, Ithaca, NY 14853 (kleinber@cs.cornell.edu). This author's research was supported in part by a David and Lucile Packard Foundation Fellowship, an ONR Young Investigator Award, and NSF Faculty Early Career Development Award CCR-9701399.

jobs to be processed in the system, abstractly represented by unit-size *tokens*. Time progresses in discrete steps called *rounds*; in a given round, each token is held by one of the nodes, which is viewed as processing the associated job, and the *load* on a node is defined to be the number of tokens it holds. The goal is to *balance* the loads, so that no single node has too many tokens; this can be accomplished by transmitting tokens between neighboring nodes of the graph, at a rate of one token per edge per round. We are particularly interested in *local* algorithms for this problem: rather than using a centralized approach to coordinate the movement of tokens, each node will simply compare its load to those of its neighbors, and decide whether to move a token across an edge based on this information.

This model is clearly very simple in a number of respects, particularly in the uniformity of the processors (nodes) and jobs (tokens), and the fact that any job can be executed on any processor. More subtly, it is not even clear in all settings that balancing the load evenly is the optimal strategy in a distributed network of processors (see, e.g., [13]). At the same time, however, the model cleanly captures the basic constraints imposed by an underlying interconnection topology in the process of distributing jobs through a network, as evidenced by the results of previous analysis [1, 18, 6, 23, 24]; the simplicity of the model allows one to reason very clearly about the effect of these constraints.

Early work on the model focused on the *static* version of the problem: each node is given a set of tokens initially, and nodes must exchange tokens as rapidly as possible so that they all end up with approximately the same number [1, 18, 6, 24]. However, load balancing is a natural setting in which to study algorithms designed to run indefinitely; jobs (tokens) may arrive and depart from the network, and at all times, the algorithm must maintain an approximately uniform load across nodes. This is a type of *stability* condition: no load should diverge arbitrarily from the average as time progresses. For a number of different models, such *dynamic* algorithms have been studied in a probabilistic framework, where one assumes an underlying randomized process that generates job arrivals and departures; see, e.g., [15, 22] and the references therein.

An adversarial model. Motivated by work in the related area of packet routing [8, 9, 12, 4], Muthukrishnan and Rajaraman proposed an *adversarial* framework for studying dynamic load balancing in the token-based model we have been discussing [23]. Rather than considering a probabilistic process that generates tokens, they posit an *adversary* that is allowed at the beginning of each round to introduce tokens at some nodes (corresponding to new jobs) and remove tokens from others (corresponding to jobs that have finished). Subsequently, an algorithm is allowed to move tokens across edges, as described above, so as to try to maintain balanced loads. This alternation of moves by the adversary and algorithm continues for an arbitrary number of rounds. Note that by allowing the adversary to control the removal of tokens as well as their arrival, one is modeling a worst-case assumption that jobs may have arbitrary duration, and the algorithm does not know how much processing time a job has remaining until the moment it ends.

If we let a_t denote the average number of tokens per node in the system at the beginning of round t and $h_t(v)$ denote the number of tokens at node v (the *height* of v) at round t, then the goal of a dynamic load balancing algorithm in this model is to keep $h_t(v)$ close to a_t for all nodes v and rounds t. Formally, we say that an algorithm is *stable* against a given adversary if there is a constant B such that $|h_t(v) - a_t| \leq B$ for all nodes v and rounds t. Note that stability in this context imposes a bound on deviation from the average; it is not required that the actual number of tokens in the system remain bounded.

As in the case of packet routing [12, 4], one needs to find a suitable restriction on the adversary: an arbitrarily powerful adversary could flood a particular set of nodes $S \subseteq V$ with tokens much faster than these nodes can spread the tokens out to the rest of the graph, and thereby prevent any algorithm from maintaining stability. This consideration motivates the following natural definition of an adversary [23] with *rate* r. For a set $S \subseteq V$, let $e(S)$ denote the set of edges with exactly one end in S, and $\delta_t(S)$ the net increase in tokens in set S due to the addition and removal of jobs in round t (note that $\delta_t(S)$ could be negative). If the heights of nodes in S were to change precisely according to the average, then the net change in tokens in S would be $|S| \cdot (a_{t+1} - a_t)$. One wants the difference between these two quantities to be "accounted for" by the edges in $e(S)$. We say that the adversary has *rate* r if for all $S \subseteq V$, one has

$$|\delta_t(S) - |S|(a_{t+1} - a_t)| \leq r \cdot |e(S)|. \tag{1.1}$$

For rate $r > 1$, there are adversaries against which no algorithm (whether online or offline) can be stable. Muthukrishnan and Rajaraman gave a local-control algorithm that is stable against all adversaries of rate r, for every $r < 1$. As an open question, they asked whether there exists a local-control algorithm that is stable against all adversaries of rate 1.

The present work. We begin by providing a local-control load balancing algorithm that is stable against every adversary of rate 1, thereby resolving the open question of Muthukrishnan and Rajaraman. In fact, we show that the following simple rule has this stability property for every value of the parameter θ:

> At any round t, if the number of tokens on node u exceeds the number
> of tokens on its neighbor v by at least θ, then u moves a token to v.

This type of algorithm was considered in earlier work on the static model by Aiello et al. [1] as well as in many subsequent papers. Setting $\theta = 2\Delta + 1$, where Δ is the maximum node degree in G, yields the specific algorithm studied by Muthukrishnan and Rajaraman.

Beyond simply showing the stability of local algorithms at the *critical rate* $r = 1$, our analysis is based on a new proof technique in which a potential function bound is maintained not only for the entire node set V but for every subset of V. Compared to [23], we obtain significantly improved bounds on the deviation from the average, and a simpler proof. Specifically, we show that the maximum possible deviation from the average is $O(\Delta n)$, where n is the number of nodes of G, and this is asymptotically optimal in the worst case; the analysis in [23] had established a bound of $O(\Delta^2 n^{2.5}(1 - r)^{-1})$ when $r < 1$. Our analysis also shows stability in a more general model where edges of G can appear and disappear over time.

Following this, we introduce a *multicommodity* version of this load balancing model. We consider a system in which there are k distinct types of jobs. The jobs of one given type induce the same load on each processor; but the different types of jobs place different resource requirements on the nodes, and so we require the load balancing condition to apply to each type separately. Formally, we have the same adversarial model as before with a network G and a collection of tokens; but now the tokens are partitioned into k *commodities* and the stability requirement must hold when the tokens of each commodity are considered separately. In a single round, at most one token in total can be sent across any one edge. (This is in keeping with the standard multicommodity notion that constraints at nodes must be satisfied by each commodity separately, while shared edge capacities must be respected by the commodities cumulatively.)

We show that the natural rate condition on adversaries—essentially obtained by summing (1.1) over the commodities—can be related in a precise sense to the cut condition for standard multicommodity network flow. As a result, applying well-known results on the cut condition [19, 20, 21], we find that for every $k > 2$, there is a k-commodity adversary of rate $r_k \leq 1$ against which no load balancing algorithm can be stable.

For $k = 2$, however, the cut condition does not pose an obstacle to having algorithms that are stable all the way up to rate 1. Indeed, we are able to generalize our first result to show that for 2-commodity load balancing, there is a simple local-control algorithm that is stable against every adversary of rate 1. We also use the relationship between adversaries and cut conditions to provide a new proof of Hu's Max-Flow Min-Cut Theorem for 2-commodity flow [19]. While our proof is not necessarily shorter than other proofs discovered subsequent to Hu's [21, 25], it is arguably more elementary: it does not require linear programming duality (as in [21]) or even the traditional Max-Flow Min-Cut Theorem for single-commodity flow (as in [25]).

Finally, we further develop the connection between dynamic load balancing and network flows by extending our analysis to packet routing in the adversarial model considered by Aiello et al. [2] and Gamarnik [17]. We give an adaptive routing algorithm that is stable against adversaries of rate 1 in the case where packets can be injected at multiple sources but are destined for a single sink; our algorithm is stable in a dynamic network model where edges can appear and disappear. A stable algorithm for this version of the problem was previously given in a recent paper of Awerbuch et al. [7], using a different, but essentially more general, notion of a dynamic network; our proof, a direct adaptation of the analysis of our single-commodity load balancing algorithm, is considerably shorter and simpler.

Recent progress. Since the original publication of this work, several papers have studied a model wherein nodes can exchange arbitrarily many jobs in one step, and each node with a nonempty load executes one job in each time step. This type of load balancing algorithm (often called the diffusion algorithm [14]) has been very well studied and is known to have many nice properties, although its behavior in the presence of an adversary was unknown until recently. In this model, the cut condition becomes trivial; instead, the adversary is allowed to insert at most n jobs in each time step, and stability is defined in terms of an upper bound on the load of all nodes.

Berenbrink, Friedetzky, and Goldberg [10] show that the work-stealing algorithm, in which only processors with empty queues request jobs from others, is stable against adversaries of rate strictly below 1. They assume that processors can request jobs from any other processor. Anagnostopoulos, Kirsch, and Upfal [3] show the same type of stability for a local protocol that makes nodes equalize load with their neighbors. Most recently, Berenbrink, Friedetzky, and Martin [11] proved stability against rate-1 adversaries for a protocol in which nodes exchange load with all their neighbors. Both the protocol and the analysis are very similar to those in the present work.

2. Single-commodity load balancing.

In this section, we will study the load balancing problem for a single commodity and, in particular, prove that the natural balancing algorithm is stable at rate 1, thus settling an open question from [23]. We first precisely define the model and the algorithm and introduce the necessary notation.

2.1. Model and algorithm. The network is represented by a connected undirected graph $G = (V, E)$ with $n = |V|$ nodes. In each round, the adversary first adds or removes tokens (this is called the *Adversary step*). Subsequently, in the *Redistribution step*, up to one token can be moved along each edge $e \in E$ by the algorithm (or up to c_e tokens in the case of networks with edge capacities).

The adversary is limited by the following *cut condition*: For a subset $S \subseteq V$ of nodes, let $\delta_t(S)$, a_t, and $e(S)$ be defined as above. Then, the insertion and removal of tokens by the adversary during round t has to satisfy

$$|\delta_t(S) - |S|(a_{t+1} - a_t)| \leq |e(S)|. \tag{2.1}$$

Nodes have queues associated with them, in which they store their tokens. The *height* $h_t(v)$ of a node v is the number of tokens in v's queue at the beginning of round t. The *imbalance* is $b_t(v) = h_t(v) - a_t$, i.e., the number of excess (or missing) tokens at node v with respect to the average over the entire network. $\overline{h}_t(v)$ and $\overline{b}_t(v)$ denote the same quantities after the Adversary step of round t.

It is the decision of the algorithm along which edges to send tokens. The goal of any balancing algorithm is to keep the imbalance bounded for all nodes. If an algorithm ensures that there is an absolute bound B such that $|b_t(v)| \leq B$ for all nodes v and all times t, we call the algorithm *stable*. We will show that the following very simple family of local-control algorithms is stable against every adversary respecting the cut condition. It has a *threshold parameter* $\theta \geq 1$, which determines how aggressively the algorithm balances.

```
Algorithm SCLB$_\theta$
At each time $t$, for each edge $e = (u, v)$:
    If $h_t(u) \geq h_t(v) + \theta$, then send a token from $u$ to $v$.
    If $h_t(v) \geq h_t(u) + \theta$, then send a token from $v$ to $u$.
```

This algorithm does not specify whether tokens are sent along an edge (u, v) when $|h_t(u) - h_t(v)| < \theta$. All of our subsequent statements will remain true independently of what the algorithm does in this case. (Notice that this algorithm requires only local information and can therefore be executed in a distributed fashion in a network.)

2.2. Stability of the algorithm. Our main theorem in this section is that the algorithm SCLB$_\theta$ is stable against any adversary respecting the cut condition of inequality (2.1). We allow for tokens to be in the system at time 1 and let $H := \max_{v \in V} h_1(v)$.

THEOREM 2.1. *For any adversary respecting the cut condition, and any $\theta \geq 1$, the algorithm* SCLB$_\theta$ *is stable; i.e., there is a constant B (depending on H, θ, and G) such that $|b_t(v)| \leq B$ for all nodes v at all times t.*

The intuition behind our proof is based on the (incorrect) observation that the algorithm seems to ensure that the height difference between adjacent nodes cannot grow beyond θ. Hence, the largest difference between the heights of any two nodes should be achieved when G is a simple path, and the two nodes are the endpoints of the path—having a height difference of about $n\theta$.

It is, however, easy to see that the height difference between two adjacent nodes can become more than θ, because the adversary can "rearrange" the heights within sets to a certain extent. Imagine, for instance, a long path with the adversary adding tokens to the first node of the path, until the first node has height $n\theta$, and the height of each successive node decreases by θ. Then it is possible to rearrange the tokens on the nodes previously having heights $n\theta$, $(n-1)\theta$, and $(n-2)\theta$ so that they each have $(n-1)\theta$ tokens, which would result in the height difference of 2θ between the third and fourth nodes of the path, since the height of the fourth node remains $(n-3)\theta$. The adversary can do this by adding two tokens to the node with height $(n-2)\theta$ and subtracting one from the node with height $n\theta$ for θ successive rounds. During each Redistribution step of this process, a token will move from the third to the fourth node, but these tokens will continue traveling downhill along the path, so that the height of the fourth node will never grow to be much larger than $(n-3)\theta$. This process obeys the cut condition, but afterward, there are two adjacent nodes with almost 2θ height difference. Fortunately, although there are now more nodes with large heights, the adversary had to pay for this rearrangement by making the highest queue smaller. In an amortized sense, the situation has not become worse.

These observations suggest maintaining height bounds for each subset of the nodes and showing that these bounds form an invariant. For convenience, we will write $b_t(S) = \sum_{v \in S} b_t(v)$ for any set $S \subseteq V$ of vertices (and similarly for other quantities like $h_t(S)$ and $\delta_t(S)$). With Δ denoting the maximum degree of any vertex, we write $\gamma = 2\Delta + \theta$. The key invariant is the following:

$$|b_t(S)| \leq \sum_{j=n-|S|+1}^{n} (H + \gamma \cdot j) \quad \text{for all } S \subseteq V. \tag{2.2}$$

Figure 2.1 pictorially illustrates the upper bound of this invariant as the sum of the column heights of the rightmost $|S|$ columns.

Below, we prove Lemma 2.2, showing that inequality (2.2) is indeed an invariant over time for the algorithm $\texttt{SCLB}_\theta$, against any adversary respecting the cut condition.

LEMMA 2.2. *If the adversary respects the cut condition and the invariant* (2.2) *holds at the beginning of round t, then it holds at the beginning of round $t+1$.*

Using this lemma, the proof of Theorem 2.1 is straightforward.

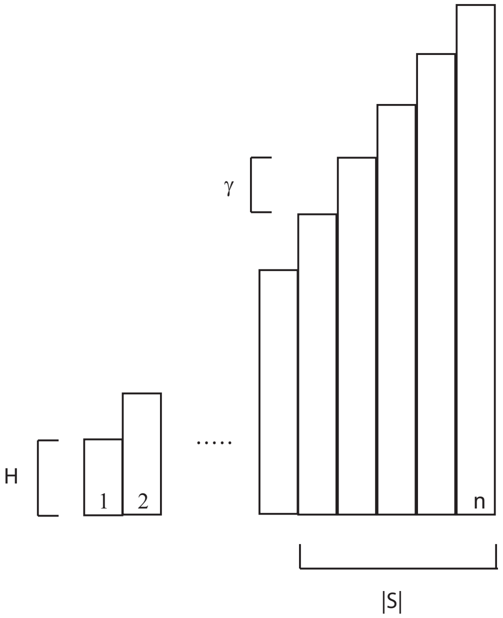


FIG. 2.1. *An illustration of invariant* (2.2).

Proof of Theorem 2.1. We prove by induction that (2.2) holds at every time t. At time 1, $h_1(v) \leq H$ for all v by definition, so

$$|b_1(S)| \quad \leq \quad \sum_{v \in S} H \quad \leq \quad \sum_{j=n-|S|+1}^{n} (H + \gamma \cdot j)$$

for all sets $S \subseteq V$. The induction step from t to $t+1$ follows from Lemma 2.2, and we can apply the resulting guarantee to the singleton sets $\{v\}$, yielding a bound of $B = H + \gamma \cdot n$. □

Proof of Lemma 2.2. The proof is by contradiction. Assume that the invariant (2.2) holds at the beginning of round t but not at the beginning of round $t+1$. Let S be a set maximizing

$$\Phi(S) := |b_{t+1}(S)| - \sum_{j=n-|S|+1}^{n} (H + \gamma \cdot j).$$

If several sets achieve the maximum value, let S have minimal size among all these sets. First, notice that the choice of S guarantees that either all $u \in S$ have positive $b_{t+1}(u)$, or they all have negative $b_{t+1}(u)$, and hence $|b_{t+1}(S)| = \sum_{u \in S} |b_{t+1}(u)|$.

Since (2.2) was assumed to hold at the beginning of round t and fails at the beginning of round $t+1$, we know that $|b_{t+1}(S)| > |b_t(S)|$. How can the values $h_t(u)$ for nodes $u \in S$ change?

Adversary step. Substituting the definitions of $\overline{b}$ and b, we obtain that for any set S,

$$\begin{aligned} |\overline{b}_t(S)| &= |\overline{h}_t(S) - |S| \cdot a_{t+1}| \\ &= |h_t(S) + \delta_t(S) - |S| \cdot a_{t+1}| \\ &= |b_t(S) + \delta_t(S) - |S| \cdot (a_{t+1} - a_t)| \\ &\leq |b_t(S)| + |\delta_t(S) - |S| \cdot (a_{t+1} - a_t)| \\ &\leq |b_t(S)| + |e(S)|. \end{aligned}$$

The two inequalities hold because of the triangle inequality and the cut condition on the adversary.

Redistribution step. Fix an edge $e = (u, v)$ with $u \in S$ and $v \notin S$. Because S maximizes Φ and has minimal size, u has imbalance $|b_{t+1}(u)| > H + \gamma \cdot (n - |S| + 1)$, since otherwise $\Phi(S - u) \geq \Phi(S)$. In particular, $|b_{t+1}(u)| > \gamma$.

Because v was not included in S, its imbalance $b_{t+1}(v)$ must either have sign opposite to the sign of $b_{t+1}(S)$ or have absolute value $|b_{t+1}(v)| \leq H + \gamma \cdot (n - |S|)$. In either case, $|b_{t+1}(u) - b_{t+1}(v)| > \gamma$, and simply substituting the definition of γ, we also obtain $|h_{t+1}(u) - h_{t+1}(v)| > 2\Delta + \theta$. During the Redistribution step of round t, at most Δ tokens can have moved to or from nodes u and v, so their heights can have changed by at most Δ each, and therefore their previous height difference is at least $|\overline{h}_t(u) - \overline{h}_t(v)| > \theta$. Figure 2.2 illustrates the gap of at least γ between the queue heights of nodes in S and outside S.

If $b_{t+1}(u) \geq 0$, then $\overline{h}_t(u) - \overline{h}_t(v) > \theta$, so the algorithm $\mathtt{SCLB}_\theta$ moves a token from u to v along e, and no tokens from v to u, thereby decreasing $\overline{b}_t(u)$ by 1. On the other hand, if $b_{t+1}(u) < 0$, then $\overline{h}_t(v) - \overline{h}_t(u) > \theta$, and a token must be moved from v to u along e, increasing the (negative) imbalance $\overline{b}_t(u)$ by 1. Because $|b_{t+1}(u)| > \gamma = \theta + 2\Delta$, and therefore $|\overline{b}_t(u)| > \theta + \Delta$, the sign of the imbalance does

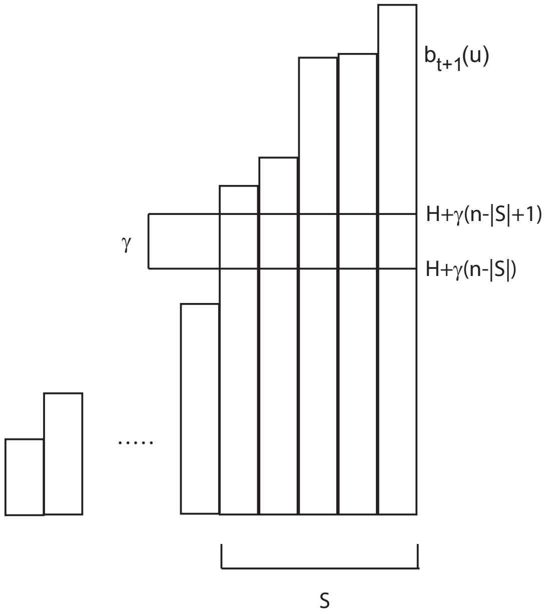


FIG. 2.2. *The gap of γ between the queue heights of nodes in S and outside S in the case that $b_{t+1}(S)$ is positive.*

not change during the Redistribution step, even if Δ tokens were moved to or from u, and hence, $|\bar{b}_t(u)|$ decreased by 1 as a result of edge e.

This holds for every edge $e \in e(S)$, and using the fact that the average a does not change during the Redistribution step, we obtain that

$$\begin{aligned} |b_{t+1}(S)| &= \sum_{u \in S} |b_{t+1}(u)| \\ &\leq \left(\sum_{u \in S} |\bar{b}_t(u)| \right) - |e(S)| \\ &= |\bar{b}_t(S)| - |e(S)|. \end{aligned}$$

Putting the arguments for the two steps together, we obtain that $|b_{t+1}(S)| \leq |\bar{b}_t(S)| - |e(S)| \leq |b_t(S)|$. This contradicts our assumption that $|b_{t+1}(S)| > |b_t(S)|$ and thus completes the proof. □

Notice that our bound $B = H + \gamma \cdot n$ is asymptotically tight. To see this, consider a simple path of length n. It is certainly legal for the adversary to insert one token at node n in every round, and never remove tokens. After about $\theta \cdot \frac{n^2}{2}$ rounds, each node k will contain about $k\theta$ tokens, and hence the imbalance of node n is about $\theta \cdot \frac{n}{2}$.

2.3. Capacities, dynamic networks, and time windows. The result of Theorem 2.1 can be easily extended to the case that the edges have (integer) capacities c_e associated with them, and up to c_e tokens can be sent along e in every round. We assume that whenever the algorithm decides to send tokens from u to v along e, it sends as many as possible, i.e., bounded only by the capacity c_e and the number of tokens at u. The cut condition now requires that the imbalance created by the adversary be restricted by the total capacity of the cut. Showing that the above algorithm is still stable in this capacitated version is easy given Theorem 2.1.

COROLLARY 2.3. *In the above capacitated scenario, for any adversary respecting the cut condition and any $\theta \geq 1$, the algorithm* $\texttt{SCLB}_\theta$ *is stable; i.e., there is a constant B (depending on H, θ, G, and the maximum edge capacity* $\max_e c_e$*) such that $|b_t(v)| \leq B$ for all nodes v at all times t.*

Proof. Replace each edge with capacity c_e by c_e parallel edges of capacity 1. By Theorem 2.1, the algorithm is stable on this new graph, and therefore, it is also stable on the capacitated graph. Notice that the constant γ and hence the bound B now depend on the maximum capacity, and so instead of $B = H + 2\Delta n + \theta n$, the new bound becomes $B = H + 2\Delta n \cdot \max_e c_e + \theta n$, where Δ is the maximum degree of the graph. □

Dynamic networks. Another easy extension concerns dynamically changing networks. That is, the set of available edges may change over time, and we assume that it is also controlled by the adversary. For each time t, we have a set E_t of available edges. The cut condition on the adversary must be satisfied at the specific time when the imbalance is created, i.e., $|\delta_t(S) - |S|(a_{t+1} - a_t)| \leq |e_t(S)|$. Here, $e_t(S)$ are the edges from E_t that have exactly one endpoint in S.

COROLLARY 2.4. *In the above dynamically changing network, for any adversary respecting the cut condition, and any $\theta \geq 1$, the algorithm* $\texttt{SCLB}_\theta$ *is stable, i.e., there is a constant B (depending on H, θ, and G) such that $|b_t(v)| \leq B$ for all nodes v at all times t.*

Proof. By syntactically replacing all terms $e(S)$ with $e_t(S)$ in the proof of Lemma 2.2, we obtain a proof for the model of dynamically changing networks, with exactly the same bound B. □

Time windows. An extension often considered in the contexts of load balancing or packet routing is to relax the restriction on the adversary by allowing it to violate the cut condition for a certain time, provided that it hold "in the long run." Specifically, a *window size* W is specified, and it is required that for any set S and any time window $[t, t + W)$, the imbalance created on set S over that time window be bounded by the total capacity, i.e., $|(\sum_{r=t}^{t+W-1} \delta_r(S)) - |S|(a_{t+W} - a_t)| \leq W \cdot |e(S)|$.

COROLLARY 2.5. *For any adversary respecting the cut condition on average over a window size W, and any $\theta \geq 1$, the algorithm* $\texttt{SCLB}_\theta$ *is stable, i.e., there is a constant B (depending on H, θ, W, and G), such that $|b_t(v)| \leq B$ for all nodes v at all times t.*

Proof. It is not difficult to see that by allowing B to depend on W, we can also extend the stability result to this model—once the imbalance grows too large on a set S, all edges $e \in e(S)$ will be moving tokens so as to reduce the imbalance for every single round of an entire window, so that the imbalance cannot grow further. □

3. Multicommodity load balancing. In the previous section, we considered the problem of balancing loads on processors where the loads were interchangeable. However, we are also interested in the case of different kinds of loads that are to be balanced simultaneously. For instance, think of jobs that have an emphasis on different resources of the machine they are running on. Balancing the different classes of jobs independently could be desirable in order to avoid processing time becoming a bottleneck on one machine and memory size an issue on another.

In the general multicommodity load balancing problem, we have k different kinds of jobs (or tokens), which are stored in separate queues at the nodes. Our goal is to ensure an absolute bound on the deviation of any queue height from the average queue height for that commodity. Each round t is divided into the same two steps as before, the Adversary step and the Redistribution step.

In analogy to the single-commodity case, we use the following notation: For a node v and commodity i, let $h_t^{(i)}(v)$ be the number of tokens of commodity i on node v at the beginning of round t. Similarly, $a_t^{(i)}$, $b_t^{(i)}(v)$, $\delta_t^{(i)}(v)$, $\overline{h}_t^{(i)}(v)$, and $\overline{b}_t^{(i)}(v)$ are all defined for commodity i exactly as their single-commodity equivalents.

The algorithm will now have to choose not only when to send a token across an edge but also which of several available (and conflicting) kinds of tokens to send. Our class of algorithms is practically identical to the one from [2] and [7] and can be formalized as follows:

```
Algorithm MCLB$_\theta$
At each time $t$, for each edge $e = (u, v)$:
    Choose $i$ to maximize $|h_t^{(i)}(u) - h_t^{(i)}(v)|$.
    If $h_t^{(i)}(u) \geq h_t^{(i)}(v) + \theta$, then send a token of commodity $i$ from $u$ to $v$.
    If $h_t^{(i)}(v) \geq h_t^{(i)}(u) + \theta$, then send a token of commodity $i$ from $v$ to $u$.
```

In the case of a single commodity, this algorithm specializes to $\texttt{SCLB}_\theta$.

The natural analogue of the cut condition for a single commodity is to require that the adversary satisfy

$$\sum_i |\delta_t^{(i)}(S) - |S|(a_{t+1}^{(i)} - a_t^{(i)})| \leq |e(S)| \tag{3.1}$$

for all node sets $S \subseteq V$ and times t. This would require that the total imbalance for set S created by the adversary could be "balanced" along edges leaving S.

Unfortunately, inequality (3.1) is too weak a restriction—it allows the adversary to create patterns of addition and removal that cannot be balanced by any algorithm, whether offline or online. At the end of this section, we show how to use a reduction from the multicommodity flow problem to create such an adversary with $k \geq 3$ commodities.

For the special case $k = 2$, however, the Max-Flow Min-Cut Theorem still holds, and, in fact, we can show that the cut condition is sufficient to ensure that algorithm $\texttt{MCLB}_\theta$ is stable.

3.1. Stability for $k = 2$.

We let $H = \max_{v \in V}\{h_1^{(1)}(v) + h_1^{(2)}(v)\}$ be the maximum height of the queues at any node at the start of the execution, and Δ the maximum degree of any vertex. This time, we define γ' slightly differently, namely, $\gamma' = 2\Delta + 2\theta$.

THEOREM 3.1. *There is a constant B (depending on H, θ, and G), such that for any adversary respecting the cut condition,* $\texttt{MCLB}_\theta$ *ensures $|b_t^{(i)}(v)| \leq B$ at all times t, for all vertices v, and for commodities $i = 1, 2$.*

Proof. At the start of the execution, $|b_1^{(1)}(S)| + |b_1^{(2)}(S)| \leq \sum_{v \in S} H$, by definition of H. The key lemma, Lemma 3.2, establishes that for all $S \subseteq V$, times t, and commodities $i = 1, 2$,

$$|b_t^{(1)}(S)| + |b_t^{(2)}(S)| \leq \sum_{j=n-|S|+1}^{n} (H + \gamma' \cdot j). \tag{3.2}$$

We can then apply the result to all singleton sets $\{v\}$, proving the theorem. □

Lemma 3.2. *If* (3.2) *holds at the beginning of round* t, *it holds at the beginning of round* $t+1$.

Proof. The proof is by contradiction. Let S be a set (of minimum size in case of ties) maximizing

$$\Phi(S) := |b_t^{(1)}(S)| + |b_t^{(2)}(S)| - \sum_{j=n-|S|+1}^{n} (H + \gamma' \cdot j).$$

In the case of two commodities, a node might be included in S because it contributes a lot to the imbalance in one of the commodities, although its contribution to the other commodity might actually be negative. To capture the imbalance contribution of a node to each commodity, we define the *signed imbalance* as $\beta_{t'}^{(i)}(v) := \operatorname{sgn}(b_{t+1}^{(i)}(S)) \cdot b_{t'}^{(i)}(v)$, $\overline{\beta}_{t'}^{(i)}(v) := \operatorname{sgn}(b_{t+1}^{(i)}(S)) \cdot \overline{b}_{t'}^{(i)}(v)$ for every node $v \in V$ and every time step t'. Here, sgn denotes the sign of a term. Notice that we use the sign of $b_{t+1}^{(i)}(S)$ at all time steps t' in the definition of $\beta_{t'}^{(i)}(v)$ (not the sign of $b_{t'}^{(i)}(S)$ or of $b_{t'+1}^{(i)}(S)$). We can now rewrite the total imbalance over the set S at time $t+1$ as

$$|b_{t+1}^{(1)}(S)| + |b_{t+1}^{(2)}(S)| = \sum_{u \in S} (\beta_{t+1}^{(1)}(u) + \beta_{t+1}^{(2)}(u)). \tag{3.3}$$

Again, we show that the change in the imbalance for set S cannot be positive, by comparing the increase in the imbalance of S during the Adversary step with the decrease during the Redistribution step, and thus obtain a contradiction.

Adversary step. We know that for each commodity $i = 1, 2$, the imbalance on set S after the Adversary step is at most $|\overline{b}_t^{(i)}(S)| \le |b_t^{(i)}(S)| + |\delta_t^{(i)}(S) - |S| \cdot (a_{t+1}^{(i)} - a_t^{(i)})|$ by the triangle inequality. Now, summing over $i = 1, 2$ and applying the cut condition on the adversary yield that

$$|\overline{b}_t^{(1)}(S)| + |\overline{b}_t^{(2)}(S)| \le |b_t^{(1)}(S)| + |b_t^{(2)}(S)| + |e(S)|.$$

Redistribution step. Fix a node $u \in S$, and an edge $e = (u, v)$ of G with $v \notin S$. As in the proof of Lemma 2.2, we can use the definition of S (as maximizing Φ and being of minimal size) to obtain that the signed imbalances at nodes u and v satisfy $\beta_{t+1}^{(1)}(u) + \beta_{t+1}^{(2)}(u) > \beta_{t+1}^{(1)}(v) + \beta_{t+1}^{(2)}(v) + \gamma'$. As u and v can lose and gain at most Δ tokens each during the Redistribution step,

$$\overline{\beta}_t^{(1)}(u) + \overline{\beta}_t^{(2)}(u) > \overline{\beta}_t^{(1)}(v) + \overline{\beta}_t^{(2)}(v) + 2\theta. \tag{3.4}$$

In particular, there must be a commodity i such that $\overline{\beta}_t^{(i)}(u) > \overline{\beta}_t^{(i)}(v) + \theta$, and thus also $|\overline{h}_t^{(i)}(u) - \overline{h}_t^{(i)}(v)| > \theta$. Hence, $\mathtt{MCLB}_\theta$ moved a token along edge e during the Redistribution step (w.l.o.g., it was a token of commodity 1). We want to show that this token actually decreased the signed imbalance of node u. Assume that it did not. This means that if the token moved from u to v, then $\operatorname{sgn}(b_{t+1}^{(1)}(S))$ is negative, and if the token moved from v to u, then $\operatorname{sgn}(b_{t+1}^{(1)}(S))$ is positive. In either case, the signed imbalance for commodity 1 at node v must be higher than at node u, and so

$$\overline{\beta}_t^{(1)}(v) - \overline{\beta}_t^{(1)}(u) = |\overline{b}_t^{(1)}(v) - \overline{b}_t^{(1)}(u)| = |\overline{h}_t^{(1)}(v) - \overline{h}_t^{(1)}(u)|.$$

Because $\texttt{MCLB}_\theta$ maximizes the difference in its choice of commodity, we obtain that

$$\begin{aligned}\overline{\beta}_t^{(1)}(v) - \overline{\beta}_t^{(1)}(u) &= |\overline{h}_t^{(1)}(v) - \overline{h}_t^{(1)}(u)| \\ &\geq |\overline{h}_t^{(2)}(v) - \overline{h}_t^{(2)}(u)| \\ &= |\overline{\beta}_t^{(2)}(v) - \overline{\beta}_t^{(2)}(u)| \\ &\geq \overline{\beta}_t^{(2)}(u) - \overline{\beta}_t^{(2)}(v).\end{aligned}$$

Rearranging this inequality yields that

$$\overline{\beta}_t^{(1)}(v) + \overline{\beta}_t^{(2)}(v) \geq \overline{\beta}_t^{(1)}(u) + \overline{\beta}_t^{(2)}(u)$$

and thus results in a contradiction with inequality (3.4). Therefore, every edge (u, v) with $v \notin S$ decreases the signed imbalance of u by 1. Summing over all edges and all nodes $u \in S$ gives us $\beta_{t+1}^{(1)}(S) + \beta_{t+1}^{(2)}(S) \leq \overline{\beta}_t^{(1)}(S) + \overline{\beta}_t^{(2)}(S) - |e(S)|$. Using (3.3) and the fact that $\overline{\beta}_t^{(1)}(S) + \overline{\beta}_t^{(2)}(S) \leq |\overline{b}_t^{(1)}(S)| + |\overline{b}_t^{(2)}(S)|$, we obtain $|b_{t+1}^{(1)}(S)| + |b_{t+1}^{(2)}(S)| \leq |\overline{b}_t^{(1)}(S)| + |\overline{b}_t^{(2)}(S)| - |e(S)|$.

Combining the arguments for the two steps $|b_t^{(1)}(S)| + |b_t^{(2)}(S)|$ increases by at most $|e(S)|$ during the Adversary step and decreases by at least $|e(S)|$ during the Redistribution step. Therefore, in total $|b_{t+1}^{(1)}(S)| + |b_{t+1}^{(2)}(S)| \leq |b_t^{(1)}(S)| + |b_t^{(2)}(S)|$, which is a contradiction. □

The result for two commodities can be extended to networks with edge capacities, adversarially changing edge sets, and adversaries with restrictions only for larger window sizes, just like for the single-commodity case.

3.2. Load balancing and flows. By omitting the adversarial and dynamic nature in the load balancing problem, and forcing the adversary to repeat the same pattern of token additions in every round, we can infer from the stability of the load balancing algorithm the existence of a multicommodity flow. Suppose that we are given a multicommodity flow instance with edge capacities c_e, source-sink pairs (s_i, t_i), and demands d_i. Let $D = \max_i d_i$, and let $\mathcal{A}$ be the adversary inserting, in every round and for each commodity i, $D + d_i$ tokens at node s_i, $D - d_i$ tokens at node t_i, and D tokens everywhere else.

LEMMA 3.3. *If there is a load balancing algorithm that is stable against the adversary $\mathcal{A}$, then there is a (fractional) multicommodity flow f with source-sink pairs (s_i, t_i) and demands d_i.*

Proof. Because we assumed the algorithm to be stable, all imbalances $b_t^{(i)}(v)$ are always bounded in absolute value by some constant B. Therefore, there are at most $(2B+1)^{kn}$ different combinations of imbalances for the entire network, and so there must exist two time steps t and t' such that $t < t'$ and $b_t^{(i)}(v) = b_{t'}^{(i)}(v)$ for all nodes v and commodities i.

For each edge $e = (u, v)$, let $\sigma_r^{(i)}(u, v)$ denote the number of tokens of commodity i sent from u to v in round r, and define a flow f by

$$f_{(u,v)}^{(i)} := \frac{1}{t' - t} \cdot \sum_{r=t}^{t'-1} (\sigma_r^{(i)}(u, v) - \sigma_r^{(i)}(v, u)).$$

Notice that we define negative flows, but only for symmetry and ease of notation. We want to verify that f is indeed a feasible multicommodity flow for demands (s_i, t_i, d_i).

Capacity constraints. The total flow along any edge (u,v) is

$$\begin{aligned}\sum_i |f^{(i)}_{(u,v)}| &\le \frac{1}{t'-t}\cdot\sum_i\sum_{r=t}^{t'-1}|\sigma^{(i)}_r(u,v)-\sigma^{(i)}_r(v,u)|\\ &= \frac{1}{t'-t}\cdot\sum_{r=t}^{t'-1}\sum_i|\sigma^{(i)}_r(u,v)-\sigma^{(i)}_r(v,u)|\\ &\le \frac{1}{t'-t}\cdot\sum_{r=t}^{t'-1}c_{(u,v)}\\ &= c_{(u,v)}.\end{aligned}$$

The first inequality is simply the triangle inequality, and the second inequality holds because the balancing algorithm never exceeds the capacity of any edge with any of its token moves, and therefore both $\sigma^{(i)}_r(u,v)$ and $\sigma^{(i)}_r(v,u)$ lie between 0 and $c_{(u,v)}$.

Flow conservation. For any node v and commodity i, we can write

$$\begin{aligned}&(t'-t)\sum_{(u,v)\in E} f^{(i)}_{(u,v)}\\ &=\sum_{(u,v)\in E}\sum_{r=t}^{t'-1}(\sigma^{(i)}_r(u,v)-\sigma^{(i)}_r(v,u))\\ &=\sum_{r=t}^{t'-1}\sum_{(u,v)\in E}\sigma^{(i)}_r(u,v)-\sum_{r=t}^{t'-1}\sum_{(u,v)\in E}\sigma^{(i)}_r(v,u)\\ &=h^{(i)}_{t'}(v)-h^{(i)}_t(v)-\sum_{r=t}^{t'-1}\delta^{(i)}_r(v)\\ &=b^{(i)}_{t'}(v)+a^{(i)}_{t'}-b^{(i)}_t(v)-a^{(i)}_t-(t'-t)\cdot\delta^{(i)}_t(v)\\ &=(t'-t)(D-\delta^{(i)}_t(v)).\end{aligned}$$

The third equality is true because $h^{(i)}_{t'}(v)-h^{(i)}_t(v)$ is exactly the number of tokens that entered v during the time period $[t,t'-1]$, minus the number of tokens that left v during this period. In the last equality, we used that $b^{(i)}_{t'}(v)=b^{(i)}_t(v)$ for all i and v, and that $a^{(i)}_r=r\cdot D$ for all times r. Now, if node v is neither the source nor the sink for commodity i, then $\delta^{(i)}_t(v)=D$, so flow is conserved. If v is the source of commodity i, then $\delta^{(i)}_t(v)=D+d_i$, so the total flow entering node s_i is $-d_i$. If v is the sink for commodity i, then the total flow entering node t_i is d_i, because $\delta^{(i)}_t(v)=D-d_i$ by definition. Hence, f satisfies flow conservation and all demands.

f conserves flow, satisfies all demands, and does not exceed any edge capacities, so it is a feasible multicommodity flow for the given demands. □

By combining Lemma 3.3 with the stability of $\mathtt{MCLB}_\theta$ proved in Theorem 3.1, we obtain as a corollary an alternate proof of the 2-commodity Max-Flow Min-Cut Theorem. It remains only to verify that the adversary $\mathcal{A}$ as defined in Lemma 3.3 indeed respects the cut condition.

COROLLARY 3.4 (2-commodity Max-Flow Min-Cut). *Let $G=(V,E)$ be a graph with edge capacities c_e and two demand pairs (s_1,t_1), (s_2,t_2) with demands d_1, d_2*

such that for any vertex set $S \subseteq V$, the total demand of commodities $i \in \{1, 2\}$ with exactly one of $\{s_i, t_i\}$ in S is at most $\sum_{e \in e(S)} c_e$. Then, there exists a feasible 2-commodity flow sending d_i units of flow from s_i to t_i for $i = 1, 2$.

Proof. Define the adversary $\mathcal{A}$ as in Lemma 3.3. To show that the algorithm $\texttt{MCLB}_\theta$ is stable against $\mathcal{A}$, we merely have to verify that $\mathcal{A}$ satisfies the cut condition. Let $S \subseteq V$ be arbitrary. For convenience, we write $[u \in S] := 1$ if $u \in S$, and 0 otherwise. Then, for any time t

$$\begin{aligned}
&\sum_{i=1,2} |\delta_t^{(i)}(S) - |S| \cdot (a_{t+1}^{(i)} - a_t^{(i)})| \\
&= \sum_{i=1,2} ||S| \cdot D + [s_i \in S] \cdot d_i - [t_i \in S] \cdot d_i - |S| \cdot D| \\
&= \sum_{i=1,2} d_i \cdot |[s_i \in S] - [t_i \in S]|.
\end{aligned}$$

In the first equality, we used the definition of the insertion pattern for $\mathcal{A}$. The contribution of commodity i to this sum is d_i if and only if exactly one of s_i, t_i lies in S—otherwise, it is 0. Hence, the value of the sum is the total demand of commodities i with exactly one of $\{s_i, t_i\}$ in S, which by assumption is bounded by $\sum_{e \in e(S)} c_e$. Hence, $\mathcal{A}$ satisfies the cut condition.

We can therefore apply Theorem 3.1 to obtain that $\texttt{MCLB}_\theta$ is stable against $\mathcal{A}$, which in turn implies the existence of a feasible multicommodity flow f for the given instance via Lemma 3.3. □

The 2-commodity Max-Flow Min-Cut Theorem was first proved by Hu [19], essentially repeating Ford and Fulkerson's original [16] augmenting paths argument for two commodities. Seymour [25] showed a short and simple explicit reduction to the single-commodity case. Subsequently, Linial, London, and Rabinovich [21] gave a novel proof using geometric embeddings and linear programming duality. Our proof uses yet different (and much more elementary) techniques and does not rely on the single-commodity Max-Flow Min-Cut Theorem.

Another corollary we obtain from Lemma 3.3 is the existence of an adversary respecting the cut condition for $k = 3$ commodities, such that no algorithm (offline or online) can balance the insertion pattern. This is not surprising, since load balancing algorithms (and our algorithm, in particular) attempt to generate a flow from overloaded nodes to underloaded nodes, and we know that the best flows can be significantly worse than the best cuts for more than two commodities (i.e., the Max-Flow Min-Cut Theorem does not hold for $k \geq 3$). This means that the constraint on the adversary is not powerful enough to guarantee the existence of a good flow and therefore a good algorithm. We make the above discussion precise in the following corollary.

COROLLARY 3.5. *There exists an adversary respecting the cut condition for $k = 3$ commodities, such that no algorithm (offline or online) is stable against this adversary.*

Proof. To prove this, we simply take a 3-commodity instance with a graph $G = (V, E)$ and demand pairs (s_i, t_i), $i \in \{1, 2, 3\}$ (with demands d_i), such that for all cuts $(S, V \backslash S)$, the total demand across the cut is at most the capacity of the edges crossing the cut, yet there is no (fractional) multicommodity flow satisfying all demands. The first such example for $k = 3$ was given in [19]. (A simpler well-known example for $k = 4$ is the complete bipartite graph $K_{2,3}$ with a unit demand between every pair of nodes that are at a distance of 2 from each other.) Let $\mathcal{A}$ be the adversary defined from this instance as in Lemma 3.3. If any load balancing algorithm were stable against $\mathcal{A}$, Lemma 3.3 would guarantee a feasible multicommodity flow, which is a contradiction. $\square$

The above corollary and Lemma 3.3 illustrate the relation of multicommodity flows to load balancing algorithms and tell us that the cut condition is not enough to guarantee the existence of a stable algorithm for $k \geq 3$. In section 5, we will discuss an alternative approach for a restriction on a multicommodity adversary.

4. Packet routing. There is a natural connection between the load balancing problem studied in the previous sections and the problem of routing packets in an adversarial network. It has been observed previously [2, 7] that the natural balancing algorithm $\texttt{SCLB}_\theta$ is also stable for packet routing.

The model for packet routing differs from the load balancing one in that after the Redistribution step, there is an additional *Removal step*, during which all packets which have reached their destination are removed from the network. Stability of an algorithm is now defined as meaning that there is an absolute bound on all queue heights at all times, i.e., $h_t^{(i)}(v) \leq B$ for some constant B.

In the single-commodity packet routing problem, we can again restrict the adversary by a cut condition: the total number $\delta_t(S)$ of packets inserted into a set S must be at most $|e(S)|$ for any set S not containing the sink of the packets. If S does contain the sink, then there is no restriction. In the multicommodity case, the adversary specifies a source s_i and a sink t_i for each packet inserted and must guarantee that there is a set of edge-disjoint paths connecting all (s_i, t_i) pairs.

In [7], it was shown that for the single-commodity case, the algorithm $\texttt{SCLB}_\theta$ is stable against an adversary guaranteeing the existence of a path for all packets inserted, even when edges dynamically appear and disappear. Aiello et al. [2] proved that an algorithm essentially equivalent to $\texttt{MCLB}_\theta$ is stable for the multicommodity packet routing problem if the paths specified by the adversary not only are disjoint but also leave an ε fraction of capacity for every edge unused over a given window length W. Recently, [5] extended this result to dynamically changing networks.

Our techniques from section 2 can be used to obtain an alternate (and simpler) proof for the stability of algorithm $\texttt{SCLB}_\theta$ in the packet routing model. Our proof also works for the case of adversarially appearing and disappearing edges, although the restriction on the adversary is different from (and essentially less general than) the one in [7].

We define Δ and H as before and let $\gamma = 2\Delta + \theta$. Then, the stability of $\texttt{SCLB}_\theta$ against an adversary respecting the cut condition is guaranteed by the following theorem.

THEOREM 4.1. *For any time t and set $S \subseteq V$,*

$$h_t(S) \leq \sum_{j=n-|S|+1}^{n} (H + \gamma \cdot j). \tag{4.1}$$

Proof. By definition of H, invariant (4.1) certainly holds at time 1. Assume that the theorem is wrong, and let t be the earliest time such that there is a set S violating (4.1) at time $t+1$. Among all such sets, let S be the one maximizing $h_{t+1}(S) - \sum_{j=n-|S|+1}^{n} (H + \gamma \cdot j)$, and break ties for minimal size. Then, we can show as in the proof of Lemma 2.2 that for all nodes $u \in S$, $v \notin S$, $h_{t+1}(u) > h_{t+1}(v) + \gamma$, and $h_{t+1}(u) > H$. In particular, the set S cannot contain the sink (because the sink contains no tokens after the Removal step).

Therefore, the adversary can have inserted at most $|e(S)|$ tokens into S in the Adversary step. During the Redistribution step of round t, a token leaves the set S along each edge $e = (u, v) \in e(S)$, because even if u had lost Δ tokens and v had gained Δ tokens during the Redistribution step, $\overline{h}_t(u)$ would still exceed $\overline{h}_t(v)$ by at least θ. Taken together, this shows that the number of tokens in S cannot have increased, contradicting the choice of t and S. □

Extensions. Like the proofs in the previous sections, the proof of Theorem 4.1 can be easily extended to deal with dynamically changing networks, edge capacities, and time windows in the cut condition. In addition, this proof extends to directed graphs G if we redefine $e(S)$ in the cut condition to be the edges coming out of S. We can also show stability for a wider class of single-commodity balancing algorithms. Specifically, let $g : \mathbb{N} \to \mathbb{N}$ be a function with $g(x) \geq x$ for all x. The balancing algorithm $\texttt{SCLB}_g$ always sends a token from u to v (and never sends a token from v to u) if there is an edge $e = (u, v) \in E$ and $h_t(u) > g(h_t(v))$. It does not matter what the algorithm does if neither $h_t(u) > g(h_t(v))$ nor $h_t(v) > g(h_t(u))$. Extending the above proof only slightly, we obtain that $\texttt{SCLB}_g$ is stable for all such functions g. Of course, the bound B now depends on the rate of growth of g.

Our packet routing results also imply the stability of an interesting load balancing scenario, which is very similar to the one in [11]. Suppose that each node processes one job per round so that one token is removed in every round from each node with positive queue height. If the adversary $\mathcal{A}$ satisfies the constraint that the number of tokens it adds to S be at most $e(S) + |S|$, then we can use Theorem 4.1 to show that the queue heights are bounded. Consider adding a sink node v to our graph, and add an edge from every node to v. We can now think of $\mathcal{A}$ as adding packets to the new graph, with v being the packets' destination. The number of edges coming out of a set S in the new graph is exactly $e'(S) = e(S) + |S|$, so $\mathcal{A}$ satisfies the condition needed for Theorem 4.1. It is easy to see that if the queue heights are bounded in this new packet routing scenario, then they are also bounded in the original load balancing one.

The proofs for Theorems 4.1 and 2.1 (and the proofs in [2] and [23]) are so similar in nature that one suspects a formal reduction from the packet routing problem to the load balancing problem (which seems more general). However, we have not yet been able to determine such a reduction. It would certainly be interesting, since it would allow us to focus on the load balancing problem in the future.

As with the load balancing problem, we can obtain a multicommodity flow if $\texttt{MCLB}_\theta$ is stable against a suitably defined adversary $\mathcal{A}$. The proof is practically identical to the one for Lemma 3.3, and we therefore omit it.

LEMMA 4.2. *Let $\mathcal{A}$ be an adversary inserting d_i tokens of commodity i (whose destination is t_i) into node s_i in every round. If there is a routing algorithm that is stable against this adversary, then there is a (fractional) multicommodity flow f with source-sink pairs (s_i, t_i) and demands d_i.*

5. Conclusions.

In this paper, we have shown that a simple local load balancing algorithm is stable against dynamic adversarial addition and removal of jobs in a network, so long as the adversary is bounded by a natural extension of the cut condition in the sense defined in [23]. This settles an open question from [23]. Our proof techniques extend to the cases of balancing two commodities at once and to routing packets injected by an adversary. They yield easier proofs and essentially tight bounds for the general case. In addition, the stability of the load balancing algorithm for two commodities gives a new proof of the 2-commodity Max-Flow Min-Cut Theorem.

This work leaves open a number of interesting questions. Most importantly, we would like to be able to show stability of the multicommodity load balancing algorithm for an arbitrary number of commodities, both for the problem of routing packets and balancing loads. If we want to prove the stability of load balancing algorithms for more commodities, we will have to use a different condition on the adversary. As a consequence of Lemma 3.3, any reasonable restriction on the adversary will have to guarantee the existence of multicommodity flows for all instances where we hope to prove stability. We therefore suggest the following restriction:

> First, define demands $d_t^{(i)}(v)$ for commodity i, node v, and time t by $d_t^{(i)}(v) := \delta_t^{(i)}(v) - (a_{t+1}^{(i)} - a_t^{(i)})$. Then, the adversary is restricted to moves that guarantee the existence of a (fractional) multicommodity flow in G satisfying all these demands.

The disadvantage of this condition is that it bears no direct relation to the load balancing problem—it arises from observing the insufficiency of the more natural cut condition rather than from having an actual meaning for the problem of balancing loads. Nevertheless, this condition should be considered the right restriction on the adversary to measure the quality of $\texttt{MCLB}_\theta$ or other load balancing algorithms.

Alternatively, we might investigate whether the cut condition is sufficient for balancing multiple loads if we restrict our attention to specific networks. For example, it is well known that for trees or cycles, the cut condition implies the existence of multicommodity flows, and we might hope that it would hence be sufficient to prove stability.

Acknowledgments. We thank Martin Pál and Christian Scheideler for valuable discussions on the subjects of load balancing and adversarial packet routing.

REFERENCES

[1] W. Aiello, B. Awerbuch, B. Maggs, and S. Rao, *Approximate load balancing on dynamic and asynchronous networks*, in Proceedings of the 25th ACM Symposium on Theory of Computing, ACM, New York, 1993, pp. 632–641.

[2] W. Aiello, E. Kushilevitz, R. Ostrovsky, and A. Rosén, *Adaptive packet routing for bursty adversarial traffic*, in Proceedings of the 30th ACM Symposium on Theory of Computing, ACM, New York, 1998, pp. 359–368.

[3] A. Anagnostopoulos, A. Kirsch, and E. Upfal, *Load balancing in arbitrary network topologies with stochastic adversarial input*, SIAM J. Comput., 34 (2005), pp. 616–639.

[4] D. M. Andrews, B. Awerbuch, A. Fernández, J. Kleinberg, F. T. Leighton, and Z. Liu, *Universal stability results for greedy contention–resolution protocols*, J. ACM, 48 (2001), pp. 39–69.

[5] M. Andrews, K. Jung, and A. Stolyar, *Stability of the max-weight routing and scheduling protocol in dynamic networks and at critical loads*, in Proceedings of the 38th ACM Symposium on Theory of Computing, ACM, New York, 2007, pp. 145–154.

[6] F. Meyer auf der Heide, B. Oesterdiekhoff, and R. Wanka, *Strongly adaptive token distribution*, Algorithmica, 15 (1996), pp. 413–427.

[7] B. Awerbuch, P. Berenbrink, A. Brinkmann, and C. Scheideler, *Simple routing strategies for adversarial systems*, in Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 2001, pp. 158–167.

[8] B. Awerbuch and F. T. Leighton, *A simple local-control approximation algorithm for multicommodity flow*, in Proceedings of the 34th IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 1993, pp. 459–468.

[9] B. Awerbuch and F. T. Leighton, *Improved approximation algorithms for the multicommodity flow problem and local competitive routing in dynamic networks*, in Proceedings of the 26th ACM Symposium on Theory of Computing, ACM, New York, 1994.

[10] P. Berenbrink, T. Friedetzky, and L. A. Goldberg, *The natural work-stealing algorithm is stable*, SIAM J. Comput., 32 (2003), pp. 1260–1279.

[11] P. Berenbrink, T. Friedetzky, and R. Martin, *Dynamic diffusion load balancing*, in Proceedings of the 32nd International Colloquium on Automata, Languages and Programming, Lisbon, Portugal, 2005, pp. 1386–1398.

[12] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. Williamson, *Adversarial queueing theory*, J. ACM, 48 (2001), pp. 13–38.

[13] M. Crovella, M. Harchol-Balter, and C. Murta, *Task assignment in a distributed system: Improving performance by unbalancing load*, in Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems, ACM, New York, 1998, pp. 268–269.

[14] G. Cybenko, *Load balancing for distributed memory multiprocessors*, J. Parallel Distrib. Comput., 7 (1989), pp. 279–301.

[15] D. Eager, E. Lazowska, and J. Zahorjan, *Adaptive load sharing in homogeneous distributed systems*, IEEE Trans. Software Engineering, 12 (1986), pp. 662–675.

[16] L. Ford and D. Fulkerson, *Maximal flow through a network*, Canad. J. Math., 8 (1956), pp. 399–404.

[17] D. Gamarnik, *Stability of adaptive and non-adaptive packet routing policies in adversarial queueing networks*, in Proceedings of the 31st ACM Symposium on Theory of Computing, ACM, New York, 1999, pp. 206–214.

[18] B. Ghosh, F. T. Leighton, B. Maggs, S. Muthukrishnan, C. Plaxton, R. Rajaraman, A. Richa, R. Tarjan, and D. Zuckerman, *Tight analyses of two local load balancing algorithms*, in Proceedings of the 27th ACM Symposium on Theory of Computing, ACM, New York, 1995.

[19] T. C. Hu, *Multi-commodity network flows*, Oper. Res., 11 (1963), pp. 344–360.

[20] F. T. Leighton and S. Rao, *Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms*, J. ACM, 46 (1999), pp. 787–832.

[21] N. Linial, E. London, and Y. Rabinovich, *The geometry of graphs and some of its algorithmic applications*, Combinatorica, 15 (1995), pp. 215–245.

[22] M. Mitzenmacher, *On the analysis of randomized load balancing schemes*, in Proceedings of the 9th ACM Symposium on Parallel Algorithms and Architectures, ACM, New York, 1997, pp. 292–301.

[23] S. Muthukrishnan and R. Rajaraman, *An adversarial model for distributed dynamic load balancing*, in Proceedings of the 10th ACM Symposium on Parallel Algorithms and Architectures, ACM, New York, 1998, pp. 47–54.

[24] D. Peleg and E. Upfal, *The token distribution problem*, SIAM J. Comput., 18 (1989), pp. 229–243.

[25] P. D. Seymour, *A short proof of the two-commodity flow theorem*, J. Combin. Theory Ser. B, 26 (1979), pp. 370–371.

[26] B. Shirazi, A. Hurson, and K. Kavi, *Scheduling and Load Balancing in Parallel and Distributed Systems*, IEEE Computer Society, Los Alamitos, CA, 1995.

SIAM J. COMPUT.
Vol. 37, No. 5, pp. 1674–1701

THE COMPLEXITY OF QUANTIFIED CONSTRAINT SATISFACTION: COLLAPSIBILITY, SINK ALGEBRAS, AND THE THREE-ELEMENT CASE[*]

HUBIE CHEN[†]

Abstract. The constraint satisfaction probem (CSP) is a well-acknowledged framework in which many combinatorial search problems can be naturally formulated. The CSP may be viewed as the problem of deciding the truth of a logical sentence consisting of a conjunction of constraints, in front of which all variables are existentially quantified. The quantified constraint satisfaction problem (QCSP) is the generalization of the CSP where universal quantification is permitted in addition to existential quantification. The general intractability of these problems has motivated research studying the complexity of these problems under a restricted *constraint language*, which is a set of relations that can be used to express constraints. This paper introduces collapsibility, a technique for deriving positive complexity results on the QCSP. In particular, this technique allows one to show that, for a particular constraint language, the QCSP reduces to the CSP. We show that collapsibility applies to three known tractable cases of the QCSP that were originally studied using disparate proof techniques in different decades: QUANTIFIED 2-SAT (Aspvall, Plass, and Tarjan in 1979), QUANTIFIED HORN-SAT (Karpinski, Kleine Büning, and Schmitt in 1987), and QUANTIFIED AFFINE-SAT (Creignou, Khanna, and Sudan in 2001). This reconciles and reveals common structure among these cases, which are describable by constraint languages over a two-element domain. In addition to unifying these known tractable cases, we study constraint languages over domains of larger size.

Key words. computational complexity, constraint satisfaction, quantified satisfiability, universal algebra

AMS subject classifications. 08A70, 68Q17, 68Q25

DOI. 10.1137/060668572

1. Introduction.

1.1. Background. The *constraint satisfaction problem (CSP)* is a general framework in which many combinatorial search problems can be naturally formulated. An instance of the CSP consists of a set of variables, a domain, and a set of *constraints* on the variables, where a constraint is a pair consisting of a tuple $(v_1, \dots, v_k)$ of variables and a relation $R \subseteq D^k$ over the domain D that specifies allowed values for the variable tuple. The question is to decide whether or not there exists an assignment to the variables satisfying all of the constraints. Examples of problems that fall into this framework include boolean satisfiability problems [30, 11], graph homomorphism problems [19, 18], and the problem of solving a system of equations over an algebraic structure [27]. The CSP can be equivalently formulated as the problem of deciding if there is a homomorphism between two relational structures [17], as well as the database problems of conjunctive-query containment and evaluation [26].

In its general formulation, the CSP is NP-complete; this intractability, coupled with the ubiquity of the CSP, has given rise to a far-reaching research program that aims to identify restricted cases of the CSP that are polynomial-time tractable. One of the principal directions within this program is the study of the CSP where the set

[*] Received by the editors August 30, 2006; accepted for publication (in revised form) October 29, 2007; published electronically February 14, 2008.
http://www.siam.org/journals/sicomp/37-5/66857.html

[†] Departament de Tecnologia, Universitat Pompeu Fabra, 08002 Barcelona, Spain (hubie.chen@upf.edu).

of relations permitted in constraints, called the *constraint language*, is restricted. This direction has its origins in a 1978 paper of Schaefer [30], who gave a classification theorem showing that all constraint languages over a two-element domain give rise to a case of the CSP that is either polynomial-time tractable or NP-complete. The nontrivial tractable cases given by this result all readily reduce to one of the following three problems: 2-SAT, where each constraint is equivalent to a clause of length 2; Horn-SAT, where each constraint is equivalent to a propositional Horn clause; or Affine-SAT, where each constraint is an equation over the two-element field. In the 1990s, an approach to studying the complexity of constraint languages based on concepts and tools from universal algebra was introduced [22]. A key idea underlying this approach is to associate, to each constraint language, an algebra whose operations are the *polymorphisms* of the constraint language; roughly speaking, an operation is a polymorphism of a constraint language Γ if each relation of Γ is closed under the operation. This algebra is then used to derive information about the constraint language. One celebrated achievement of this algebraic viewpoint is the CSP complexity classification of constraint languages over a three-element domain, which is due to Bulatov [9]. See [13, 8, 5, 7, 15, 6, 24] and the references therein for more examples of work along these lines.

The present work is concerned with the *quantified constraint satisfaction problem (QCSP)*. Viewing CSP as the problem of deciding a logical sentence consisting of a conjunction of constraints and a quantifier prefix in which all variables are existentially quantified, the QCSP can be defined as the generalization of the CSP where universal quantification is permitted in addition to existential quantification. As is well known, the extra expressiveness of the QCSP comes at the cost of complexity: the QCSP is in general PSPACE-complete. Indeed, the Quantified Boolean Formula (QBF) problem, of which the QCSP is a generalization, was historically one of the first problems recognized to be PSPACE-complete [31], and is now a prototypical example of a PSPACE-complete problem.

A classification theorem describing the QCSP complexity of constraint languages over a two-element domain has been established [30, 12, 11]. The polynomial-time tractable cases of the QCSP given by this classification correspond to the problems Quantified 2-SAT, Quantified Horn-SAT, and Quantified Affine-SAT. These are precisely the quantified generalizations of the discussed tractable cases of Schaefer's CSP theorem. Although the tractability of all three of these problems was claimed without proof by Schaefer [30], the first published proofs we are aware of for these problems were given in different decades, and were proved using disparate proof techniques: Aspvall, Plass, and Tarjan [1] proved the tractability of Quantified 2-SAT in 1979 by giving an algorithm which analyzes a directed graph, called the *implication graph*, induced by a problem instance; Karpinski, Kleine Büning, and Schmitt [23] proved the tractability of Quantified Horn-SAT in 1987 by studying a generalized form of unit resolution (see [25] for subsequent work on this problem); and Creignou, Khanna, and Sudan [11] proved the tractability of Quantified Affine-SAT in 2001 by giving an algorithm that iteratively eliminates the innermost quantified variable.

1.2. Contributions. In this article, we introduce *collapsibility*, a technique for deriving positive complexity results on the QCSP, which is based on the algebraic approach to studying constraint languages. This technique allows one to show, for certain constraint languages, that the QCSP over the constraint language polynomial-time reduces to the CSP over the constraint language, via a simple reduction. Thus, this technique, when applied to a constraint language under which the CSP is known to be tractable, implies the tractability of the QCSP under the constraint language. *We show that this technique applies to all three of the aforementioned tractable cases of the QCSP over a two-element domain.* This reconciles and reveals common structure among these three cases, which, as discussed, were originally studied using strikingly disparate techniques.

In addition to unifying known QCSP tractability results, the technique of collapsibility allows us to make significant progress towards understanding the complexity of the QCSP in domains of arbitrary size. For example, we are able to demonstrate the QCSP tractability of the large classes of constraint languages studied, in the CSP setting, by Jeavons, Cohen, and Cooper [21] and Bulatov and Dalmau [6]. We are also able to classify the *conservative constraint languages* giving rise to a tractable QCSP; a conservative constraint language is a constraint language containing all unary relations, and such constraint languages were studied by Bulatov [5] in the CSP.

Based on the notion of collapsibility, we identify a class of algebras that we call *sink algebras*, and show that any constraint language whose algebra "excludes" sink algebras (in a manner made precise) and does not obey a known sufficient condition for CSP intractability is amenable to our collapsibility technique. We analyze three-element sink algebras and show that they all contain a particular semilattice operation. This in turn allows us to give a classification for the three-element case up to a "forbidden polymorphism": for the constraint languages over a three-element domain not having the semilattice operation as a polymorphism, we provide a description of exactly which give rise to a tractable QCSP.

Overall, the techniques of this article involve an interplay among the areas of complexity theory, algebra, and logic.

1.3. Other related work.

The other papers on the QCSP and constraint languages on domains of arbitrary size that we are aware of are Börner et al. [3] and Martin and Madelaine [28]. The results of Börner et al. [3] include the identification of a Galois connection relevant to the QCSP and two tractable cases of the QCSP. Our collapsibility technique applies to the tractable cases they give, which are defined in terms of Mal'tsev polymorphisms and dual discriminator polymorphisms. Martin and Madelaine [28] study constraint languages consisting of a single binary relation that is symmetric and antireflexive, that is, an undirected graph; they obtain both tractability and intractability results for such constraint languages.

1.4. Organization.

This article is organized as follows. In section 2, we present the terminology, notation, and background concepts to be used throughout the paper. In section 3, we give the basic definitions and results that underlie our collapsibility technique. In particular, we define what it means for a constraint language to be *collapsible*, and show that the QCSP over a collapsible constraint language reduces to the CSP over the same constraint language. Section 4 presents a theorem that can be used to prove the collapsibility of constraint languages and gives example applications of the theorem. In section 5, we develop algebraic machinery for demonstrating collapsibility results and illustrate our ideas using examples. Section 6 defines and studies *sink algebras*. In section 7, we analyze three-element sink algebras, showing that any such algebra must have a particular structure; this permits us to give our classification of constraint languages over a three-element domain.

2. Preliminaries.

We use $[n]$ to denote the set containing the first n positive integers, $\{1, \ldots, n\}$. When $f : A^k \to A$ is an operation on a set A and $B_1, \ldots, B_k \subseteq A$ are subsets of A, we use $f(B_1, \ldots, B_k)$ to denote the set $\{f(b_1, \ldots, b_k) : b_1 \in$

$B_1, \ldots, b_k \in B_k\}$. When $f : A^k \to A$ is an operation on a set A and $B \subseteq A$, we use $f|_B$ to denote the restriction of f to B^k, and when F is a set of functions $f : A^k \to A$, we use $F|_B$ to denote $\{f|_B : f \in F\}$.

2.1. Quantified constraint satisfaction. We now describe the basic terminology of quantified constraint satisfaction to be used throughout the paper.

DEFINITION 2.1. *Let A be a set. A* relation *over the set A is a subset of A^k for some integer $k \geq 1$, called the arity of the relation. A* constraint *over the set A is a formula of the form $R(w_1, \ldots, w_k)$, where R is a relation over A of arity k (viewed as a predicate) and each w_i is either a variable or a* constant *(an element of A). A* constraint language *is a set of relations, all of which are over the same domain.*

Note that, in this paper, we always permit constants to appear in constraints. Clearly, our positive complexity results will apply to constraints not containing any constants, since such constraints are a subclass of the constraints we allow here.

DEFINITION 2.2. *A* quantified constraint formula *is a formula of the form*

$$Q_1 v_1 \ldots Q_m v_m \mathcal{C}$$

having an associated set A called the domain, where

- *for all $i \in [m]$, Q_i is a quantifier from the set $\{\forall, \exists\}$ and v_i is a variable,*
- *the variables $\{v_1, \ldots, v_m\}$ are assumed to be pairwise distinct, and*
- *$\mathcal{C}$ is a finite conjunction of constraints over A having variables from the set $\{v_1, \ldots, v_m\}$.*

A quantified constraint formula is said to be over a constraint language Γ if each of its constraints has relation from Γ.

Truth of a quantified constraint formula is defined as in first-order logic. Note that the quantification of the variables is understood to be over the domain A of the formula. We will generally use A to denote the domain of a quantified constraint formula. We assume that all domains of quantified constraint formulas are finite.

The QCSP can now be defined as the problem of deciding, given a quantified constraint formula, whether or not it is true. We are interested in the following parameterized version of the QCSP.

DEFINITION 2.3. *Let Γ be a constraint language. The decision problem $\mathsf{QCSP}_{\mathsf{c}}(\Gamma)$ is to decide, given as input a quantified constraint formula over Γ, whether or not it is true.*

We will also discuss and use the following parameterized version of the CSP.

DEFINITION 2.4. *The decision problem $\mathsf{CSP}_{\mathsf{c}}(\Gamma)$ is the restriction of $\mathsf{QCSP}_{\mathsf{c}}(\Gamma)$ to quantified constraint formulas having only existential quantifiers.*

In the previous two definitions, we use the subscript c to emphasize that constants are permitted in constraints.

We now review a characterization of truth for quantified constraint formulas which will be used throughout this paper. This characterization comes from the concept of Skolemization [16] and conceives of a quantified constraint formula as a game between two players: a *universal player* that sets the universally quantified variables and an *existential player* that sets the existentially quantified variables. Variables are set in the order dictated by the quantifier prefix, and the existential player is said to win if, after the variables have been set, the conjunction of constraints is true. The formula is true if and only if the existential player can always win, no matter how the universal player sets the universally quantified variables.

We now formalize this viewpoint. When Φ is a quantified constraint formula, let V^Φ denote the variables of Φ, let E^Φ denote the existentially quantified variables of

Φ, let U^Φ denote the universally quantified variables of Φ, and for each $x \in E^\Phi$, let U_x^Φ denote the variables in U^Φ that come before x in the quantifier prefix of Φ. Let $[B \to A]$ denote the set of functions mapping from B to A.

DEFINITION 2.5. *A* strategy *for a quantified constraint formula Φ is a sequence of partial functions*

$$\sigma = \{\sigma_x : [U_x^\Phi \to A] \to A\}_{x \in E^\Phi}.$$

That is, a strategy has a mapping σ_x for each existentially quantified variable $x \in E^\Phi$, which tells how to set the variable x in response to an assignment to the universal variables coming before x. Let $\tau : U^\Phi \to A$ be an assignment to the universal variables. We define $\langle \sigma, \tau \rangle$ to be the mapping from V^Φ to A such that $\langle \sigma, \tau \rangle(v) = \tau(v)$ for all $v \in U^\Phi$, and $\langle \sigma, \tau \rangle(x) = \sigma_x(\tau|_{U_x^\Phi})$ for all $x \in E^\Phi$. The mapping $\langle \sigma, \tau \rangle$ is undefined if $\sigma_x(\tau|_{U_x^\Phi})$ is not defined for all $x \in E^\Phi$. The intuitive point here is that a strategy σ along with an assignment τ to the universally quantified variables naturally yields an assignment $\langle \sigma, \tau \rangle$ to all of the variables, so long as the mappings σ_x are defined at the relevant points.

We have the following characterization of truth for quantified constraint formulas.

FACT 2.6. *A quantified constraint formula Φ is true if and only if there exists a strategy σ for Φ such that for all mappings $\tau : U^\Phi \to A$, the assignment $\langle \sigma, \tau \rangle$ is defined and satisfies the constraints of Φ.*

Note that a strategy satisfying the condition of Fact 2.6 must consist only of total functions. We have defined a strategy to be a sequence of *partial* functions as we will be interested in strategies σ that need not yield an assignment $\langle \sigma, \tau \rangle$ for all τ.

2.2. Algebra. This subsection presents the algebraic background used in this paper; for more information, we refer the reader to the books [29, 32]. We begin by defining the notion of polymorphism. This notion will be used in this subsection to define further algebraic notions, and, as explained in the next subsection, will also be used throughout the paper to study the complexity of constraint languages.

DEFINITION 2.7. *An operation $f : A^k \to A$ is a* polymorphism *of a relation $R \subseteq A^m$ if for any choice of k tuples $(t_{11}, \dots, t_{1m}), \dots, (t_{k1}, \dots, t_{km}) \in R$, the tuple $(f(t_{11}, \dots, t_{k1}), \dots, f(t_{1m}, \dots, t_{km}))$ is in R. That is, applying f coordinatewise to any k tuples in R yields another tuple in R. An operation f is a polymorphism of a constraint language Γ if f is a polymorphism of all relations $R \in \Gamma$. When f is a polymorphism of a relation R (respectively, a constraint language Γ), we also say that R (respectively, Γ) is* invariant *under f.*

We now introduce the notion of a *clone*.

DEFINITION 2.8. *An operation $f : A^k \to A$ is a* projection *if there exists $i \in [k]$ such that $f(a_1, \dots, a_k) = a_i$ for all $a_1, \dots, a_k \in A$. When $f : A^n \to A$ is an arity n operation and $f_1, \dots, f_n : A^m \to A$ are arity m operations, the composition of f with $f_1, \dots, f_n$ is defined to be the arity m operation $g : A^m \to A$ such that $g(a_1, \dots, a_m) = f(f_1(a_1, \dots, a_m), \dots, f_n(a_1, \dots, a_m))$ for all $a_1, \dots, a_m \in A$.*

DEFINITION 2.9. *A* clone *on a set A is a set of finitary operations that contains all projections and is closed under composition.*

It is known that the set of polymorphisms of a constraint language is always a clone.

Next, we define the basic notion of an *algebra*.

DEFINITION 2.10. *An* algebra *is a pair $\mathbb{A} = (A, F)$ where A is a nonempty set called the* universe *and F is a set of finitary operations on A.*

We say that an algebra is *nontrivial* if its universe does not have size one. An algebra is *finite* if its universe is of finite size.

Definition 2.11. *An operation f on the set A is* idempotent *if $f(a,\dots,a)=a$ for all $a\in A$. An algebra (A,F) is* idempotent *if all operations in F are idempotent.*

Note that, in this paper, we are concerned almost exclusively with finite idempotent algebras.

Definition 2.12. *The* term operations *of an algebra (A,F) are the operations in the clone generated by F.*

We now present two standard means of constructing new algebras from an existing algebra. The first is the construction of a *subalgebra* from an algebra.

Definition 2.13. *Let $\mathbb{A}=(A,F)$ be an algebra. An algebra of the form $(B,F|_B)$, where $B\subseteq A$ is invariant under all operations $f\in F$, is called a* subalgebra *of $\mathbb{A}$.*

As a subalgebra $(B,F|_B)$ of an algebra (A,F) is determined by its universe B, we will at times use the universe B to denote the subalgebra $(B,F|_B)$. We say that a subalgebra $(B,F|_B)$ of (A,F) is a *proper* subalgebra if B is a proper subset of A, and that it is a *maximal proper* subalgebra if it is proper and the only subalgebra whose universe properly contains B is (A,F) itself.

Next, we give another construction of a new algebra from an existing algebra, namely, the construction of a *homomorphic image*.

Definition 2.14. *Let $\mathbb{A}=(A,F)$ be an algebra. A* congruence *of $\mathbb{A}$ is an equivalence relation $\theta\subseteq A\times A$ that is invariant under all operations $f\in F$. When θ is a congruence of $\mathbb{A}$, the equivalence class of θ containing $a\in A$ is denoted a^θ, and, for each operation $f\in F$, the operation f^θ given by $f^\theta(a_1^\theta,\dots,a_k^\theta)=(f(a_1,\dots,a_k))^\theta$, where k denotes the arity of f, is well defined. The set A^θ is defined as $\{a^\theta : a\in A\}$, and the set F^θ is defined as $\{f^\theta : f\in F\}$.*

Definition 2.15. *Let $\mathbb{A}=(A,F)$ be an algebra. An algebra of the form (A^θ,F^θ), where θ is a congruence of $\mathbb{A}$, is called a* homomorphic image *of $\mathbb{A}$.*

The following fact is known.

Fact 2.16. *If $\mathbb{A}$ is an idempotent algebra and θ is a congruence of $\mathbb{A}$, then each equivalence class of θ is a subalgebra of $\mathbb{A}$.*

We give a proof of Fact 2.16 for completeness.

Proof. Let $f: A^k\to A$ be an operation of $\mathbb{A}$, let B be an equivalence class of θ, and let $b_1,\dots,b_k$ be elements of B. We want to show that $f(b_1,\dots,b_k)\in B$. Fix b to be any element of B. We have $(b,b_1),\dots,(b,b_k)\in\theta$. Since θ is a congruence of $\mathbb{A}$, the operation f is a polymorphism of θ, and $(f(b,\dots,b),f(b_1,\dots,b_k))$ is an element of θ. By idempotence of f, we have $f(b,\dots,b)=b$, from which it follows that b and $f(b_1,\dots,b_k)$ are in the same θ-equivalence class and $f(b_1,\dots,b_k)\in B$. □

We now combine the subalgebra and homomorphic image constructions to define the notion of a *factor*.

Definition 2.17. *A* factor *of an algebra $\mathbb{A}$ is a homomorphic image of a subalgebra of $\mathbb{A}$.*

A known fact that we will make use of is that a factor of a factor of an algebra is a factor of the algebra.

2.3. Complexity. In this subsection, we discuss known results concerning the complexity of the problems $\mathsf{QCSP_c}(\Gamma)$ and $\mathsf{CSP_c}(\Gamma)$. We call a problem *tractable* if it is decidable in polynomial time, that is, it is in P, and the only notion of reduction we will use is many-one polynomial-time reduction.

The set of polymorphisms of a constraint language Γ has been used to study the complexity of the problems $\mathsf{CSP_c}(\Gamma)$ and $\mathsf{QCSP_c}(\Gamma)$ [22, 20, 3, 7]. For instance, the following is known.

THEOREM 2.18 (follows from [20, 3]). *Let Γ_1, Γ_2 be finite constraint languages having the same idempotent polymorphisms. Then the problems $\mathsf{CSP}_\mathsf{c}(\Gamma_1)$, $\mathsf{CSP}_\mathsf{c}(\Gamma_2)$ reduce to each other, and likewise, the problems $\mathsf{QCSP}_\mathsf{c}(\Gamma_1)$, $\mathsf{QCSP}_\mathsf{c}(\Gamma_2)$ reduce to each other.*

Intuitively, Theorem 2.18 might be taken as saying that the polymorphisms of a constraint language Γ contain all of the information needed to determine the complexity of Γ. In fact, it has been shown that when one considers the algebra having these polymorphisms as its operations, algebraic concepts such as those in the previous subsection can be employed to study complexity [7]. We will make use of this algebra and this approach.

DEFINITION 2.19. *When Γ is a constraint language over set A, the algebra $\mathbb{A}_\Gamma$ is defined to be the algebra (A, F), where F is the set of idempotent polymorphisms of Γ.*

We will use the presence of idempotent polymorphisms to prove positive results concerning $\mathsf{QCSP}_\mathsf{c}(\Gamma)$ complexity. In particular, we will make use of the following fact.

FACT 2.20. *Let $\mathcal{C}$ be a finite conjunction of constraints over the variable set $\{v_1, \ldots, v_m\}$, assume that $f : A^k \to A$ is an idempotent polymorphism of all relations in $\mathcal{C}$, and let $g_1, \ldots, g_k : \{v_1, \ldots, v_m\} \to A$ be assignments satisfying $\mathcal{C}$. Then the assignment $g : \{v_1, \ldots, v_m\} \to A$ defined by $g(v_i) = f(g_1(v_i), \ldots, g_k(v_i))$ for all v_i satisfies $\mathcal{C}$.*

This fact is known; it is implicit, for instance, in [7]. We give a proof for completeness.

Proof. Let $R(w_1, \ldots, w_n)$ be a constraint where f is a polymorphism of R and let it be satisfied by the assignments $g_1, \ldots, g_k$. It suffices to show that the assignment g also satisfies R. For each $i \in [k]$, let $g'_i : A \cup \{v_1, \ldots, v_m\} \to A$ be the extension of g_i that acts as the identity on A; likewise, let $g' : A \cup \{v_1, \ldots, v_m\} \to A$ be the extension of g that acts as the identity on A. Since each g_i satisfies the constraint, we have $(g'_i(w_1), \ldots, g'_i(w_n)) \in R$ for all $i \in [k]$. Since f is a polymorphism of R, we have that $(f(g'_1(w_1), \ldots, g'_k(w_1)), \ldots, f(g'_1(w_n), \ldots, g'_k(w_n)))$ is contained in R. We claim that this tuple is equal to $(g'(w_1), \ldots, g'(w_n))$, which would give the proof. Suppose that w_i is a variable. Then we have $f(g'_1(w_i), \ldots, g'_k(w_i)) = f(g_1(w_i), \ldots, g_k(w_i)) = g(w_i) = g'(w_i)$. Suppose that w_i is a constant. Then we have $f(g'_1(w_i), \ldots, g'_k(w_i)) = f(w_i, \ldots, w_i) = w_i = g'(w_i)$, as f is idempotent. □

To derive negative complexity results, we will make use of the following known result.

DEFINITION 2.21. *An algebra (A, F) is a* G-set *if its universe is not one-element and every operation $f \in F$ is of the form $f(x_1, \ldots, x_k) = \pi(x_i)$, where $i \in [k]$ and π is a permutation on A.*

We would like to emphasize that, in this paper, we require a G-set to be nontrivial.

THEOREM 2.22 (see [7]). *Let Γ be a constraint language. If $\mathbb{A}_\Gamma$ has a G-set as a factor, then the problem $\mathsf{CSP}_\mathsf{c}(\Gamma)$ is NP-complete, and hence the problem $\mathsf{QCSP}_\mathsf{c}(\Gamma)$ is NP-hard.*[1]

Theorem 2.22 shows that the absence of a G-set as a factor of $\mathbb{A}_\Gamma$ is necessary for the problem $\mathsf{CSP}_\mathsf{c}(\Gamma)$ to be tractable. It has been conjectured that this absence is also

[1] Note that, since $\mathsf{QCSP}_\mathsf{c}(\Gamma)$ is a more general problem than $\mathsf{CSP}_\mathsf{c}(\Gamma)$, the NP-hardness of $\mathsf{CSP}_\mathsf{c}(\Gamma)$ immediately implies the NP-hardness of $\mathsf{QCSP}_\mathsf{c}(\Gamma)$.

sufficient for $\mathsf{CSP_c}(\Gamma)$ tractability [7]. The following classification results of Bulatov confirm this conjecture for two broad classes of constraint languages.

THEOREM 2.23 (Bulatov [9]). *Let Γ be a constraint language on a three-element domain. If $\mathbb{A}_\Gamma$ does not contain a G-set as factor, then $\mathsf{CSP_c}(\Gamma)$ is in P.*

THEOREM 2.24 (Bulatov [5]). *Let Γ be a constraint language on a finite domain A containing all subsets of A. If $\mathbb{A}_\Gamma$ does not contain a G-set as factor, then $\mathsf{CSP_c}(\Gamma)$ is in P.*

In the case of a two-element algebra, there is a nice description of idempotent algebras stating that either an algebra must contain one of four particular operations or be a G-set. This description can be obtained from a classification theorem due to Post; see [2] for a presentation of this theorem. To give the description, we require a couple of definitions. When A is a two-element set, we define the *majority operation on A*, denoted by $\mathsf{majority}_A : A^3 \to A$, to be the ternary operation satisfying the identities $\mathsf{majority}_A(x, x, y) = \mathsf{majority}_A(x, y, x) = \mathsf{majority}_A(y, x, x) = x$. That is, the operation $\mathsf{majority}_A$ returns the element of A that occurs two or three times. Also, we define the *minority operation on A*, denoted by $\mathsf{minority}_A : A^3 \to A$, to be the ternary operation satisfying the identities $\mathsf{minority}_A(x, x, y) = \mathsf{minority}_A(x, y, x) = \mathsf{minority}_A(y, x, x) = y$. That is, the operation $\mathsf{minority}_A$ is idempotent, and if both elements of A occur as arguments, it returns the element that occurs once.

THEOREM 2.25 (see [2]). *Let $\mathbb{A}$ be an idempotent algebra with universe $\{0, 1\}$. Either $\mathbb{A}$ is a G-set or it contains as term operation one of the following four operations:*

- *the binary AND operation $\wedge$,*
- *the binary OR operation $\vee$,*
- *the operation $\mathsf{majority}_{\{0,1\}}$,*
- *the operation $\mathsf{minority}_{\{0,1\}}$.*

We can now readily state the classification theorem for problems $\mathsf{QCSP_c}(\Gamma)$ over a two-element domain.

THEOREM 2.26 (follows from [11] and Theorem 2.25). *Let Γ be a constraint language on the two-element domain $\{0, 1\}$. If Γ has as polymorphism one of the four operations given in the statement of Theorem 2.25, then $\mathsf{QCSP_c}(\Gamma)$ is in P. Otherwise, $\mathbb{A}_\Gamma$ is a G-set and $\mathsf{QCSP_c}(\Gamma)$ is PSPACE-complete.*

3. Collapsings.

In this section, we introduce the definitions and ideas that lie at the base of our methodology for proving QCSP complexity results. Our methodology allows one to demonstrate that for certain constraint languages Γ, the problem $\mathsf{QCSP_c}(\Gamma)$ can be reduced to the problem $\mathsf{CSP_c}(\Gamma)$. More specifically, we will show that for certain constraint languages Γ, an instance of $\mathsf{QCSP_c}(\Gamma)$ is true if and only if all instances in an ensemble of simpler instances of $\mathsf{QCSP_c}(\Gamma)$ are true. The simpler instances are derived from the original instance by instantiating all but a bounded number of the universally quantified variables with a constant; we will show that these simpler instances can be formulated as an instance of $\mathsf{CSP_c}(\Gamma)$. Their precise definition is as follows.

DEFINITION 3.1. *Let Φ be a quantified constraint formula with domain A. A quantified constraint formula Φ' is a* (j, a)-collapsing *of Φ if it can be obtained from Φ by choosing a subset U' of the universally quantified variables U^Φ with size $|U'| \leq j$ and instantiating the variables in $U^\Phi \setminus U'$ with the constant $a \in A$. A quantified constraint formula Φ' is a* j-collapsing *of Φ if for some $a \in A$, the formula Φ' is a (j, a)-collapsing of Φ. (Throughout this section, we assume $j \geq 0$.)*

By instantiating a variable v of a quantified constraint formula with a constant, we mean that the variable and its quantifier are removed from the quantifier prefix and that all instances of the variable in constraints are replaced with the constant.

Example 3.2. Consider the quantified constraint formula

$$\Phi = \forall y_1 \exists x_1 \forall y_2 \forall y_3 \exists x_2 (R_1(y_1, x_1) \wedge R_2(y_2, x_2) \wedge R_3(y_2, x_2, y_3)).$$

Suppose that its domain is $A = \{a, b\}$. There are four $(1, b)$-collapsings of Φ, corresponding to the choices $U' = \{y_1\}$, $U' = \{y_2\}$, $U' = \{y_3\}$, and $U' = \emptyset$, respectively:

$$\begin{aligned}
\forall y_1 \exists x_1 \exists x_2 (R_1(y_1, x_1) \wedge R_2(b, x_2) \wedge R_3(b, x_2, b)),\\
\exists x_1 \forall y_2 \exists x_2 (R_1(b, x_1) \wedge R_2(y_2, x_2) \wedge R_3(y_2, x_2, b)),\\
\exists x_1 \forall y_3 \exists x_2 (R_1(b, x_1) \wedge R_2(b, x_2) \wedge R_3(b, x_2, y_3)),\\
\exists x_1 \exists x_2 (R_1(b, x_1) \wedge R_2(b, x_2) \wedge R_3(b, x_2, b)).
\end{aligned}$$

Example 3.3. An example of the type of result we will prove is the following. Suppose Γ is a constraint language over the domain $\{0, 1\}$ having the boolean AND function $\wedge$ as polymorphism. Then an instance Φ of $\mathsf{QCSP_c}(\Gamma)$ is true if and only if all $(1, 1)$-collapsings of Φ are true; this is proved below in Theorem 4.3.

As discussed in section 2.1, each quantified constraint formula can be viewed as a two-player game. Recall that in this game view, a *universal player* sets the universally quantified variables, and an *existential player* sets the existentially quantified variables. The existential player attempts to satisfy the constraints of the formula, while the universal player attempts to falsify a constraint. In order for us to prove results on the (j, a)-collapsings of a formula, it will be useful for us to consider a modified version of this game where the universal player has less power: each universally quantified variable y has a subset of A associated with it, and the universal player must set each universally quantified variable to a value falling within the subset associated to y. To formalize the idea of associating subsets of A with universally quantified variables, we define the notion of *adversary*.

DEFINITION 3.4. *An* adversary $\mathcal{A}$ of length n *is a tuple* $\mathcal{A} \in (\wp(A) \setminus \{\emptyset\})^n$. *We use* $\mathcal{A}_i$ *to denote the ith coordinate of the adversary* $\mathcal{A}$*, that is,* $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_n)$.

Let us say that an adversary $\mathcal{A}$ is an adversary *for* a quantified constraint formula Φ if the length of $\mathcal{A}$ matches the number of universally quantified variables in Φ. When this is the case, the adversary $\mathcal{A}$ naturally induces the set of assignments $\mathcal{A}[\Phi] = \{\tau : U^{\Phi} \to A : \tau(y_i) \in \mathcal{A}_i \text{ for all } i \in [n]\}$. Here, we assume that $y_1, \ldots, y_n$ are the universally quantified variables of Φ, ordered according to quantifier prefix, from outside to inside.

We say that an adversary is Φ-*winnable* if in the modified game, the existential player can win, that is, if there is a strategy that can handle all assignments that the adversary gives rise to, as formalized in the following definition.

DEFINITION 3.5. *Let* Φ *be a quantified constraint formula, and let* $\mathcal{A}$ *be an adversary for* Φ. *We say that* $\mathcal{A}$ *is* Φ-winnable *if there exists a strategy* σ *for* Φ *such that for all assignments* $\tau \in \mathcal{A}[\Phi]$*, the assignment* $\langle \sigma, \tau \rangle$ *is defined and satisfies the constraints of* Φ.

We have previously given a characterization of truth for quantified constraint formulas (Fact 2.6). This characterization can be formulated in the terminology just introduced. Let A^n denote the adversary $(A, \ldots, A)$ of length n that is equal to A at all coordinates.

FACT 3.6. *The adversary* A^n *is* Φ*-winnable if and only if* Φ *is true.*

Proof. The claim is immediate from Fact 2.6 and Definition 3.5. □

In general, when B is a nonempty subset of A, we will use B^n to denote the adversary $(B, \ldots, B)$ of length n that is equal to B at all coordinates.

Fact 3.6 shows that the truth of a quantified constraint formula can be characterized in terms of an adversary. The truth of the j-collapsings of a quantified constraint formula also have an adversary-based characterization. To give this characterization, we introduce the following notation, which will be useful throughout the paper.

DEFINITION 3.7. *Let $n \geq 1$, $S \subseteq [n]$, and $C, D \subseteq A$. We use $\mathcal{A}(n, C, S, D)$ to denote the length n adversary equal to D at the coordinates in S, and equal to C at all other coordinates. That is,*

- $\mathcal{A}(n, C, S, D)_i = C$ *for* $i \in [n] \setminus S$, *and*
- $\mathcal{A}(n, C, S, D)_i = D$ *for* $i \in S$.

For $w \geq 1$, we define $\mathsf{Adv}(n, C, w, D) = \{\mathcal{A}(n, C, S, D) : S \subseteq [n], |S| \leq w\}$. That is, $\mathsf{Adv}(n, C, w, D)$ is the set of all length n adversaries that are equal to D in at most w coordinates and equal to C at all other coordinates.

Note that when using the $\mathsf{Adv}(n, C, w, D)$ notation, we will typically have $w \leq n$ and $C \subseteq D$.

Example 3.8. Suppose that $A = \{0, 1\}$. The set $\mathsf{Adv}(4, \{1\}, 1, A)$ is equal to

$$\begin{aligned}\{\,&(A, \{1\}, \{1\}, \{1\}),\\ &(\{1\}, A, \{1\}, \{1\}),\\ &(\{1\}, \{1\}, A, \{1\}),\\ &(\{1\}, \{1\}, \{1\}, A),\\ &(\{1\}, \{1\}, \{1\}, \{1\})\}.\end{aligned}$$

The following is our adversary-based characterization of the j-collapsings of a quantified constraint formula.

PROPOSITION 3.9. *Let Φ be a quantified constraint formula. The (j, a)-collapsings of Φ are true if and only if the adversaries $\mathsf{Adv}(n, \{a\}, j, A)$ are Φ-winnable. And, the j-collapsings of Φ are true if and only if the adversaries $\cup_{a \in A} \mathsf{Adv}(n, \{a\}, j, A)$ are Φ-winnable.*

Proposition 3.9 can be proved in a straightforward fashion by using Fact 3.6. We omit the proof.

Remark 3.10. *Let us say that an adversary $\mathcal{A}$ of length n is* dominated by *a second adversary $\mathcal{B}$ of length n if $\mathcal{A}_i \subseteq \mathcal{B}_i$ for all $i \in [n]$. For example, the adversary $(\{1\}, \{1\}, \{1\}, \{1\})$ in Example 3.8 is dominated by all other adversaries in that example. It is readily seen that if $\mathcal{A}$ is dominated by $\mathcal{B}$, then $\mathcal{A}$ is easier than $\mathcal{B}$ in the sense that the Φ-winnability of $\mathcal{B}$ implies the Φ-winnability of $\mathcal{A}$. Thus, the adversary $(\{1\}, \{1\}, \{1\}, \{1\})$ in Example 3.8 is in a sense superfluous as a member of $\mathsf{Adv}(4, \{1\}, 1, A)$: omitting it from this set of adversaries would not change the class of quantified constraint formulas Φ such that $\mathsf{Adv}(4, \{1\}, 1, A)$ is Φ-winnable. In general, if we defined $\mathsf{Adv}(n, C, w, D)$ as $\{\mathcal{A}(n, C, S, D) : S \subseteq [n], |S| = w\}$, demanding* exactly *$w$ coordinates of each adversary to be equal to D, then Proposition 3.9 would still hold (assuming $j \leq n$). We elected the given definition of $\mathsf{Adv}(n, C, w, D)$ because it will facilitate the presentation of certain proofs.*

To review, our goal is to show that for certain constraint languages Γ, the problem $\mathsf{QCSP_c}(\Gamma)$ is reducible to $\mathsf{CSP_c}(\Gamma)$. We will achieve this, for a constraint language, by showing that the constraint language satisfies a property that we call *collapsibility.*

DEFINITION 3.11. *A constraint language Γ is (j, a)*-collapsible *when the following holds: if all (j, a)-collapsings of an instance Φ of $\mathsf{QCSP_c}(\Gamma)$ are true, then the instance*

*Φ is true. Similarly, a constraint language Γ is j-*collapsible *when the following holds: if all j-collapsings of an instance Φ of $\mathsf{QCSP_c}(\Gamma)$ are true, then the instance Φ is true.*

When a quantified constraint formula Φ is true, then all j-collapsings of Φ are true; one way to see this is that all adversaries, and hence in particular those identified in Proposition 3.9, are dominated (in the sense of Remark 3.10) by the adversary A^n. Thus, if a constraint language Γ is j-collapsible, an instance of $\mathsf{QCSP_c}(\Gamma)$ is true *if and only if* all of its j-collapsings are true. The following proposition shows that, starting from an instance Φ of $\mathsf{QCSP_c}(\Gamma)$, the truth of the j-collapsings of Φ can be efficiently translated into an instance of $\mathsf{CSP_c}(\Gamma)$, and so the property of j-collapsibility implies a reduction from $\mathsf{QCSP_c}(\Gamma)$ to $\mathsf{CSP_c}(\Gamma)$.

PROPOSITION 3.12. *If there exists $j \geq 0$ such that the constraint language Γ is j-collapsible (or, there exists $j \geq 0$ and $a \in A$ such that the constraint language is (j, a)-collapsible), then the problem $\mathsf{QCSP_c}(\Gamma)$ reduces to (and is hence equivalent to) $\mathsf{CSP_c}(\Gamma)$.*

In order to prove Proposition 3.12, we will make use of the following lemma.

LEMMA 3.13. *For each fixed $j \geq 0$ and constraint language Γ, there is a polynomial-time algorithm that computes, given an instance Φ of $\mathsf{QCSP_c}(\Gamma)$ and a j-collapsing Φ' of Φ, an instance Φ'' of $\mathsf{CSP_c}(\Gamma)$ that is true if and only if Φ' is true.*

Proof. The idea of the proof is to create an instance Φ'' of $\mathsf{CSP_c}(\Gamma)$ where the variables are the possible output values of a strategy for Φ'. The instance Φ'' will be true if and only if there is a winning strategy for Φ'.

The variables of Φ'' are all pairs of the form (x, α), where $x \in E^{\Phi'}$ is an existentially quantified variable of Φ' and α is a mapping from $U_x^{\Phi'}$ to A. For every constraint $R(v_1, \ldots, v_m)$ and every mapping $\tau : U^{\Phi'} \to A$, we place the constraint $R(w_1, \ldots, w_m)$ in Φ'', where for each $i \in [m]$ we define $w_i = \tau(v_i)$ if v_i is universally quantified in Φ', and $w_i = (v_i, \tau|_{U_{v_i}^{\Phi'}})$ if v_i is existentially quantified in Φ'.

Suppose that $f : V^{\Phi''} \to A$ is an assignment to the variables of Φ''. It is straightforward to verify that f satisfies the constraints of Φ'' if and only if the strategy σ defined by $\sigma_x(\alpha) = f((x, \alpha))$ for all $x \in E^{\Phi'}$ and mappings $\alpha : U_x^{\Phi'} \to A$ is a winning strategy.

As Φ' has j or fewer universally quantified variables and A is fixed, the number of mappings from $U^{\Phi'}$ to A (and from $U_x^{\Phi'}$ to A for any $x \in E^{\Phi'}$) is bounded above by a constant. Having observed this, it is clear that Φ'' can be computed from Φ' in polynomial time. □

Proof of Proposition 3.12. We prove the proposition for a j-collapsible constraint language Γ; the proof is similar for a (j, a)-collapsible constraint language.

Let Φ be an instance of $\mathsf{QCSP_c}(\Gamma)$. As discussed prior to the statement of this proposition, the formula Φ is true if and only if all j-collapsings of Φ are true. We can compute all j-collapsings of Φ in polynomial time. By Lemma 3.13, each of these collapsings can be converted to an instance of $\mathsf{CSP_c}(\Gamma)$ in polynomial time. It remains to show that all of the resulting $\mathsf{CSP_c}(\Gamma)$ instances $\{\Phi_1, \ldots, \Phi_m\}$ can be formulated as a single $\mathsf{CSP_c}(\Gamma)$ instance. This can be done by renaming variables so that no two instances among $\{\Phi_1, \ldots, \Phi_m\}$ have a variable with the same name, and then creating a single instance whose constraints are all of the constraints appearing in one of the instances Φ_i. □

It is readily verified from the definitions in this section that if a constraint language is j-collapsible, then it is j'-collapsible for all $j' > j$. In addition, it is clear that the reduction from $\mathsf{QCSP_c}(\Gamma)$ to $\mathsf{CSP_c}(\Gamma)$ given by the proofs of Proposition 3.12 and Lemma 3.13 is more efficient for lower values of j than for higher values of j. Thus, j-

collapsibility results for lower values of j are preferable to such results for higher values of j, although establishing the j-collapsibility of a constraint language Γ for *any* value of j implies that there is a reduction from $\mathsf{QCSP_c}(\Gamma)$ to $\mathsf{CSP_c}(\Gamma)$ (Proposition 3.12).

4. Composing adversaries. In the previous section, we identified the properties of j-collapsibility and (j,a)-collapsibility; we showed that when a constraint language Γ has one of these properties, $\mathsf{QCSP_c}(\Gamma)$ can be reduced to $\mathsf{CSP_c}(\Gamma)$. Our goal now is to develop and deploy machinery that allows one to prove collapsibility results—results showing that a constraint language is either j- or (j,a)-collapsible. To prove a collapsibility result, by definition (see Definition 3.11), we need to prove the truth of a formula based on the truth of its collapsings. We will accomplish this in the language of adversaries. In particular, we know that the truth of collapsings can be characterized by the winnability of certain adversaries (Proposition 3.9), while the truth of a quantified constraint formula can be characterized by the winnability of the "full adversary" A^n (Fact 3.6). Thinking of the adversaries corresponding to collapsings as simple adversaries, our method will be to assume the winnability of these simple adversaries, and then derive the winnability of more and more complex adversaries, until the winnability of the full adversary is derived.

In this section, we present a notion of composition that will allow us to derive the winnability of more complex adversaries from simpler ones. After this, we give examples of how this notion of composition can be used to prove collapsibility results and hence derive positive $\mathsf{QCSP_c}(\Gamma)$ complexity results. We begin by describing our notion of composition.

Let $f : A^k \to A$ be an operation and let $\mathcal{A}, \mathcal{B}_1, \ldots, \mathcal{B}_k$ be adversaries of length n. We say that $\mathcal{A}$ is *f-composable* from $\mathcal{B}_1, \ldots, \mathcal{B}_k$, denoted $\mathcal{A} \triangleleft f(\mathcal{B}_1, \ldots, \mathcal{B}_k)$, if for all $i \in [n]$, it holds that $\mathcal{A}_i \subseteq f(\mathcal{B}_{1i}, \ldots, \mathcal{B}_{ki})$. The following key theorem allows us to derive the winnability of an adversary based on the winnability of adversaries from which it can be composed.

THEOREM 4.1. *Let Φ be a quantified constraint formula, assume that $f : A^k \to A$ is an idempotent polymorphism of all relations of Φ, and let $\mathcal{A}, \mathcal{B}_1, \ldots, \mathcal{B}_k$ be adversaries for Φ. If each of the adversaries $\mathcal{B}_1, \ldots, \mathcal{B}_k$ is Φ-winnable and $\mathcal{A} \triangleleft f(\mathcal{B}_1, \ldots, \mathcal{B}_k)$, then the adversary $\mathcal{A}$ is Φ-winnable.*

Proof. For all $j \in [k]$, let σ^j be a strategy witnessing the Φ-winnability of $\mathcal{B}_j$. Since $\mathcal{A} \triangleleft f(\mathcal{B}_1, \ldots, \mathcal{B}_k)$, there exist functions $g_i^j : \mathcal{A}_i \to \mathcal{B}_{ji}$ (with $i \in [n]$, $j \in [k]$) such that for all $i \in [n]$ and $a \in \mathcal{A}_i$, it holds that $a = f(g_i^1(a), \ldots, g_i^k(a))$. Let us denote the universal variables of Φ by $y_1, \ldots, y_n$, and assume that they are ordered according to the quantifier prefix, as in the discussion before Definition 3.5.

The idea of the proof is this. We would like to construct a strategy σ for the adversary $\mathcal{A}$ based on the strategies σ^j for the adversaries $\mathcal{B}_j$. The strategy σ simulates the strategies σ^j. Upon being given a value $a \in \mathcal{A}_i$ for the universally quantified variable y_i, the strategy σ passes the values $g_i^1(a), \ldots, g_i^k(a)$, which form an "inverse" of a under f, to the strategies $\sigma^1, \ldots, \sigma^k$. When the strategy σ needs to set an existentially quantified variable x, it takes the assignments given to x by $\sigma^1, \ldots, \sigma^k$ and applies f to them to obtain its setting. The assignment produced by σ will be equal to the assignments produced by $\sigma^1, \ldots, \sigma^k$ mapped under f (applied pointwise). We now make this idea precise.

When τ is a function in $\mathcal{A}[\Phi]$, or the restriction of a function in $\mathcal{A}[\Phi]$, for all $j \in [k]$, we define $g^j(\tau)$ to be the function with the same domain as τ and where $g^j(\tau)(y_i) = g_i^j(\tau(y_i))$ for all elements y_i of this domain. Observe that for all $\tau \in \mathcal{A}[\Phi]$ and $j \in [k]$, it holds that $g^j(\tau) \in \mathcal{B}_j[\Phi]$.

We define the strategy σ for $\mathcal{A}$ as follows. For all $x \in E^{\Phi}$, define σ_x by

$$\sigma_x(\tau) = f(\sigma_x^1(g^1(\tau)), \dots, \sigma_x^k(g^k(\tau)))$$

for all functions $\tau : U_x^{\Phi} \to A$ that arise as the restriction of a function in $\mathcal{A}[\Phi]$ to U_x^{Φ}.

We claim that for all $\tau \in \mathcal{A}[\Phi]$ and for all variables $v \in V^{\Phi}$,

$$\langle \sigma, \tau \rangle(v) = f(\langle \sigma^1, g^1(\tau)\rangle(v), \dots, \langle \sigma^k, g^k(\tau)\rangle(v)).$$

Showing this claim suffices to give the theorem by Fact 2.20 and the assumption that the σ^j are winning strategies for the $\mathcal{B}_j$.

For universal variables $y_i \in U^{\Phi}$, we have

$$\begin{aligned}\langle \sigma, \tau \rangle(y_i) &= \tau(y_i) \\ &= f(g_i^1(\tau(y_i)), \dots, g_i^k(\tau(y_i))) \\ &= f(g^1(\tau)(y_i), \dots, g^k(\tau)(y_i)) \\ &= f(\langle \sigma^1, g^1(\tau)\rangle(y_i), \dots, \langle \sigma^k, g^k(\tau)\rangle(y_i)).\end{aligned}$$

For existential variables $x \in E^{\Phi}$, we have

$$\begin{aligned}\langle \sigma, \tau \rangle(x) &= \sigma_x(\tau|_{U_x^{\Phi}}) \\ &= f(\sigma_x^1(g^1(\tau|_{U_x^{\Phi}})), \dots, \sigma_x^k(g^k(\tau|_{U_x^{\Phi}}))) \\ &= f(\sigma_x^1(g^1(\tau)|_{U_x^{\Phi}}), \dots, \sigma_x^k(g^k(\tau)|_{U_x^{\Phi}})) \\ &= f(\langle \sigma^1, g^1(\tau)\rangle(x), \dots, \langle \sigma^k, g^k(\tau)\rangle(x)). \qquad \square\end{aligned}$$

Remark 4.2. The hypothesis of Theorem 4.1 that f is an *idempotent* polymorphism can be removed if we are concerned with quantified constraint formulas where constants do *not* appear in relations. Indeed, the only place in the proof of Theorem 4.1 where the idempotence of f is used is in the appeal to Fact 2.20, and this fact holds for nonidempotent polymorphisms when the constraints do not contain constants.

Armed with Theorem 4.1, we now present three examples of collapsibility results. The first concerns the boolean AND function as a polymorphism.

THEOREM 4.3. *Let* Γ *be a constraint language over the domain* $A = \{0, 1\}$ *having the boolean AND function* $\wedge$ *as polymorphism. The constraint language* Γ *is* $(1, 1)$*-collapsible.*

Proof. Let Φ be an instance of $\mathsf{QCSP}_{\mathsf{c}}(\Gamma)$, and assume that the $(1, 1)$-collapsings of Φ are true. By Proposition 3.9, the adversaries $\mathsf{Adv}(n, \{1\}, 1, A)$ are Φ-winnable. We want to prove that the instance Φ is true; by Fact 3.6, it suffices to prove that the adversary A^n is Φ-winnable.

We prove by induction that for all $i \in [n]$, the adversary $\mathcal{A}(n, \{1\}, [i], A)$ is Φ-winnable.

For $i = 1$, this holds by hypothesis as $\mathcal{A}(n, \{1\}, [1], A) \in \mathsf{Adv}(n, \{1\}, 1, A)$.

For the induction, let us assume that the adversary $\mathcal{A}(n, \{1\}, [i], A)$ is Φ-winnable for a value $i < n$. Observe that the adversary $\mathcal{A}(n, \{1\}, \{i+1\}, A)) \in \mathsf{Adv}(n, \{1\}, 1, A)$ is Φ-winnable by hypothesis. We claim that

$$\mathcal{A}(n, \{1\}, [i+1], A) \triangleleft \wedge(\mathcal{A}(n, \{1\}, [i], A), \mathcal{A}(n, \{1\}, \{i+1\}, A)),$$

which, by appeal to Theorem 4.1, gives the induction. We verify that

$$\mathcal{A}(n,\{1\},[i+1],A)_j \subseteq \wedge(\mathcal{A}(n,\{1\},[i],A)_j, \mathcal{A}(n,\{1\},\{i+1\},A)_j)$$

for all $j \in [n]$ as follows:

- For $j \in [i]$, we have $A \subseteq \wedge(A,\{1\})$, since $a = \wedge(a,1)$ for all $a \in A$.
- For $j = i+1$, we have $A \subseteq \wedge(\{1\},A)$, since $a = \wedge(1,a)$ for all $a \in A$.
- For $j \in [n] \setminus [i+1]$, we have $\{1\} \subseteq \wedge(\{1\},\{1\})$. □

The next two examples both have a similar proof structure. We assume the winnability of the "simple" adversaries corresponding to the relevant collapsings and derive the winnability of more and more complex adversaries in an inductive manner, ultimately deriving the winnability of the "full" adversary A^n.

Definition 4.4. *A* Mal'tsev operation *is a ternary operation $m : A^3 \to A$ satisfying $m(x,x,y) = m(y,x,x) = y$ for all $x,y \in A$.*

Example 4.5. Let A be a two-element set. The operation $\mathsf{minority}_A$, defined in section 2.3, is an example of a Mal'tsev operation.

Theorem 4.6. *Let Γ be a constraint language having a Mal'tsev operation $m : A^3 \to A$ as polymorphism. The constraint language Γ is $(1,a)$-collapsible for any $a \in A$.*

Proof. Let Φ be an instance of $\mathsf{QCSP_c}(\Gamma)$, fix $a \in A$, and assume that the $(1,a)$-collapsings of Φ are true. Then by Proposition 3.9, we have that the adversaries $\mathsf{Adv}(n,\{a\},1,A)$ are Φ-winnable, and by Fact 3.6, it suffices to prove that the adversary A^n is Φ-winnable.

We prove by induction that for all $i \in [n]$, the adversary $\mathcal{A}(n,\{a\},[i],A)$ is Φ-winnable.

For $i = 1$, this holds by hypothesis as $\mathcal{A}(n,\{a\},[1],A) \in \mathsf{Adv}(n,\{a\},1,A)$.

For the induction, let us assume that the adversary $\mathcal{A}(n,\{a\},[i],A)$ is Φ-winnable for a value $i < n$. Observe that the adversary $\mathcal{A}(n,\{a\},\{i+1\},A) \in \mathsf{Adv}(n,\{a\},1,A)$ is Φ-winnable by hypothesis. We claim that

$$\mathcal{A}(n,\{a\},[i+1],A) \triangleleft m(\mathcal{A}(n,\{a\},[i],A), \mathcal{A}(n,\{a\},\emptyset,A), \mathcal{A}(n,\{a\},\{i+1\},A)),$$

which, by appeal to Theorem 4.1, gives the induction. We verify that

$$\mathcal{A}(n,\{a\},[i+1],A)_j \subseteq m(\mathcal{A}(n,\{a\},[i],A)_j, \mathcal{A}(n,\{a\},\emptyset,A)_j, \mathcal{A}(n,\{a\},\{i+1\},A)_j)$$

for all $j \in [n]$ as follows:

- For $j \in [i]$, we have $A \subseteq m(A,\{a\},\{a\})$, since $b = m(b,a,a)$ for all $b \in A$.
- For $j = i+1$, we have $A \subseteq m(\{a\},\{a\},A)$, since $b = m(a,a,b)$ for all $b \in A$.
- For $j \in [n] \setminus [i+1]$, we have $\{a\} \subseteq m(\{a\},\{a\},\{a\})$. □

Definition 4.7. *The* dual discriminator operation *on a set A is the ternary operation $d : A^3 \to A$ defined by*

$$d(x,y,z) = \begin{cases} x & \textit{if } x = y, \\ z & \textit{otherwise.} \end{cases}$$

Example 4.8. Let A be a two-element set. The operation $\mathsf{majority}_A$, defined in section 2.3, is the dual discriminator operation on A.

Theorem 4.9. *Let Γ be a constraint language having the dual discriminator operation $d : A^3 \to A$ as polymorphism. The constraint language Γ is 1-collapsible.*

Proof. Let Φ be an instance of $\mathsf{QCSP_c}(\Gamma)$, and assume that the 1-collapsings of Φ are true. By Proposition 3.9, we have that the adversaries $\cup_{a\in A}\mathsf{Adv}(n,\{a\},1,A)$ are Φ-winnable, and by Fact 3.6, it suffices to prove that the adversary A^n is Φ-winnable.

Fix two distinct elements $b, c \in A$. We prove by induction that for all $i \in [n]$, the adversaries $\mathcal{A}(n, \{b\}, [i], A)$ and $\mathcal{A}(n, \{c\}, [i], A)$ are Φ-winnable.

For $i = 1$, this holds by hypothesis as

$$\mathcal{A}(n, \{b\}, [1], A), \mathcal{A}(n, \{c\}, [1], A) \in \cup_{a \in A} \mathsf{Adv}(n, \{a\}, 1, A).$$

For the induction, let us assume that the adversaries $\mathcal{A}(n, \{b\}, [i], A)$ and $\mathcal{A}(n, \{c\}, [i], A)$ are Φ-winnable for a value $i < n$. Observe that the adversaries

$$\mathcal{A}(n, \{b\}, \{i+1\}, A), \mathcal{A}(n, \{c\}, \{i+1\}, A) \in \cup_{a \in A} \mathsf{Adv}(n, \{a\}, 1, A)$$

are Φ-winnable by hypothesis. We claim that

$$\mathcal{A}(n, \{b\}, [i+1], A) \triangleleft d(\mathcal{A}(n, \{b\}, [i], A), \mathcal{A}(n, \{c\}, [i], A), \mathcal{A}(n, \{b\}, \{i+1\}, A))$$

and

$$\mathcal{A}(n, \{c\}, [i+1], A) \triangleleft d(\mathcal{A}(n, \{c\}, [i], A), \mathcal{A}(n, \{b\}, [i], A), \mathcal{A}(n, \{c\}, \{i+1\}, A)),$$

which, by appeal to Theorem 4.1, gives the induction. We explicitly verify the first of the two claims; the second is identical, but with the roles of b and c swapped. We verify that

$$\mathcal{A}(n, \{b\}, [i+1], A)_j \subseteq d(\mathcal{A}(n, \{b\}, [i], A)_j, \mathcal{A}(n, \{c\}, [i], A)_j, \mathcal{A}(n, \{b\}, \{i+1\}, A)_j)$$

for all $j \in [n]$ as follows:

- For $j \in [i]$, we have $A \subseteq d(A, A, \{b\})$, since $a = d(a, a, b)$ for all $a \in A$.
- For $j = i + 1$, we have $A \subseteq d(\{b\}, \{c\}, A)$, since $a = d(b, c, a)$ for all $a \in A$.
- For $j \in [n] \setminus [i+1]$, we have $\{b\} \subseteq d(\{b\}, \{c\}, \{b\})$. □

The polymorphisms addressed by the preceding three theorems are all known to imply $\mathsf{CSP_c}(\Gamma)$ tractability. For instance, the following theorem is known.

THEOREM 4.10 (see [4, 14]). *Let Γ be a constraint language having a Mal'tsev operation $m : A^3 \rightarrow A$ as polymorphism. The problem $\mathsf{CSP_c}(\Gamma)$ is polynomial-time tractable.*

Similarly, the $\mathsf{CSP_c}(\Gamma)$ tractability of the binary AND operation is implied by [22, Theorem 5.13], and the tractability of the dual discriminator operation is implied by [22, Theorem 5.7]. Using these tractability results in conjunction with our collapsibility theorems, we obtain $\mathsf{QCSP_c}(\Gamma)$ tractability results. For instance, we have the following theorem.

THEOREM 4.11. *Let Γ be a constraint language having a Mal'tsev operation $m : A^3 \rightarrow A$ as polymorphism. The problem $\mathsf{QCSP_c}(\Gamma)$ is polynomial-time tractable.*

Proof. By Theorem 4.6 and Proposition 3.12, the problem $\mathsf{QCSP_c}(\Gamma)$ reduces to the problem $\mathsf{CSP_c}(\Gamma)$. By Theorem 4.10, the problem $\mathsf{CSP_c}(\Gamma)$ is polynomial-time tractable. The theorem follows. □

The collapsibility theorems that we have just given also have a consequence for the problems $\mathsf{QCSP_c}(\Gamma)$ over a two-element domain. Namely, if such a problem $\mathsf{QCSP_c}(\Gamma)$ is tractable at all, then it is 1-collapsible. The property of collapsibility thus exposes uniform structure among such tractable problems $\mathsf{QCSP_c}(\Gamma)$.

COROLLARY 4.12. *Let Γ be a constraint language over a two-element domain. If $\mathsf{QCSP_c}(\Gamma)$ is polynomial-time tractable, then Γ is 1-collapsible (assuming that P does not equal PSPACE).*

Proof. Without loss of generality, we may assume that the domain of Γ is $A = \{0, 1\}$. By Theorem 2.26, if $\mathsf{QCSP}_{\mathsf{c}}(\Gamma)$ is polynomial-time tractable, then Γ has as polymorphism one of the operations $\{\wedge, \vee, \mathsf{majority}_{\{0,1\}}, \mathsf{minority}_{\{0,1\}}\}$. If Γ has the operation $\wedge$ as polymorphism, it is 1-collapsible by Theorem 4.3. The operation $\vee$ is equivalent to $\wedge$ with the roles of 0 and 1 reversed, so if Γ has the operation $\vee$ as polymorphism, it is 1-collapsible by an identical proof. If Γ has the operation $\mathsf{majority}_{\{0,1\}}$ as polymorphism, it is 1-collapsible by Theorem 4.9, and if Γ has the operation $\mathsf{minority}_{\{0,1\}}$ as polymorphism, it is 1-collapsible by Theorem 4.6. □

5. Collapsibility. In the previous section, we justified our definition of collapsibility for constraint languages by giving examples of collapsibility results. In section 5.1, we translate this definition into the language of algebras. We give a definition of what it means for an *algebra* to be collapsible and then demonstrate that the collapsibility results on constraint languages Γ given in the previous section can in fact be interpreted as collapsibility results on their associated algebras $\mathbb{A}_\Gamma$. The advantage of having this algebraic formulation is that we are able to establish powerful and general tools for deriving collapsibility results. In each of sections 5.2, 5.3, and 5.4, we present a technique for deriving algebraic collapsibility results and illustrate its use.

Throughout this section, $\mathbb{A} = (A, F)$ denotes a finite idempotent algebra.

5.1. Collapsibility for algebras. Let us first define what it means for an algebra to be collapsible. Say that an adversary $\mathcal{A}$ is $\mathbb{A}$-*composable* from a set of adversaries $\mathcal{S}$ if there exists a term operation $f : A^k \to A$ of $\mathbb{A}$ and adversaries $\mathcal{B}_1, \ldots, \mathcal{B}_k \in \mathcal{S}$ such that $\mathcal{A}$ is f-composable from $\mathcal{B}_1, \ldots, \mathcal{B}_k$, that is, $\mathcal{A} \triangleleft f(\mathcal{B}_1, \ldots, \mathcal{B}_k)$.

DEFINITION 5.1. *An algebra $\mathbb{A} = (A, F)$ is* collapsible *with source $S \subseteq A$ and width $w \geq 0$ if for all $n \geq 1$, the adversary A^n is $\mathbb{A}$-composable from the set of adversaries $\cup_{a \in S} \mathsf{Adv}(n, \{a\}, w, A)$.*

When an algebra is collapsible, we will sometimes not state the source or width. For instance, we will say that "an algebra $\mathbb{A}$ is collapsible with source $S \subseteq A$" if there exists a width $w \geq 0$ such that $\mathbb{A}$ is collapsible with source S and width w.

We now relate this definition of collapsibility for algebras to the definition of collapsibility for constraint languages, showing that the collapsibility of the algebra $\mathbb{A}_\Gamma$ associated to a constraint language Γ implies the collapsibility of Γ itself.

PROPOSITION 5.2. *Let Γ be a constraint language. If $\mathbb{A}_\Gamma$ is collapsible with width $w \geq 0$, then Γ is w-collapsible, and hence the problem $\mathsf{QCSP}_{\mathsf{c}}(\Gamma)$ reduces to $\mathsf{CSP}_{\mathsf{c}}(\Gamma)$. Also, if $\mathbb{A}_\Gamma$ is collapsible with width $w \geq 0$ and source $\{a\}$, for some $a \in A$, then Γ is (w, a)-collapsible, and hence the problem $\mathsf{QCSP}_{\mathsf{c}}(\Gamma)$ reduces to $\mathsf{CSP}_{\mathsf{c}}(\Gamma)$.*

Proof. We prove the first part; proof of the second part is similar. Suppose that all j-collapsings of an instance Φ of $\mathsf{QCSP}_{\mathsf{c}}(\Gamma)$ are true. We want to show that the instance Φ is true. By Proposition 3.9, the adversaries $\cup_{a \in A} \mathsf{Adv}(n, \{a\}, j, A)$ are Φ-winnable. By hypothesis, the adversary A^n is $\mathbb{A}$-composable from the adversaries $\cup_{a \in A} \mathsf{Adv}(n, \{a\}, j, A)$. Thus, by Theorem 4.1, the adversary A^n is Φ-winnable. By Fact 3.6, the formula Φ is true. □

We now show that the notion of $\mathbb{A}$-composability, on which the definition of collapsibility for algebras is based, is robust in that it satisfies a certain type of transitivity.

PROPOSITION 5.3. *Let $\mathcal{S}$ and $\mathcal{S}'$ be sets of adversaries, all of the same length. If an adversary $\mathcal{A}$ is $\mathbb{A}$-composable from $\mathcal{S}'$, and all adversaries in $\mathcal{S}'$ are $\mathbb{A}$-composable from $\mathcal{S}$, then $\mathcal{A}$ is $\mathbb{A}$-composable from $\mathcal{S}$.*

Proof. By hypothesis, we have $\mathcal{A} \lhd f(\mathcal{B}'_1, \ldots, \mathcal{B}'_k)$ for a term operation f of arity k and adversaries $\mathcal{B}'_1, \ldots, \mathcal{B}'_k \in \mathcal{S}'$. And for all $i \in [k]$ we have $\mathcal{B}'_i \lhd g_i(\mathcal{B}_{(i,1)}, \ldots, \mathcal{B}_{(i,m_i)})$ for a term operation g_i of arity m_i and adversaries $\mathcal{B}_{(i,1)}, \ldots, \mathcal{B}_{(i,m_i)} \in \mathcal{S}$. Define h to be the term operation of arity $m = \sum_{i=1}^{k} m_i$ such that $h(x_1, \ldots, x_m) = f(g_1(x_1, \ldots, x_{m_1}), g_2(x_{m_1+1}, \ldots, x_{m_1+m_2}), \ldots)$. We claim that

$$\mathcal{A} \lhd h(\mathcal{B}_{(1,1)}, \ldots, \mathcal{B}_{(1,m_1)}, \mathcal{B}_{(2,1)}, \ldots, \mathcal{B}_{(2,m_2)}, \ldots).$$

For all $i \in [n]$, where n denotes the length of the adversaries under discussion, we have

$$\begin{aligned} \mathcal{A}_i &\subseteq f(\mathcal{B}'_{1i}, \ldots, \mathcal{B}'_{ki}) \\ &\subseteq f(g_1(\mathcal{B}_{(1,1)i}, \ldots, \mathcal{B}_{(1,m_1)i}), g_2(\mathcal{B}_{(2,1)i}, \ldots, \mathcal{B}_{(2,m_2)i}), \ldots) \\ &= h(\mathcal{B}_{(1,1)i}, \ldots, \mathcal{B}_{(1,m_1)i}, \mathcal{B}_{(2,1)i}, \ldots, \mathcal{B}_{(2,m_2)i}, \ldots). \end{aligned}$$

The first inclusion follows from $\mathcal{A} \lhd f(\mathcal{B}'_1, \ldots, \mathcal{B}'_k)$, the second inclusion follows from $\mathcal{B}'_i \lhd g_i(\mathcal{B}_{(i,1)}, \ldots, \mathcal{B}_{(i,m_i)})$ for all $i \in [k]$, and the equality follows from the definition of h. □

We can now show that the collapsibility results on constraint languages obtained in the previous section can in fact be interpreted as collapsibility results on algebras. For example, let us consider the proof of Theorem 4.3, which concerned constraint languages over domain $\{0,1\}$ having the boolean AND $\wedge$ as polymorphism. In that proof, we started by assuming the adversaries $\mathsf{Adv}(n, \{1\}, 1, A)$ to be Φ-winnable and repeatedly used $\wedge$-composability to derive the winnability of larger adversary sets, eventually deriving the winnability of A^n. By Proposition 5.3, that proof establishes that for any algebra $\mathbb{A}$ with universe $\{0,1\}$ having $\wedge$ as term operation, the adversary A^n is $\mathbb{A}$-composable from the set of adversaries $\mathsf{Adv}(n, \{1\}, 1, A)$. Glancing back at Definition 5.1, we can see that the following is implied by that proof.

Theorem 5.4. *An algebra with universe* $\{0,1\}$ *and having the boolean AND* $\wedge$ *as term operation is collapsible with source* $\{1\}$ *and width* 1.

In a similar manner, the following theorems can be derived from the proofs of Theorems 4.6 and 4.9.

Theorem 5.5. *An algebra with universe A having a Mal'tsev term operation is collapsible with source* $\{a\}$ *and width* 1 *for any* $a \in A$.

Theorem 5.6. *An algebra with universe A having the dual discriminator operation (over A) as term operation is collapsible with width* 1.

The notion of collapsibility for an algebra concerns the $\mathbb{A}$-composability of the adversary A^n. It will be useful to consider, more generally, the $\mathbb{A}$-composability of adversaries B^n for B a subset of A.

Definition 5.7. *A subset B of A is $\mathbb{A}$-collapsible with source $S \subseteq A$ and width $w \geq 0$ if for all $n \geq 1$, the adversary B^n is $\mathbb{A}$-composable from the set of adversaries* $\cup_{a \in S} \mathsf{Adv}(n, \{a\}, w, A)$.

Note that, in the language of this definition, an algebra $\mathbb{A}$ is collapsible if and only if its universe A is $\mathbb{A}$-collapsible.

We end this subsection with two observations.

Observation 5.8. *Every one-element subset B of A is $\mathbb{A}$-collapsible with source B and width* 0.

Observation 5.9. *If a subalgebra $\mathbb{B} = (B, F|_B)$ of $\mathbb{A} = (A, F)$ is collapsible, then B is $\mathbb{A}$-collapsible.*

5.2. Extending subsets. We now present a technique for establishing collapsibility results based on the notion of an operation *extending* a subset to a larger subset.

DEFINITION 5.10. *Let B and B' be subsets of A with $B \subseteq B'$. We say that an operation $f : A^k \to A$ (of arity $k \geq 2$)* extends B to B' *if for every $i \in [k]$, it holds that $B' \subseteq \{f(b_1, \ldots, b_k) : b_i \in B, b_j \in B'$ for $j \in [k] \setminus \{i\}\}$.*

The following is the main theorem we have concerning this notion.

THEOREM 5.11. *Let B and B' be subsets of A with $B \subseteq B'$. Suppose that B is $\mathbb{A}$-collapsible with source S and width w, and there exists an arity k term operation of $\mathbb{A}$ extending B to B'. Then B' is $\mathbb{A}$-collapsible with source S and width $w + (k-1)$.*

Before proving this theorem, we establish a lemma that will be helpful.

LEMMA 5.12. *If B is $\mathbb{A}$-collapsible with source S and width w, then for all $n \geq 1$, all adversaries in $\mathsf{Adv}(n, B, k-1, A)$ are $\mathbb{A}$-composable from $\cup_{a \in S}\mathsf{Adv}(n, \{a\}, w + (k-1), A)$.*

Proof. Let $\mathcal{A}(n, B, I, A)$ be an arbitrary adversary from $\mathsf{Adv}(n, B, k-1, A)$. By hypothesis, there exist a term operation $g : A^m \to A$ and adversaries $\mathcal{B}_1, \ldots, \mathcal{B}_m \in \cup_{a \in S}\mathsf{Adv}(n, \{a\}, w, A)$ such that $B^n \triangleleft g(\mathcal{B}_1, \ldots, \mathcal{B}_m)$. For all $i \in [m]$, let $\mathcal{B}'_i$ be the adversary such that $\mathcal{B}'_{ij} = A$ for all $j \in I$, and $\mathcal{B}'_{ij} = \mathcal{B}_{ij}$ for all $j \in [n] \setminus I$. That is, the adversary $\mathcal{B}'_i$ is equal to the adversary $\mathcal{B}_i$, except that at the coordinates in I it is equal to A. Notice that $\mathcal{B}'_1, \ldots, \mathcal{B}'_m \in \cup_{a \in S}\mathsf{Adv}(n, \{a\}, w + (k-1), A)$. It is straightforward to verify that $\mathcal{A}(n, B, I, A) \triangleleft g(\mathcal{B}'_1, \ldots, \mathcal{B}'_m)$. □

Proof of Theorem 5.11. Let $f : A^k \to A$ be an operation extending B to B', and fix an $n \geq 1$. We show that B'^n is composable from $\cup_{a \in S}\mathsf{Adv}(n, \{a\}, w + (k-1), A)$ in a sequence of compositions, which is justified by Proposition 5.3. By Lemma 5.12, it suffices to show that B'^n is $\mathbb{A}$-composable from $\mathsf{Adv}(n, B, k-1, A)$. Since $B' \subseteq A$, it suffices to show that B'^n is $\mathbb{A}$-composable from $\mathsf{Adv}(n, B, k-1, B')$.

We show by induction that for all i such that $k - 1 < i \leq n$, it holds that all adversaries in $\mathsf{Adv}(n, B, i, B')$ are $\mathbb{A}$-composable from $\mathsf{Adv}(n, B, i-1, B')$. This suffices, since $B'^n \in \mathsf{Adv}(n, B, n, B')$. Let $\mathcal{A}(n, B, I, B')$ be an arbitrary adversary from $\mathsf{Adv}(n, B, i, B')$. If $|I| < i$, then $\mathcal{A}(n, B, I, B') \in \mathsf{Adv}(n, B, i-1, B')$ and the claim is obvious. So suppose that $|I| = i$. Let $d_1, \ldots, d_k$ be k distinct elements from I, and for all $j \in [k]$, define $\mathcal{B}_j$ to be the adversary $\mathcal{A}(n, B, I \setminus \{d_j\}, A)$. We claim that $\mathcal{A}(n, B, I, B') \triangleleft f(\mathcal{B}_1, \ldots, \mathcal{B}_k)$. For all $j \notin I$, we have $\mathcal{A}(n, B, I, B')_j \subseteq f(\mathcal{B}_{1j}, \ldots, \mathcal{B}_{kj})$ because $B \subseteq f(B, \ldots, B)$, and for $j \in I$, we have $\mathcal{A}(n, B, I, B')_j \subseteq f(\mathcal{B}_{1j}, \ldots, \mathcal{B}_{kj})$ because f extends B to B'. □

A tool that will help us apply Theorem 5.11 is the following lemma.

LEMMA 5.13. *Let $f : A^k \to A$ be an operation and $a \in A$ an element. Suppose that the operation f extends $\{a\}$ to A. Then, an algebra with universe A having f as term operation is collapsible with source $\{a\}$ and width $k-1$.*

Proof. By Observation 5.8, the subset $\{a\}$ is $\mathbb{A}$-collapsible with source $\{a\}$ and width 0; by Theorem 5.11, the subset A is $\mathbb{A}$-collapsible with source $\{a\}$ and width $k-1$. □

We now derive two collapsibility results which illustrate the use of Lemma 5.13.

When $f : A \times A \to A$ is a binary operation, let us say that $u \in A$ is a *unit element* of f if for all $a \in A$, it holds that $f(u, a) = f(a, u) = a$.

THEOREM 5.14. *An algebra with universe A having an idempotent binary term operation f with unit element u is collapsible with source $\{u\}$ and width* 1.

Proof. It is straightforward to verify that f extends $\{u\}$ to A; the theorem follows from Lemma 5.13. □

DEFINITION 5.15. *A* near-unanimity operation *is an operation $f : A^k \to A$ of arity $k \geq 3$ satisfying $x = f(y, x, x, \ldots, x) = f(x, y, x, \ldots, x) = \cdots = f(x, \ldots, x, x, y)$*

for all $x, y \in A$. In words, if all but at most one of the arguments of f are equal to x, then x is the result of the operation.

THEOREM 5.16. *An algebra with universe A having a k-ary near-unanimity term operation is collapsible with source $\{a\}$ and width $k-1$, for any $a \in A$.*

Proof. It is straightforward to verify that f extends $\{a\}$ to A for any $a \in A$; the theorem follows from Lemma 5.13. □

There is overlap between the results that we just gave and the collapsibility results of the previous section.

- Theorem 5.14 implies Theorem 4.3, since the boolean AND $\wedge$ has 1 as unit element.
- Theorems 5.16 and 4.9 overlap: a dual discriminator operation is an example of a near-unanimity operation, so Theorem 5.16 shows the collapsibility of a larger class of algebras. On the other hand, while applying Theorem 5.16 to a dual discriminator operation would give collapsibility with width 2, Theorem 4.9 shows collapsibility with width 1.
- Lemma 5.13 can also be used to derive the collapsibility of an algebra having a Mal'tsev term operation, but with a width of 2—in contrast to the width 1 obtained by Theorem 4.6.

We believe that it is worth highlighting that Lemma 5.13 permits us to derive *in a uniform manner* all of the collapsibility results discussed thus far, although specialized arguments such as those given in the previous section may give tighter width bounds.

We now collect together the collapsibility results that can be stated for the two-element case and present them as a theorem; this will be useful in our study of the three-element case. Define a binary operation $f : A \times A \to A$ to be a *semilattice operation* if it is associative, commutative, and idempotent. Note that every semilattice operation over a two-element domain $A = \{a, b\}$ has a unit element, namely, the single element contained in $A \setminus \{f(a, b), f(b, a)\}$. As examples, the boolean AND operation $\wedge$ and boolean OR operation $\vee$ are the two semilattice operations over the two-element domain $\{0, 1\}$, having unit elements 1 and 0, respectively.

THEOREM 5.17. *Suppose that $\mathbb{A}$ is a two-element idempotent algebra that is not a G-set. Then $\mathbb{A}$ is collapsible with a one-element source. In particular, at least one of the following holds.*

- *The algebra $\mathbb{A}$ contains a semilattice term operation f, and the unit element of f serves as a source.*
- *The algebra $\mathbb{A}$ contains a near-unanimity term operation, and either element of A serves as a source.*
- *The algebra $\mathbb{A}$ contains the operation $\mathsf{minority}_A$ as a term operation, and either element of A serves as a source.*

Proof. That the algebra $\mathbb{A}$ contains one of the named operations follows from Theorem 2.25. The collapsibility results with the claimed sources follow from Theorems 5.14, 5.16, and 4.6, respectively. □

5.3. From subsets to subalgebras. Here, we demonstrate that subsets appearing in an adversary can be enlarged to subalgebras. This is made precise in the following proposition.

PROPOSITION 5.18. *Suppose that an adversary $(A_1, \ldots, A_n)$ is $\mathbb{A}$-composable from a set of adversaries $\mathcal{S}$, and let B_i denote the subalgebra of $\mathbb{A}$ generated by A_i for all $i \in [n]$. The adversary $(B_1, \ldots, B_n)$ is $\mathbb{A}$-composable from $\mathcal{S}$.*

Proof. Assume that the adversary $(A_1, \ldots, A_n)$ is $\mathbb{A}$-composable from $\mathcal{S}$, and that A_k is not a subalgebra of $\mathbb{A}$, where $k \in [n]$. It suffices to show that an adversary

$(A'_1, \ldots, A'_n)$, with $A'_i \supseteq A_i$ for all $i \in [n]$ and $A'_k \supsetneq A_k$, is $\mathbb{A}$-composable from $\mathcal{S}$; iteratively applying this result gives the proposition.

Since A_k is not a subalgebra of $\mathbb{A}$, there exists an operation f of $\mathbb{A}$ such that $f(A_k, \ldots, A_k)$ contains an element outside of A_k. Define A'_i by $f(A_i, \ldots, A_i)$ for all $i \in [n]$. We have

$$(A'_1, \ldots, A'_n) \triangleleft f((A_1, \ldots, A_n), \ldots, (A_1, \ldots, A_n)),$$

and so $(A'_1, \ldots, A'_n)$ is $\mathbb{A}$-composable from $\{(A_1, \ldots, A_n)\}$. By Proposition 5.3, the adversary $(A'_1, \ldots, A'_n)$ is $\mathbb{A}$-composable from $\mathcal{S}$. The adversary $(A'_1, \ldots, A'_n)$ has the desired properties: we have $A'_i \supseteq A_i$ for all $i \in [n]$ by the idempotence of f, and $A'_k = f(A_k, \ldots, A_k)$ contains an element outside of A_k. □

Proposition 5.18 has the following consequence.

PROPOSITION 5.19. *Let B be a subset of A that is $\mathbb{A}$-collapsible with source S. The subalgebra generated by B is also $\mathbb{A}$-collapsible with source S.*

Proof. Suppose that B is $\mathbb{A}$-collapsible with source S and width $w \geq 0$. For each $n \geq 1$, we have that B^n is $\mathbb{A}$-composable from $\cup_{a \in S} \mathsf{Adv}(n, \{a\}, w, A)$. Let B' denote the subalgebra of $\mathbb{A}$ generated by B. By Proposition 5.18, the adversary B'^n is $\mathbb{A}$-composable from $\cup_{a \in S} \mathsf{Adv}(n, \{a\}, w, A)$, and so B' is $\mathbb{A}$-collapsible with source S. □

As an application of Proposition 5.19, we can observe the collapsibility of a certain class of "basic" algebras. A finite algebra $\mathbb{A}$ is *simple* if any homomorphic image of $\mathbb{A}$ smaller than $\mathbb{A}$ is one-element and is *strictly simple* if it is simple and all of its proper subalgebras are one-element.

THEOREM 5.20. *Let $\mathbb{A}$ be a strictly simple finite idempotent algebra. Either $\mathbb{A}$ is a G-set or it is collapsible with a one-element source.*

Strictly simple algebras were studied in the context of CSP complexity by Bulatov, Jeavons, and Krokhin [7]. To establish Theorem 5.20, we make use of a result from that work.

THEOREM 5.21 (follows from [7, proof of Theorem 6.2]). *Let $\mathbb{A}$ be a strictly simple finite idempotent algebra. Either $\mathbb{A}$ is a G-set or it contains a term operation of one of the following types:*

- *a dual discriminator,*
- *a Mal'tsev operation,*
- *a semilattice operation $f : A \times A \to A$ of the form*

$$f(x, y) = \begin{cases} x & \text{if } x = y, \\ m & \text{otherwise} \end{cases}$$

for an element $m \in A$.

Proof of Theorem 5.20. Suppose that $\mathbb{A}$ is not a G-set. Then it must contain a term operation of one of the three types given in the statement of Theorem 5.21. If $\mathbb{A}$ contains a dual discriminator operation or a Mal'tsev operation, it is collapsible with a one-element source by Theorem 4.9 or 4.6, respectively. So suppose that $\mathbb{A}$ contains a semilattice operation f of the described form. Fix $a \in A$ to be an element distinct from m. By Observation 5.8, the set $\{a\}$ is $\mathbb{A}$-collapsible with source $\{a\}$. Now observe that f extends $\{a\}$ to $\{a, m\}$; hence, by Theorem 5.11, the set $\{a, m\}$ is $\mathbb{A}$-collapsible with source $\{a\}$. Since $\mathbb{A}$ is strictly simple, the smallest subalgebra of $\mathbb{A}$ containing $\{a, m\}$ is $\mathbb{A}$ itself, and so $\mathbb{A}$ is collapsible with source $\{a\}$ by Proposition 5.19. □

5.4. Combining subsets. We now demonstrate that the collapsibility of the union $B_1 \cup B_2$ of two subsets can be inferred from the collapsibility of the two subsets individually, along with an assumption stating that the source of one subset must fall into the other.

THEOREM 5.22. *Let B_1, B_2 be subsets of A such that*

- *B_1 is $\mathbb{A}$-collapsible with source S_1 and width w_1, and*
- *B_2 is $\mathbb{A}$-collapsible with source S_2, where $S_2 \subseteq B_1$, and width w_2.*

Then $B_1 \cup B_2$ is $\mathbb{A}$-collapsible with source S_1 and width $w_1 + w_2$.

Proof. Let $n \geq 1$. By hypothesis, we have

$$B_1^n \lhd f(\mathcal{A}(n, \{s_1\}, I_1, A), \dots, \mathcal{A}(n, \{s_k\}, I_k, A))$$

for f a term operation of $\mathbb{A}$, elements $s_1, \dots, s_k \in S_1$, and subsets $I_1, \dots, I_k \subseteq [n]$ with $|I_1|, \dots, |I_k| \leq w_1$.

We claim that every adversary contained in $\mathsf{Adv}(n, B_1, w_2, A)$ is $\mathbb{A}$-composable from $\cup_{s \in S_1} \mathsf{Adv}(n, \{s\}, w_1 + w_2, A)$. Consider an adversary $\mathcal{A}(n, B_1, I, A)$ with $I \subseteq [n]$, $|I| \leq w_2$, that is, an arbitrary adversary from $\mathsf{Adv}(n, B_1, w_2, A)$. Observe that each of the adversaries $\mathcal{A}(n, \{s_1\}, I_1 \cup I, A), \dots, \mathcal{A}(n, \{s_k\}, I_k \cup I, A)$ is in the set $\cup_{s \in S_1} \mathsf{Adv}(n, \{s\}, w_1 + w_2, A)$. We show that

$$\mathcal{A}(n, B_1, I, A) \lhd f(\mathcal{A}(n, \{s_1\}, I_1 \cup I, A), \dots, \mathcal{A}(n, \{s_k\}, I_k \cup I, A)).$$

We verify

$$\mathcal{A}(n, B_1, I, A)_j \subseteq f(\mathcal{A}(n, \{s_1\}, I_1 \cup I, A)_j, \dots, \mathcal{A}(n, \{s_k\}, I_k \cup I, A)_j)$$

for all $j \in [n]$ as follows:

- For $j \in I$, we have $A \subseteq f(A, \dots, A)$ by the idempotence of f.
- For $j \in [n] \setminus I$, it holds that $\mathcal{A}(n, \{s_i\}, I_i, A)_j = \mathcal{A}(n, \{s_i\}, I_i \cup I, A)$, and the containment follows from $B_1^n \lhd f(\mathcal{A}(n, \{s_1\}, I_1, A), \dots, \mathcal{A}(n, \{s_k\}, I_k, A))$.

We have shown that every adversary in $\mathsf{Adv}(n, B_1, w_2, A)$ is $\mathbb{A}$-composable from $\cup_{s \in S_1} \mathsf{Adv}(n, \{s\}, w_1 + w_2, A)$. By appeal to Proposition 5.3, it suffices to show that $(B_1 \cup B_2)^n$ is $\mathbb{A}$-composable from $\mathsf{Adv}(n, B_1, w_2, A)$. By hypothesis, we have $B_2^n \lhd g(\mathcal{A}(n, \{t_1\}, J_1, A), \dots, \mathcal{A}(n, \{t_m\}, J_m, A))$ for g, a term operation of $\mathbb{A}$; elements $t_1, \dots, t_m \in S_2$; and subsets $J_1, \dots, J_m \subseteq [n]$ with $|J_1|, \dots, |J_m| \leq w_2$. We claim that $(B_1 \cup B_2)^n \lhd g(\mathcal{A}(n, B_1, J_1, A), \dots, \mathcal{A}(n, B_1, J_m, A))$. We verify this as follows. For each $j \in [n]$, we have $B_1 \subseteq g(B_1, \dots, B_1) \subseteq g(\mathcal{A}(n, B_1, J_1, A)_j, \dots, \mathcal{A}(n, B_1, J_m, A)_j)$; the first containment follows from the idempotence of g, and the second containment follows from the fact that $B_1 \subseteq \mathcal{A}(n, B_1, J_i, A)_j$ for all $i \in [m]$. Also, for each $j \in [n]$, we have that B_2 is contained in

$$g(\mathcal{A}(n, \{t_1\}, J_1, A)_j, \dots, \mathcal{A}(n, \{t_m\}, J_m, A)_j),$$

which in turn is contained in

$$g(\mathcal{A}(n, B_1, J_1, A)_j, \dots, \mathcal{A}(n, B_1, J_m, A)_j);$$

the first containment was given above, and the second containment follows from the fact that $\mathcal{A}(n, \{t_i\}, J_i, A)_j \subseteq \mathcal{A}(n, B_1, J_i, A)_j$ for all $i \in [m]$, as $\{t_i\} \subseteq B_1$ for all $i \in [m]$. □

Now we give an application of Theorem 5.22. Let us say that an algebra $\mathbb{A}$ having two or more elements is *pair minimal* if for every two-element subset B of A, the smallest subalgebra of $\mathbb{A}$ containing B is minimal in that it does not properly contain any nontrivial subalgebras.

THEOREM 5.23. *If $\mathbb{A}$ is a pair minimal algebra that does not have a G-set as factor, then $\mathbb{A}$ is collapsible.*

Proof. We show that every subset $B \subseteq A$ is $\mathbb{A}$-collapsible with a one-element source, by induction on $|B|$. For subsets B with $|B| = 1$, this is immediate from Observation 5.8.

Suppose that $B \subseteq A$ is $\mathbb{A}$-collapsible with source $\{s\}$ and suppose $a \in A$. We show that there is a subset of A containing $B \cup \{a\}$ that is $\mathbb{A}$-collapsible with a one-element source. Let B' be the smallest subalgebra of $\mathbb{A}$ containing $\{s, a\}$. Because $\mathbb{A}$ is pair minimal, B' does not properly contain any nontrivial subalgebras. By Fact 2.16, it follows that any homomorphic image of B' smaller than B' is one-element, and thus B' is a strictly simple algebra. By Theorem 5.20, and Observation 5.9, the set B' is $\mathbb{A}$-collapsible with a one-element source $\{s'\}$. By Theorem 5.22 with $B_1 = B'$, $S_1 = \{s'\}$, $B_2 = B$, and $S_2 = \{s\}$, we have that $B \cup B'$ is $\mathbb{A}$-collapsible with source $\{s'\}$. □

Any algebra $\mathbb{A} = (A, F)$ where every two-element subset B of A is a subalgebra can immediately be seen to be pair minimal. We can therefore derive the following corollary from Theorem 5.23.

COROLLARY 5.24. *If $\mathbb{A}$ is an algebra that does not have a G-set as factor where every two-element subset B of A is a subalgebra of $\mathbb{A}$, then $\mathbb{A}$ is collapsible.*

From this corollary and Bulatov's theorem (Theorem 2.24), we can derive the following result.

COROLLARY 5.25. *Let Γ be a constraint language over A containing all subsets of A. Then $\mathsf{QCSP}_\mathsf{c}(\Gamma)$ is in P if the algebra $\mathbb{A}_\Gamma$ does not have a G-set as factor and is NP-hard otherwise.*

Proof. If the algebra $\mathbb{A}_\Gamma$ has a G-set as factor, then $\mathsf{QCSP}_\mathsf{c}(\Gamma)$ is NP-hard by Theorem 2.22. If the algebra $\mathbb{A}_\Gamma$ does not have a G-set as factor, then $\mathbb{A}_\Gamma$ is collapsible by Corollary 5.24; note that, since all subsets of A are in Γ, all subsets of A are subalgebras of $\mathbb{A}_\Gamma$. By Proposition 5.2, $\mathsf{QCSP}_\mathsf{c}(\Gamma)$ reduces to $\mathsf{CSP}_\mathsf{c}(\Gamma)$, and by Theorem 2.24, the problem $\mathsf{CSP}_\mathsf{c}(\Gamma)$ is in P. □

6. Sink algebras.

In this section, we identify a class of algebras that we call *sinks*. We prove a theorem which shows that any algebra must either (1) have as factor a sink or a G-set, or (2) have the desirable property of being collapsible. Now, any constraint language whose algebra has a G-set as factor is known to be hard (Theorem 2.22), and, for any constraint language Γ whose algebra $\mathbb{A}_\Gamma$ is collapsible, the problem $\mathsf{QCSP}_\mathsf{c}(\Gamma)$ has the same complexity as $\mathsf{CSP}_\mathsf{c}(\Gamma)$ (Proposition 5.2). Therefore, this theorem effectively reduces the classification of tractable problems $\mathsf{QCSP}_\mathsf{c}(\Gamma)$ to (1) the study of sinks and (2) the classification of the problems $\mathsf{CSP}_\mathsf{c}(\Gamma)$. In the next section, we will demonstrate the utility of both the definition of *sink* and the accompanying theorem, where they will be used to study the three-element case.

We need to introduce two properties of algebras before defining the notion of a *sink*.

DEFINITION 6.1. *A finite idempotent algebra $\mathbb{A}$ is* enclosed *if for every term operation $f : A^k \to A$ of $\mathbb{A}$ and maximal proper subalgebras $B_1, \ldots, B_k$ of $\mathbb{A}$, there exists a maximal proper subalgebra B such that $f(B_1, \ldots, B_k) \subseteq B$.*

Let $\mathbb{A}$ be a finite idempotent algebra. When B, B' are two maximal proper subalgebras of $\mathbb{A}$, we say that B *overlaps* B' (or, that B and B' *overlap*) if $B \cap B' \neq \emptyset$, and we say that B and B' are *connected* if there exists a sequence of maximal proper subalgebras $B_1, \ldots, B_k$ such that $B = B_1$, $B' = B_k$, and B_i overlaps B_{i+1} for all $i \in [k-1]$.

Definition 6.2. *A finite idempotent algebra $\mathbb{A}$ is* fully connected *if all pairs of maximal proper subalgebras of $\mathbb{A}$ are connected.*

We can now give the definition of a *sink algebra*.

Definition 6.3. *A finite idempotent algebra $\mathbb{A} = (A, F)$ is a* sink *if it is enclosed, is fully connected, does not contain a G-set as factor, and is not collapsible.*

We now prove the promised theorem concerning sinks. This theorem shows that an algebra is collapsible so long as it excludes two types of "forbidden" algebras: sinks and G-sets.

Theorem 6.4. *If $\mathbb{A} = (A, F)$ is a finite idempotent algebra that does not contain a sink or a G-set as factor, then $\mathbb{A}$ is collapsible.*

Proof. We prove this theorem by induction on $|A|$, the size of the universe of the algebra $\mathbb{A}$. It is trivial for $|A| = 1$, by Observation 5.8. If $\mathbb{A}$ has no nontrivial proper subalgebra, then any homomorphic image of $\mathbb{A}$ smaller than $\mathbb{A}$ must be one-element by Fact 2.16, and hence $\mathbb{A}$ is strictly simple; in this case, the theorem follows from Theorem 5.20. We therefore assume that $\mathbb{A}$ has a nontrivial proper subalgebra.

If the algebra $\mathbb{A}$ is not enclosed, then, by definition, there exists a term operation $f : A^k \to A$ of $\mathbb{A}$ and maximal proper subalgebras $B_1, \ldots, B_k$ of $\mathbb{A}$ such that the smallest subalgebra of $\mathbb{A}$ containing $f(B_1, \ldots, B_k)$ is $\mathbb{A}$ itself. By induction and Observation 5.9, we may assume that each of the sets $B_1, \ldots, B_k$ is $\mathbb{A}$-collapsible. Because $f(B_1, \ldots, B_k)^n \triangleleft f(B_1^n, \ldots, B_k^n)$ for all $n \geq 1$, we have that $f(B_1, \ldots, B_k)$ is $\mathbb{A}$-collapsible; by Proposition 5.19 with $B = f(B_1, \ldots, B_k)$, we have that A is $\mathbb{A}$-collapsible. Let us therefore assume for the rest of the proof that the algebra $\mathbb{A}$ is enclosed.

The following lemma provides some structural information concerning $\mathbb{A}$.

Lemma 6.5. *Suppose that $\mathbb{A}$ is a finite idempotent algebra that is enclosed. Then, $\mathbb{A}$ is either fully connected or its maximal proper subalgebras are disjoint.*

Proof. Let $\mathbf{B} = \{B_i\}_{i \in I}$ be a collection of maximal proper subalgebras that is closed in the sense that if a maximal proper subalgebra B is connected to a B_i in the collection, then B is in the collection. It suffices to show that $\cup_{i \in I} B_i$ is in fact a subalgebra of $\mathbb{A}$. We will establish the following claim.

Claim. Let $f : A^k \to A$ be a term operation of $\mathbb{A}$, let $B_1, \ldots, B_k$ be subalgebras from $\mathbf{B}$, and let $B'_1, \ldots, B'_k$ be subalgebras from $\mathbf{B}$ such that B_i overlaps B'_i for all $i \in [k]$. If B is a maximal proper subalgebra such that $f(B_1, \ldots, B_k) \subseteq B$, then there exists a maximal proper subalgebra B' overlapping B such that $f(B'_1, \ldots, B'_k) \subseteq B'$.

We can use the claim to show that $\cup_{i \in I} B_i$ is a subalgebra, as follows. Let $f : A^k \to A$ be an arbitrary term operation of $\mathbb{A}$. Let $i_1, \ldots, i_k \in I$ be arbitrary. Fix B_0 to be any member of $\mathbf{B}$. We have $f(B_0, \ldots, B_0) \subseteq B_0$. Since B_0 is connected to each of $B_{i_1}, \ldots, B_{i_k}$, by repeated application of the claim, we can establish that $f(B_{i_1}, \ldots, B_{i_k})$ is contained in a maximal proper subalgebra connected to B_0, implying that $f(B_{i_1}, \ldots, B_{i_k})$ is contained in a member of $\mathbf{B}$. Since the indices $i_1, \ldots, i_k$ were arbitrary, this implies that $\cup_{i \in I} B_i$ is a subalgebra of $\mathbb{A}$.

We now prove the claim. We have $f(B_1, \ldots, B_k) \subseteq B$. It follows that $f(B_1 \cap B'_1, \ldots, B_k \cap B'_k) \subseteq B$. Since $f(B_1 \cap B'_1, \ldots, B_k \cap B'_k) \subseteq f(B'_1, \ldots, B'_k)$, we have that B overlaps $f(B'_1, \ldots, B'_k)$. Since $\mathbb{A}$ is enclosed, $f(B'_1, \ldots, B'_k)$ is contained in a maximal proper subalgebra B'. From the facts that B overlaps $f(B'_1, \ldots, B'_k)$ and that $f(B'_1, \ldots, B'_k) \subseteq B'$, we have that B overlaps B', giving the claim. □

By Lemma 6.5, we have that $\mathbb{A}$ is either fully connected, or that its maximal proper subalgebras are disjoint.

Suppose that $\mathbb{A}$ is fully connected. Then we have that $\mathbb{A}$ is fully connected, is enclosed, and (by hypothesis) does not contain a G-set as factor; also by hypothesis,

$\mathbb{A}$ is not a sink, so by definition of a sink, we have that $\mathbb{A}$ is collapsible.

Suppose that the maximal proper subalgebras of $\mathbb{A}$ are disjoint. In this case, the following lemma shows the collapsibility of $\mathbb{A}$.

Lemma 6.6. *Suppose that $\mathbb{A}$ is a finite idempotent algebra that is enclosed and such that the maximal proper subalgebras of $\mathbb{A}$ are disjoint. Then the equivalence relation $\theta \subseteq A \times A$ having the maximal proper subalgebras of $\mathbb{A}$ as its equivalence classes is a congruence. And if the homomorphic image of $\mathbb{A}$ given by θ is collapsible with source S, then $\mathbb{A}$ is collapsible with any source T such that $S \subseteq \{a^\theta : a \in T\}$.*

Proof. Let θ be the equivalence relation having the maximal proper subalgebras of $\mathbb{A}$ as its equivalence classes. We first verify that θ is a congruence. We need to show that each operation $f \in F$ is a polymorphism of θ. Let f be an arity k operation from F, and suppose $(b_1, b_1'), \ldots, (b_k, b_k') \in \theta$. We have to demonstrate that $(f(b_1, \ldots, b_k), f(b_1', \ldots, b_k')) \in \theta$. By definition of θ, for all $i \in [k]$, there exists a maximal proper subalgebra B_i such that $b_i, b_i' \in B_i$. Since $\mathbb{A}$ is enclosed, there exists a maximal proper subalgebra B such that $f(B_1, \ldots, B_k) \subseteq B$. Thus $f(b_1, \ldots, b_k), f(b_1', \ldots, b_k') \in B$, and $(f(b_1, \ldots, b_k), f(b_1', \ldots, b_k')) \in \theta$.

Now assume that the homomorphic image (A^θ, F^θ) of $\mathbb{A}$ is collapsible with source S and width $w \geq 0$. Assume also that T is a subset of A such that $S \subseteq \{a^\theta : a \in T\}$. We claim that $\mathbb{A}$ is collapsible with source T and width w.

Let $n \geq 1$. There exists an operation $f \in F$ such that $(A^\theta)^n \triangleleft f^\theta(\mathcal{B}_1, \ldots, \mathcal{B}_k)$, where $\mathcal{B}_1, \ldots, \mathcal{B}_k \in \cup_{s \in S} \mathsf{Adv}(n, \{s\}, w, A^\theta)$. For all $i \in [k]$, define the element s_i to be an element of S such that $\mathcal{B}_i \in \mathsf{Adv}(n, \{s_i\}, w, A^\theta)$. Let t_i be an element in T such that $s_i = t_i^\theta$. For all $i \in [k]$, define $\mathcal{B}_i' \in \mathsf{Adv}(n, \{t_i\}, w, A)$ to be the adversary such that

$$\mathcal{B}_{ij}' = \begin{cases} A & \text{if } \mathcal{B}_{ij} = A^\theta, \\ \{t_i\} & \text{if } \mathcal{B}_{ij} = \{s_i\}. \end{cases}$$

We have that $A^\theta \subseteq f^\theta(\mathcal{B}_{1j}, \ldots, \mathcal{B}_{kj})$ for all $j \in [n]$. Observe that $\mathcal{B}_{ij} = (\mathcal{B}_{ij}')^\theta$ for all $i \in [k]$ and $j \in [n]$. Thus $A^\theta \subseteq f^\theta(\mathcal{B}_{1j}, \ldots, \mathcal{B}_{kj}) \subseteq f^\theta((\mathcal{B}_{1j}')^\theta, \ldots, (\mathcal{B}_{kj}')^\theta) \subseteq (f(\mathcal{B}_{1j}', \ldots, \mathcal{B}_{kj}'))^\theta$. Letting A_j denote $f(\mathcal{B}_{1j}', \ldots, \mathcal{B}_{kj}')$, we have $A^\theta \subseteq (A_j)^\theta$ and $(A_1, \ldots, A_n) \triangleleft f(\mathcal{B}_1', \ldots, \mathcal{B}_k')$. Since for each $j \in [n]$ we have the containment $A^\theta \subseteq (A_j)^\theta$, the set A_j contains one element from each equivalence class of θ, and hence the subalgebra generated by A_j is A itself. By Proposition 5.18, we conclude that A^n is $\mathbb{A}$-composable from $\cup_{t \in T} \mathsf{Adv}(n, \{t\}, w, A)$. □

Note that the homomorphic image of $\mathbb{A}$ given by θ is smaller than $\mathbb{A}$, as $\mathbb{A}$ has a nontrivial proper subalgebra. This homomorphic image is thus collapsible by induction, and we can apply Lemma 6.6 to derive the collapsibility of $\mathbb{A}$. □

7. The three-element case.

This section uses the ideas developed throughout this paper to investigate the complexity of $\mathsf{QCSP_c}(\Gamma)$ for constraint languages over a three-element domain. In particular, we analyze three-element sink algebras, showing that any such algebra must have a particular semilattice operation as term operation. This result will allow us to establish a classification of $\mathsf{QCSP_c}(\Gamma)$ for all constraint languages Γ that do not have the identified semilattice operation as polymorphism.

We begin by observing that there are no one- or two-element sinks.

Observation 7.1. *There are no one- or two-element sinks.*

Proof. By definition, a sink does not contain a G-set as factor and is not collapsible. Observation 5.8 implies that any one-element algebra is collapsible, and Theorem 5.17 shows that any two-element algebra not containing a G-set as factor is collapsible. □

We will show, however, that there are three-element sinks. We begin our investigation of three-element sinks by showing that any such sink must contain a particular semilattice operation.

THEOREM 7.2. *Let $\mathbb{A} = (A, F)$ be a three-element idempotent algebra that is a sink. Then, $\mathbb{A}$ has exactly two subalgebras of size two. Let c denote the common element of these two subalgebras, let a, b denote the other two elements, and let $s_{abc} : A \times A \to A$ denote the semilattice operation defined by $s_{abc}(x, y) = c$ if $x \neq y$, and $s_{abc}(x, y) = x$ if $x = y$. The algebra $\mathbb{A}$ has s_{abc} as a term operation.*

Proof. Suppose that $\mathbb{A}$ is a sink having three elements. Since a sink is fully connected by definition, each element of A must be contained in a proper subalgebra of size strictly greater than one. Hence, each element of A is contained in a subalgebra of size two, from which it follows that there are either two or three subalgebras of size two. If there are three subalgebras of size two, then by Corollary 5.24, the algebra $\mathbb{A}$ is collapsible, contradicting that $\mathbb{A}$ is a sink. We now have that $\mathbb{A}$ contains exactly two subalgebras. Let us denote these two subalgebras by $\alpha = \{a, c\}$ and $\beta = \{b, c\}$.

By Theorem 5.17, each of the subalgebras α and β are collapsible with a one-element source. If either one of them is collapsible with source $\{c\}$, then by Theorem 5.22, we have that $\alpha \cup \beta = A$ is $\mathbb{A}$-collapsible (that is, $\mathbb{A}$ is collapsible), contradicting that $\mathbb{A}$ is a sink. Hence, neither of α, β is collapsible with source $\{c\}$; by Theorem 5.17, we have that

- the subalgebra α contains a semilattice term operation s_α with a as unit element,
- the subalgebra β contains a semilattice term operation s_β with b as unit element, and
- neither of the subalgebras α, β contains a semilattice term operation with c as the unit element.

It follows that $\mathbb{A}$ contains a term operation s_a whose restriction to $\{a, c\}$ is s_α, and a term operation s_b whose restriction to $\{b, c\}$ is s_β.

Observe that, for every binary term operation f of $\mathbb{A}$, we cannot have $f(a, c) = f(c, a) = a$; otherwise the subalgebra α would contain a semilattice term operation with c as the unit element. Therefore, the restriction of f to $\{a, c\}$ is either a projection (when $f(a, c) \neq f(c, a)$) or the semilattice operation s_a (when $f(a, c) = f(c, a) = c$). Likewise, the restriction of f to $\{b, c\}$ is either a projection or the semilattice operation s_b. From these observations, it is straightforward to verify that the binary term operation $s' : A \times A \to A$ defined by $s'(x, y) = s_b(s_a(x, y), s_a(y, x))$ is equal to s_{abc}, except possibly at the points $(a, b), (b, a)$. Because $\{a, b\}$ is not a subalgebra of $\mathbb{A}$, there exists a binary term operation $r : A \times A \to A$ of $\mathbb{A}$ with $r(a, b) = c$. Using the observations again for r, it is straightforward to verify that the binary term operation $s'' : A \times A \to A$ defined by $s''(x, y) = s'(r(x, y), r(y, x))$ is equal to s_{abc}. □

By combining this result with the previous section's theorem on sinks and Bulatov's theorem (Theorem 2.23), we can state a $\mathsf{QCSP_c}(\Gamma)$ tractability classification of all constraint languages Γ, over a three-element domain, that do not have the identified semilattice polymorphism.

THEOREM 7.3. *Let Γ be a constraint language over a three-element domain D. Suppose that there is no way to label the elements of D as a, b, c such that s_{abc} (the operation defined in the statement of Theorem* 7.2*) is a polymorphism of Γ. Then, the problem $\mathsf{QCSP_c}(\Gamma)$ is in P if the algebra $\mathbb{A}_\Gamma$ does not have a G-set as factor, and NP-hard otherwise.*

Proof. If the algebra $\mathbb{A}_\Gamma$ has a G-set as factor, then it is NP-hard, by Theorem 2.22. So suppose that the algebra $\mathbb{A}_\Gamma$ does not have a G-set as factor. Because s is not a polymorphism of Γ, by Theorem 7.2, $\mathbb{A}_\Gamma$ is not a sink. Moreover, it does not contain a sink as factor, by Observation 7.1. By Theorem 6.4, the algebra $\mathbb{A}_\Gamma$ is collapsible. By Proposition 5.2 $\mathsf{QCSP}_\mathsf{c}(\Gamma)$ reduces to $\mathsf{CSP}_\mathsf{c}(\Gamma)$, and, by Bulatov's theorem (Theorem 2.23), $\mathsf{CSP}_\mathsf{c}(\Gamma)$ is in P. $\square$

We have shown that any three-element sink must have s_{abc} as a polymorphism, but we have not yet demonstrated that any three-element sinks exist! We now give a family of examples of three-element sinks, which includes the algebra $(\{a, b, c\}, \{s_{abc}\})$ as a member.

Let us introduce some concepts and terminology. Suppose that $\mathbb{A}$ is a three-element sink. We have seen (Theorem 7.2) that $\mathbb{A}$ must have two subalgebras of size two. Let us denote these two subalgebras by $\alpha = \{a, c\}$ and $\beta = \{b, c\}$. Let us say that a term operation $f : A^k \to A$ of $\mathbb{A}$ can be *realized* as an operation $g : \{\alpha, \beta\}^k \to \{\alpha, \beta\}$ if for all $S_1, \ldots, S_k \in \{\alpha, \beta\}$, it holds that $f(S_1, \ldots, S_k) \subseteq g(S_1, \ldots, S_k)$. Notice that because $\mathbb{A}$ is fully enclosed, every term operation $f : A^k \to A$ of $\mathbb{A}$ can be realized as some operation $g : \{\alpha, \beta\}^k \to \{\alpha, \beta\}$. However, it is certainly possible that a term operation of $\mathbb{A}$ can be realized as more than one operation. For example, let us consider the operation s_{abc}. The following containments hold:

$$s_{abc}(\alpha, \alpha) \subseteq \alpha,$$

$$s_{abc}(\alpha, \beta) \subseteq \{c\},$$

$$s_{abc}(\beta, \alpha) \subseteq \{c\},$$

$$s_{abc}(\beta, \beta) \subseteq \beta.$$

Since $\{c\}$ is contained in both α and β, the operation s_{abc} can be realized as any operation $g : \{\alpha, \beta\} \times \{\alpha, \beta\} \to \{\alpha, \beta\}$ such that $g(\alpha, \alpha) = \alpha$ and $g(\beta, \beta) = \beta$, that is, any idempotent binary operation on $\{\alpha, \beta\}$. Let us say that a term operation f of $\mathbb{A}$ is *$\alpha\beta$-projective* if it can be realized by an operation g on $\{\alpha, \beta\}$ that is a projection. As an example, s_{abc} can be realized as either of the two binary projections over $\{\alpha, \beta\}$ and hence is $\alpha\beta$-projective.

The following theorem gives a family of examples of three-element sinks.

THEOREM 7.4. *Let $\mathbb{A} = (A, F)$ be a three-element idempotent algebra with $A = \{a, b, c\}$ and $s_{abc} \in F$. If all operations in F are $\alpha\beta$-projective, then $\mathbb{A}$ is a sink.*

As just discussed, the operation s_{abc} is $\alpha\beta$-projective, and hence this theorem implies that the algebra $(\{a, b, c\}, \{s_{abc}\})$ is a sink.

Proof. Let $\mathbb{A}$ be an algebra of the described form. Observe that $\{a, b\}$ is not a two-element subalgebra of $\mathbb{A}$, since it is not preserved by s_{abc}. We show that $\alpha = \{a, c\}$ and $\beta = \{b, c\}$ are two-element subalgebras of $\mathbb{A}$, which implies that $\mathbb{A}$ is fully connected. Let $f : A^k \to A$ be an operation from F. Since f is $\alpha\beta$-projective, it can be realized as a projection $g : \{\alpha, \beta\}^k \to \{\alpha, \beta\}$. Observe that $f(\alpha, \ldots, \alpha) \subseteq g(\alpha, \ldots, \alpha) = \alpha$ and $f(\beta, \ldots, \beta) \subseteq g(\beta, \ldots, \beta) = \beta$, implying that α and β are both preserved by f. We have shown that α and β are subalgebras of $\mathbb{A}$.

We now prove that every term operation of $\mathbb{A}$ is $\alpha\beta$-projective. This implies that every term operation of $\mathbb{A}$ can be realized as an operation on $\{\alpha, \beta\}$, which in turn implies that $\mathbb{A}$ is enclosed, as α and β are exactly the maximal proper subalgebras of $\mathbb{A}$. First, observe that any projection $f : A^k \to A$ can be realized by the projection $g : \{\alpha, \beta\}^k \to \{\alpha, \beta\}$ that projects onto the same coordinate as f. Second, let f :

$A^n \to A$ and $f_1, \ldots, f_n : A^m \to A$ be operations that are $\alpha\beta$-projective, and let $g : \{\alpha, \beta\}^n \to \{\alpha, \beta\}$ and $g_1, \ldots, g_n : \{\alpha, \beta\}^m \to \{\alpha, \beta\}$ be projections realizing them, respectively. Then, the composition of f with $f_1, \ldots, f_n$, namely, the operation $f(f_1(x_1, \ldots, x_m), \ldots, f_n(x_1, \ldots, x_m))$, is realized by the corresponding composition $g(g_1(y_1, \ldots, y_m), \ldots, g_n(y_1, \ldots, y_m))$ which is a projection.

We now show that $\mathbb{A}$ does not contain a G-set as factor. First, observe that $\mathbb{A}$ itself is not a G-set, because s_{abc} is not essentially unary. Also, observe that the subalgebra α is not a G-set, because $s_{abc}|_\alpha$ is not essentially unary; likewise, the subalgebra β is not a G-set, because $s_{abc}|_\beta$ is not essentially unary. Any other factor of $\mathbb{A}$ of nontrivial size not isomorphic to $\mathbb{A}$, the subalgebra α, or the subalgebra β must be a homomorphic image of $\mathbb{A}$ having size two. By Fact 2.16, such a homomorphic image must arise from a congruence θ having α and $\{b\}$ as its equivalence classes, or a congruence θ having β and $\{a\}$ as its equivalence classes. In either case, the operation s^θ_{abc} is not essentially unary.

We have shown that $\mathbb{A}$ is fully connected, enclosed, and does not contain a G-set as factor. To establish that $\mathbb{A}$ is a sink, it remains to show that $\mathbb{A}$ is not collapsible. To achieve this, it suffices to demonstrate that $\mathbb{A}$ is not collapsible with source A; that is, for any $w \geq 1$, there exists $n \geq 1$ such that the adversary A^n is not $\mathbb{A}$-composable from $\cup_{a \in A} \mathsf{Adv}(n, \{a\}, w, A)$. Fix $w \geq 1$, let $n = w + 1$, and consider an adversary $(A_1, \ldots, A_n)$ such that $(A_1, \ldots, A_n)$ is $\mathbb{A}$-composable from $\cup_{a \in A} \mathsf{Adv}(n, \{a\}, w, A)$. We have $(A_1, \ldots, A_n) \triangleleft f(\mathcal{B}_1, \ldots, \mathcal{B}_k)$ for $\mathcal{B}_1, \ldots, \mathcal{B}_k \in \mathsf{Adv}(n, \{a\}, w, A)$ and f a term operation of $\mathbb{A}$. We have shown that every term operation of $\mathbb{A}$ is $\alpha\beta$-projective, so let us assume that f is realized by the projection $g : \{\alpha, \beta\}^k \to \{\alpha, \beta\}$ which projects onto its ith coordinate, where $i \in [k]$. Since $n > w$, there exists a coordinate $j \in [n]$ such that $|\mathcal{B}_{ij}| = 1$. Thus $\mathcal{B}_{ij} \subseteq \alpha$ or $\mathcal{B}_{ij} \subseteq \beta$. From this and the fact that f is realized by g, which is the projection onto the ith coordinate, we have $f(\mathcal{B}_{1j}, \ldots, \mathcal{B}_{kj}) \subseteq \alpha$ or $f(\mathcal{B}_{1j}, \ldots, \mathcal{B}_{kj}) \subseteq \beta$. From $(A_1, \ldots, A_n) \triangleleft f(\mathcal{B}_1, \ldots, \mathcal{B}_k)$, we have $A_j \subseteq f(\mathcal{B}_{1j}, \ldots, \mathcal{B}_{kj})$ and thus either $A_j \subseteq \alpha$ or $A_j \subseteq \beta$, implying that $A_j \neq A$ and hence $(A_1, \ldots, A_n) \neq A^n$. □

If it could be shown that the problem $\mathsf{QCSP_c}(\Gamma)$ is polynomial-time tractable for any constraint language Γ having s_{abc} as a polymorphism, then this result along with Theorem 7.3 would imply a classification of those problems $\mathsf{QCSP_c}(\Gamma)$ over a three-element domain that are polynomial-time tractable. Unfortunately, this is not the case: it is known that there exists a finite constraint language Γ over domain $\{a, b, c\}$ having s_{abc} as polymorphism such that $\mathsf{QCSP_c}(\Gamma)$ is coNP-hard [10]. We leave further investigation of the properties and complexity of three-element sinks as an issue for future research.

Acknowledgments. The author thanks Manuel Bodirsky, Andrei Bulatov, and Víctor Dalmau for interesting and helpful comments. The author also thanks Argimiro Arratia and Andras Salamon for their suggestions on drafts of this paper.

REFERENCES

[1] B. Aspvall, M. F. Plass, and R. E. Tarjan, *A linear-time algorithm for testing the truth of certain quantified Boolean formulas*, Inform. Process. Lett., 8 (1979), pp. 121–123.

[2] E. Böhler, N. Creignou, S. Reith, and H. Vollmer, *Playing with Boolean blocks, part* I*: Post's lattice with applications to complexity theory*, ACM SIGACT-Newsletter, 34 (4) (2003), pp. 38–52.

[3] F. Börner, A. Bulatov, A. Krokhin, and P. Jeavons, *Quantified constraints: Algorithms and complexity*, in Computer Science Logic, Lecture Notes in Comput. Sci. 2803, Springer-Verlag, Berlin, 2003, pp. 58–70.

[4] A. Bulatov, *Mal'tsev Constraints Are Tractable*, Technical report PRG-RR-02-05, Oxford University, Oxford, UK, 2002.

[5] A. Bulatov, *Tractable conservative constraint satisfaction problems*, in Proceedings of the 18th IEEE Symposium on Logic in Computer Science (LICS '03), 2003, pp. 321–330.

[6] A. Bulatov and V. Dalmau, *A simple algorithm for Mal'tsev constraints*, SIAM J. Comput., 36 (2006), pp. 16–27.

[7] A. Bulatov, P. Jeavons, and A. Krokhin, *Classifying the complexity of constraints using finite algebras*, SIAM J. Comput., 34 (2005), pp. 720–742.

[8] A. Bulatov, A. Krokhin, and P. Jeavons, *The complexity of maximal constraint languages*, in Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing, 2001, pp. 667–674.

[9] A. A. Bulatov, *A dichotomy theorem for constraint satisfaction problems on a* 3*-element set*, J. ACM, 53 (2006), pp. 66–120.

[10] H. Chen, *Quantified constraint satisfaction and* 2*-semilattice polymorphisms*, in Principles and Practice of Constraint Programming (CP 2004), Lecture Notes in Comput. Sci. 3258, Springer-Verlag, Berlin, 2004, pp. 168–181.

[11] N. Creignou, S. Khanna, and M. Sudan, *Complexity Classifications of Boolean Constraint Satisfaction Problems*, SIAM Monogr. Discrete Math. Appl. 7, SIAM, Philadelphia, 2001.

[12] V. Dalmau, *Some Dichotomy Theorems on Constant-free Quantified Boolean Formulas*, Technical report LSI-97-43-R, Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona, 1997.

[13] V. Dalmau, *A new tractable class of constraint satisfaction problems*, in Proceedings of the 6th International Symposium on Artificial Intelligence and Mathematics, 2000.

[14] V. Dalmau, *Mal'tsev Constraints Made Simple*, ECCC technical report, 2004.

[15] V. Dalmau, *Generalized majority-minority operations are tractable*, Log. Methods Comput. Sci., 2 (2006), 14 pp.

[16] H. D. Ebbinghaus, J. Flum, and W. Thomas, *Mathematical Logic*, Springer-Verlag, New York, 1984.

[17] T. Feder and M. Y. Vardi, *The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory*, SIAM J. Comput., 28 (1998), pp. 57–104.

[18] P. Hell and J. Nesetril, *Graphs and Homomorphisms*, Oxford University Press, Oxford, UK, 2004.

[19] P. Hell and J. Nešetřil, *On the complexity of H-coloring*, J. Combin. Theory Ser. B, 48 (1990), pp. 92–110.

[20] P. Jeavons, *On the algebraic structure of combinatorial problems*, Theoret. Comput. Sci., 200 (1998), pp. 185–204.

[21] P. Jeavons, D. Cohen, and M. Cooper, *Constraints, consistency, and closure*, Artificial Intelligence, 101 (1998), pp. 251–265.

[22] P. G. Jeavons, D. A. Cohen, and M. Gyssens, *Closure properties of constraints*, J. ACM, 44 (1997), pp. 527–548.

[23] M. Karpinski, H. Kleine Büning, and P. H. Schmitt, *On the computational complexity of quantified Horn clauses*, in Proceedings of the First Workshop on Computer Science Logic, Lecture Notes in Comput. Sci. 329, Springer-Verlag, Berlin, 1987, pp. 129–137.

[24] E. Kiss and M. Valeriote, *On tractability and congruence distributivity*, Log. Methods Comput. Sci., 3 (2007), 20 pp.

[25] H. Kleine Büning, M. Karpinski, and A. Flögel, *Resolution for quantified Boolean formulas*, Inform. Comput., 117 (1995), pp. 12–18.

[26] Ph. G. Kolaitis and M. Y. Vardi, *Conjunctive-query containment and constraint satisfaction*, J. Comput. System Sci., 61 (2000), pp. 302–332.

[27] B. Larose and L. Zádori, *Taylor terms, constraint satisfaction and the complexity of polynomial equations over finite algebras*, Internat. J. Algebra Comput., 16 (2006), pp. 563–581.

[28] B. Martin and F. Madelaine, *Towards a trichotomy for quantified H-coloring*, in Logical Approaches to Computational Barriers, Second Conference on Computability in Europe, Lecture Notes in Comput. Sci. 3988, Springer-Verlag, Berlin, 2006, pp. 342–352.

[29] R. N. McKenzie, G. F. McNulty, and W. F. Taylor, *Algebras, Lattices, and Varieties*, Vol. I, Wadsworth and Brooks, Monterey, CA, 1987.

[30] T. J. Schaefer, *The complexity of satisfiability problems*, in Proceedings of the Tenth Annual ACM Symposium on Theory of Computing (STOC), 1978, pp. 216–226.

[31] L. Stockmeyer and A. R. Meyer, *Word problems requiring exponential time*, in Proceedings of the 5th Annual ACM Symposium on Theory of Computing, 1973, pp. 1–10.

[32] A. Szendrei, *Clones in Universal Algebra*, Séminaire de Mathématiques Supérieures 99, University of Montreal Press, Montreal, 1986.

SPECIAL ISSUE ON FOUNDATIONS OF COMPUTER SCIENCE

This volume comprises the polished and fully refereed versions of a selection of papers presented at the Forty-Sixth Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), held in Pittsburgh, Pennsylvania, October 23–25, 2005. Unrefereed preliminary versions of the papers presented at the symposium appeared in the proceedings of the meeting, published by IEEE.

The FOCS 2005 Program Committee consisted of Ziv Bar-Yossef, Paul Beame, Ran Canetti, Irit Dinur, Ashish Goel, Venkatesan Guruswami, Sariel Har-Peled, Michael Kearns, Richard Lipton, Frank McSherry, Satish Rao, Omer Reingold, Eva Tardos, Mikkel Thorup, Berthold Voecking, John Watrous, Mihalis Yannakakis, and David Zuckerman. Of 276 extended abstracts submitted to the FOCS 2005 Program Committee, 62 were selected for presentation at the symposium. Eight of those 62 papers are included in this volume. This collection encompasses a wide variety of topics and methods in theoretical computer science, often shedding new light on entire areas with a fresh approach. The topics include algorithms, learning, property testing, combinatorics, cryptography, and distributed and quantum computing. All papers were refereed in accordance with SICOMP's stringent standards, and most were substantially updated in the process.

We take this opportunity to thank all the referees whose anonymous work has significantly contributed to the value of this volume. It was an honor to edit this special section in *SIAM Journal on Computing*.

Irit Dinur and Eva Tardos
Guest Editors

SIAM J. COMPUT.
Vol. 37, No. 6, pp. 1703–1727

A CHARACTERIZATION OF THE (NATURAL) GRAPH PROPERTIES TESTABLE WITH ONE-SIDED ERROR*

NOGA ALON† AND ASAF SHAPIRA‡

Abstract. The problem of characterizing all the testable graph properties is considered by many to be the most important open problem in the area of property testing. Our main result in this paper is a solution of an important special case of this general problem: Call a property tester *oblivious* if its decisions are independent of the size of the input graph. We show that a graph property $\mathcal{P}$ has an oblivious one-sided error tester *if and only if* $\mathcal{P}$ is *semihereditary.* We stress that any "natural" property that can be tested (either with one-sided or with two-sided error) can be tested by an oblivious tester. In particular, all the testers studied thus far in the literature were oblivious. Our main result can thus be considered as a precise characterization of the natural graph properties, which are testable with one-sided error. One of the main technical contributions of this paper is in showing that any hereditary graph property can be tested with one-sided error. This general result contains as a special case all the previous results about testing graph properties with one-sided error. More importantly, as a special case of our main result, we infer that some of the most well-studied graph properties, both in graph theory and computer science, are testable with one-sided error. Some of these properties are the well-known graph properties of being perfect, chordal, interval, comparability, permutation, and more. None of these properties was previously known to be testable.

Key words. property testing, hereditary properties, one-sided error, regularity lemma

AMS subject classifications. 05D99, 05C85, 68W20, 68W25

DOI. 10.1137/06064888X

1. Introduction.

1.1. Definitions and background.

The meta problem in the area of property testing is the following: Given a combinatorial structure S, distinguish if S satisfies some property $\mathcal{P}$ or if S is ϵ-far from satisfying $\mathcal{P}$, where S is said to be ϵ-far from satisfying $\mathcal{P}$ if an ϵ-fraction of its representation should be modified in order to make S satisfy $\mathcal{P}$. The main goal is to design randomized algorithms, which look at a very small portion of the input, and using this information distinguish with high probability between the above two cases. Such algorithms are called *property testers* or simply *testers* for the property $\mathcal{P}$. Preferably, a tester should look at a portion of the input whose size is a function of ϵ only. Blum, Luby, and Rubinfeld [14] were the first to formulate a question of this type, and the general notion of property testing was first formulated by Rubinfeld and Sudan [36], who were motivated in studying various algebraic properties such as linearity of functions.

*Received by the editors January 3, 2006; accepted for publication (in revised form) January 8, 2007; published electronically March 26, 2008. A preliminary version of this paper appeared in *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science* (FOCS 2005), Pittsburgh, PA, 2005, pp. 429–438.

http://www.siam.org/journals/sicomp/37-6/64888.html

†Schools of Mathematics and Computer Science, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv 69978, Israel, and IAS, Princeton, NJ 08540 (nogaa@tau.ac.il). This author's research was supported in part by a grant from the Israel Science Foundation, by the Hermann Minkowski Minerva Center for Geometry at Tel Aviv University, and by the Von Neumann Fund.

‡School of Computer Science, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv 69978, Israel (asafico@tau.ac.il). This work forms part of the author's Ph.D. thesis. This author's research was supported by a Charles Clore Foundation Fellowship and an IBM Ph.D. Fellowship.

The main focus of this paper is in testing properties of graphs, where a property of graphs is simply a family of graphs closed under isomorphism. Throughout the paper we consider only decidable graph properties, that is, properties $\mathcal{P}$ for which it is possible to decide in finite time whether an input graph G satisfies $\mathcal{P}$. In this case a graph G is said to be ϵ-far from satisfying a property $\mathcal{P}$ if one needs to add/delete at least ϵn^2 edges to/from G in order to turn it into a graph satisfying $\mathcal{P}$. A tester for $\mathcal{P}$ should distinguish with high probability, say 2/3, between the case that G satisfies $\mathcal{P}$ from the case that G is ϵ-far from satisfying $\mathcal{P}$. Here we assume that the tester can query some oracle, whether a pair of vertices, i and j, are adjacent in the input graph G. In what follows we will say that a tester for a graph property $\mathcal{P}$ has *one-sided error* if it accepts any graph satisfying $\mathcal{P}$ with probability 1 (and still rejects those that are ϵ-far with probability at least 2/3). If the tester may reject graphs satisfying $\mathcal{P}$ with nonzero probability, then it is said to have *two-sided error*.

The study of the notion of testability for combinatorial structures, and mainly for labelled graphs, was introduced in the seminal paper of Goldreich, Goldwasser, and Ron [22], where it was shown that many natural graph properties such as k-colorability, having a large clique and having a large cut, have a tester, whose *query complexity*, that is, the number of oracle queries of type "does (i, j) belong to $E(G)$," can be upper bounded by a function that depends only on ϵ and is independent of the size of the input (the dependence on ϵ may vary between properties). In this paper we will say that properties having such efficient testers, that is, whose query complexity can be upper bounded by a function of ϵ only, are simply *testable*. Following [22], many other graph properties were shown to be testable, while others were shown to be nontestable. We note that the model of graph property testing that we study here is the so-called *dense graph* model as defined in [22]. Graph property testing has also been studied in other models, such as the *bounded degree* model [23] and the *general density* model [30].

1.2. Related work. The most interesting results in property testing are those that show that large families of problems are testable. The main result of [22] states that a certain abstract graph partition problem, which includes as special cases k-colorability, having a large cut and having a large clique, is testable. The authors of [24] gave a characterization of the partition problems discussed in [22] that are testable with one-sided error. In [4], a logical characterization of a family of testable graph properties was obtained. According to this characterization, every first-order graph property of type $\exists\forall$ (see subsection 2.3.2) is testable, while there are first-order graph properties of type $\forall\exists$ that are not testable. These results were extended in [16]. There are also several general testability and nontestability results in other areas besides testing graph properties. In [5] it is proved that every regular language is testable. This result was extended to any read-once branching program in [29]. On the other hand, it was proved in [19] that there are read-twice branching programs that are not testable. The main result of [8] states that any constraint satisfaction problem is testable.

With this abundance of general testability results, a natural question is, What makes a combinatorial property testable? As graphs are the most well-studied combinatorial structures in the theory of computation, it is natural to consider the problem of characterizing the testable graph properties as the most important open problem in the area of property testing. Regretfully, though, finding such a characterization remains a challenging open problem. The main result of this paper, Theorem 2, resolves an important natural special case of this open problem, which concerns property testers with one-sided error. For additional results and references on graph property testing as well as on testing properties of other combinatorial structures, the reader is referred to [17], [21], [35], and [12].

2. The new results.

2.1. The main technical result and its immediate applications. A graph property is *hereditary* if it is closed under removal of vertices (and *not* necessarily under removal of edges). Equivalently, such properties are closed under taking induced subgraphs. The main technical result of this paper is the following.

Theorem 1 (main technical result). *Every hereditary graph property is testable with one-sided error.*

As we will see later, the testing algorithms we design for a given hereditary property $\mathcal{P}$ simply sample a set of vertices S and accept if and only if the graph induced by S satisfies $\mathcal{P}$. This immediately implies that these testers have one-sided error. Of course, the main difficulty lies in proving that if the input is ϵ-far from satisfying $\mathcal{P}$, then the graph induced by a large enough S (but only large enough as a function of ϵ) will not satisfy $\mathcal{P}$ with high probability.

We note that besides certain partition properties such as having a large cut and having a large clique, which were proved to be testable with two-sided error in [22], essentially any graph property that was studied in the literature is hereditary. Thus, Theorem 1 combined with the graph partition problems of [22] implies the testability of (nearly) any natural graph property. To demonstrate the generality of Theorem 1, we use it to infer that many graph properties, which prior to this paper were not known to be testable, are in fact testable with one-sided error. These include the following hereditary properties:

- *Perfect graphs.* A graph G is perfect if for every *induced* subgraph of G, denoted G', the chromatic number of G' equals the size of the largest clique in G'.
- *Chordal graphs.* A graph is chordal if it contains no *induced* cycle of length at least 4.
- *Interval graphs.* A graph G on n vertices is an interval graph if there are closed intervals on the real line $I_1, \ldots, I_n$ such that $(i, j) \in E(G)$ if and only if $I_i \cap I_j \neq \emptyset$.
- *Circular-arc graphs.* A graph G on n vertices is a circular-arc graph if there are closed intervals on a cycle $I_1, \ldots, I_n$ such that $(i, j) \in E(G)$ if and only if $I_i \cap I_j \neq \emptyset$.
- *Comparability graphs.* A graph G is a comparability graph if its edges can be oriented such that if there is a directed edge from i to j and from j to k, then there is one from i to k.
- *Permutation graphs.* A graph G on n vertices is a permutation graph if there is a permutation σ of $\{1, \ldots, n\}$ such that $(i, j) \in E(G)$ if and only if (i, j) is an inversion under σ.
- *Asteroidal triple-free graphs:* G is asteroidal triple-free if it contains no independent set of three vertices such that each pair is joined by a path that avoids the neighborhood of the third.
- *Split graphs.* G is a split graph if $V(G)$ can be split into a clique and an independent set.

Another abstract family of hereditary graph properties, which have been extensively studied, are the so-called *intersection graph properties.* In this case we fix a certain “type” T of sets and say that a graph G on n vertices has the intersection property defined by T, if there are n sets $S_1, \ldots, S_n$ of type T, such that vertices i and j are connected in G if and only if $S_i \cap S_j \neq \emptyset$. For example, the property of being a d-Box (see [15] and its references) is obtained by letting the "type" of the sets be axis parallel boxes in R^d. See the monograph [28] for more information and examples of intersection graph properties.

It is clear that the above surveyed properties are some of the most well-studied properties in graph theory as well as in theoretical and applied computer science. These properties also arise naturally in chemistry, biology, social sciences, statistics, etc. See [25], [28], [32] and their references, where other hereditary properties and their applications are also discussed.

To further convey the reader of the power of Theorem 1, we mention that Lemma 4.2, which is the main result needed to obtain Theorem 1, immediately implies, for example, that for every ϵ there is $c = c(\epsilon)$, such that if a graph G is ϵ-far from being chordal, then G contains an *induced* cycle of length at most c, and that similar results hold for any other hereditary property. This is nontrivial, as it is not clear a priori that there is no graph that is, say, $\frac{1}{100}$-far from being chordal and yet contains only induced cycles of length at least, say, $\Omega(\log n)$. Put another way, if G has the property that all its induced subgraphs of size $c = c(\epsilon)$ are chordal, then G is ϵ-close to being chordal. This gives a strong connection between the *local* properties of a graph and its *global* properties. In fact, we can show that an analogous result holds for any graph property; see Theorem 6.

2.2. The main result: Oblivious testing with one-sided error. By a result of [4] and [24], it is possible to assume that a property tester works by making its queries nonadaptively. In other words, the tester first picks a random subset of vertices S, then queries all pairs (i, j) $i, j \in S$, and then continues without making additional queries. Inspecting previous results on property testing motivates the following notion of a slightly more restricted tester, which works while being "oblivious" to the size of the input.[1]

DEFINITION 2.1 (oblivious tester). *A tester (one-sided or two-sided) for a property $\mathcal{P}$ is said to be* oblivious *if it works as follows: Given ϵ, the tester computes an integer $Q = Q(\epsilon)$ and asks an oracle for a subgraph induced by a set of vertices S of size Q, where the oracle chooses S randomly and uniformly from the vertices of the input graph. If Q is larger than the size of the input graph, then the oracle returns the entire graph. The tester then accepts or rejects (possibly randomly) according to ϵ and the graph induced by S.*

Note that by insisting that the oracle chooses the set of vertices S, an oblivious tester indeed operates without knowing the size of the input, because if the tester had to choose S, then it would have to know the size of the input graph in order to specify a vertex of the graph. We believe that the above definition captures the essence of property testing in the dense graph model: all the testers that have been analyzed in this model were in fact oblivious or could trivially be turned into oblivious testers. Even the testers for properties such as having an independent set of size $\frac{1}{2}n$ or a cut with at least $\frac{1}{8}n^2$ edges (see [22]), whose definition involves the size of the graph, have oblivious testers. The reason is simply that these properties can easily be expressed without using the size of the graph. For example, in order to test if a graph has a cut with at least $\frac{1}{8}n^2$ edges one can sample some $Q = Q(\epsilon)$ vertices and accept the input

[1] The tester implied by the results of [24] and [4] may use the size of the input in order to determine the query complexity and in order to make its decisions.

if and only if the graph induced on the sample has a cut of size at least $(\frac{1}{8} - \frac{\epsilon}{2})Q^2$ (of course, one needs to prove that this sampling scheme indeed works; see [22]). Another family of graph properties for which we can confine ourselves to oblivious testers is the family of hereditary properties, which is shown to be testable by an oblivious tester in the present paper. We finally note that most "applications" of property testing (see [17] and [35]) involve testing properties of huge networks such as the Internet, whose size is unknown anyway.

Observe that there are two restrictions that the above definition imposes on an oblivious tester. The first is that it cannot use the size of the input in order to determine the size, Q, of the sample of vertices. In other words, Q is only a function of ϵ and not a function of ϵ and n. The reader should note that a tester for a testable graph property (as defined in section 1) may have a query complexity that is *bounded* by a function of ϵ but one that *depends* on the size of the graph (e.g., $Q(\epsilon, n) = 1/\epsilon + (-1)^n$). Though this seems like an annoying technicality, it was proved in [10] that this subtlety may have nontrivial ramifications. The second, seemingly more severe, restriction on an oblivious tester is that it cannot use the size of the input in order to make its decisions after the subgraph induced on the set S of Q vertices has been obtained. One can easily "cook" graph properties that cannot be tested by an oblivious tester. However, these properties are somewhat nonnatural. One example out of many is the following property, which we denote by $\mathcal{P}'$: A graph on an even number of vertices satisfies $\mathcal{P}'$ if and only if it is bipartite, while a graph on an odd number of vertices satisfies $\mathcal{P}'$ if and only if it is triangle-free. A tester for $\mathcal{P}'$ clearly must use the size of the input in order to make its decision regarding the graph induced by the sample.

We now turn to the main result of this paper, which gives a characterization of the graph properties that can be tested with one-sided error by an oblivious tester. Intuitively, in order to test a property with one-sided error the tester must "find" some kind of proof that the input does not satisfy the property. Of course the graph itself is such a proof, but as we confine ourselves to testers whose number of queries is independent of the size of the input, the tester must find a *small* proof of this fact. For hereditary properties, such proofs exists and are, in fact, (relatively) abundant. These are small induced subgraphs that do not satisfy the property. In fact, this is the main idea behind our algorithm for testing hereditary properties. See Lemma 4.2, which is the main technical result of this paper.

A natural question is if other nonhereditary properties have such small proofs. For example, having a clique of size $\frac{1}{2}n$ obviously does not have such small proofs. The reason is that for any fixed graph C there are graphs that contain C as an induced subgraph and have a clique of size $\frac{1}{2}n$, and graphs that contain C as an induced subgraph and are far from having a clique of size $\frac{1}{2}n$. In [24] it was shown that when considering the partition problems of [22], which contain the clique property as a special case, the nonhereditary partition properties cannot be tested with one-sided error. For general properties, the situation is much more involved. However, considering only oblivious testers enables us to precisely characterize the graph properties, which are testable with one-sided error. To state this characterization, we need the following definition.

DEFINITION 2.2 (semihereditary). *A graph property $\mathcal{P}$ is semihereditary if there exists a hereditary graph property $\mathcal{H}$ such that the following hold:*

1. *Any graph satisfying $\mathcal{P}$ also satisfies $\mathcal{H}$.*
2. *There is a function $M : (0, 1) \mapsto \mathbb{N}$, such that any graph G of size at least*

$M(\epsilon)$, which is ϵ-far from satisfying $\mathcal{P}$, contains an induced subgraph that does not satisfy $\mathcal{H}$.

COMMENT 2.3. *As $\mathcal{H}$ is hereditary, a simpler condition equivalent to item 2 above is that G itself does not satisfy $\mathcal{H}$. However, the statement above will be more convenient for the proof of Theorem 2.*

Clearly, any hereditary graph property $\mathcal{P}$ is also semihereditary because we can take $\mathcal{H}$ in the above definition to be $\mathcal{P}$ itself. In simple words, a semihereditary $\mathcal{P}$ is obtained by taking a hereditary graph property $\mathcal{H}$ and removing from it a (possibly infinite) set of graphs. This means that the first item in Definition 2.2 is satisfied. As there are graphs not satisfying $\mathcal{P}$ that do satisfy $\mathcal{H}$, these graphs do not contain any induced subgraph that does not satisfy $\mathcal{H}$ (because $\mathcal{H}$ is hereditary). The only restriction, which is needed in order to get item 2 in Definition 2.2, is that $\mathcal{P}$ will be such that for any $\epsilon > 0$ there will be only finitely many graphs that are ϵ-far from satisfying it and yet contain no induced subgraph that does not satisfy $\mathcal{H}$.

We are now ready to state the main result of this paper.

THEOREM 2 (main result). *A graph property $\mathcal{P}$ has an oblivious one-sided error tester if and only if $\mathcal{P}$ is semihereditary.*

Returning to the graph property $\mathcal{P}'$ discussed above, note that by Theorem 1 this property, which is not semihereditary, can be tested with one-sided error by a nonoblivious tester. Therefore, it is not the case that a graph property is testable if and only if it is semihereditary. However, if we disregard this and other nonnatural graph properties, then we may assume that in order to test them we can confine ourselves to oblivious testers. Theorem 2 can thus be considered as a *precise characterization* of the "natural" graph properties, which are testable with one-sided error. We believe that it may be very interesting to further study property testing via the framework of oblivious testers; see section 7.

Theorems 1 and 2 suggest many questions, some of which we discuss and resolve in the following subsections, while others are discussed in section 7 and are left as interesting open problems.

2.3. Additional results.

2.3.1. On the (im)possibility of relaxing the notion of property testing. Theorem 1 implies that any hereditary graph property is testable when one uses the standard notion of ϵ-far as defined in section 1. Suppose we forbid addition of edges and define a graph G on n vertices to be ϵ-far$_{del}$ from satisfying property $\mathcal{P}$ if one needs to delete from G at least ϵn^2 edges in order to turn it into a graph satisfying $\mathcal{P}$. We say that property $\mathcal{P}$ is testable$_{del}$ if there is a tester for distinguishing between graphs satisfying $\mathcal{P}$ from those that are ϵ-far$_{del}$ from satisfying it, whose number of queries can be upper bounded by a function of ϵ. A natural question is, Which graph properties are testable$_{del}$? Obviously, any hereditary property which is also closed under removal of edges (such as k-colorability) is testable$_{del}$, as in these cases being ϵ-far$_{del}$ is equivalent to ϵ-far. The following theorem is a sharp contrast to Theorems 1 and 2.

THEOREM 3. *For any hereditary property $\mathcal{P}$, which is not satisfied by all graphs, and is satisfied by any complete graph, there is a constant $\delta = \delta(\mathcal{P}) > 0$ such that testing$_{del}$ property $\mathcal{P}$ (even with two-sided error) requires n^{δ} queries.*

Note that any natural hereditary property, such as any of those discussed in subsection 2.1, is satisfied by any complete graph, and thus the above result applies to these properties. We briefly mention that we can also prove a similar statement when one allows only edge additions. See section 6.

2.3.2. Unbounded first-order graph properties. A first-order graph property is one involving the boolean operators $\wedge, \vee, \neg$, the $\forall, \exists$ quantifiers, the equality operator $=$, and the adjacency relation $\sim$. For example, the triangle-freeness property can be written as $\forall\ v_1, v_2, v_3 \neg(v_1 \sim v_2 \wedge v_2 \sim v_3 \wedge v_1 \sim v_3)$. The main result of [4] states that every first-order graph property without quantification $\forall\exists$ is testable (possibly with two-sided error). The main tool in [4] was a theorem stating that any hereditary graph property which is expressible in terms of a *finite* family of forbidden induced subgraphs is testable. Theorem 1 is a powerful extension of this result, as it allows the family of forbidden induced subgraphs to be infinite. One may thus ask whether Theorem 1 can be used in order to extend the result of [4]. Theorem 4 below gives a positive answer to this question. To state this extension, we need the following definition.

DEFINITION 2.4 (unbounded first-order properties of type $\exists\forall$). *An unbounded first-order graph property of type $\exists\forall$ is of the form*

$$\exists x_1, \ldots, x_t \bigwedge_{i=1}^{\infty} \forall y_1, \ldots, y_i\ A_i(x_1, \ldots, x_t, y_1, \ldots, y_i), \tag{1}$$

where each $A_i(x_1, \ldots, x_t, y_1, \ldots, y_i)$ is a quantifier-free first-order expression.

The main result of [4] states that any graph property that can be expressed as above while using a *single* relation A_i is testable. Using the main techniques of this paper, we can extend this to expressions containing *infinitely* many expressions A_i.

THEOREM 4. *Every graph property describable by an unbounded first-order graph property of type $\exists\forall$ is testable (possibly with two-sided error).*

It should be noted that it is proved in [4] that there are first-order graph properties with alternation of type $\forall\exists$ which are not testable, and thus Theorem 4 is in some sense best possible.

2.3.3. *AND*ing hereditary graph properties. We next describe a consequence of Theorem 1 (in fact, of the main step of proving Theorem 1), which does not assert the testability of some graph property, but rather one that may be useful in the general study of graph property testing. Suppose $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_1, \ldots, \}$ is a (possibly infinite) set of monotone graph properties, that is, properties that are closed under removal of vertices and edges. It was proved in [11] that in this case there is a function $\delta : (0, 1) \mapsto (0, 1)$ such that if a graph G is ϵ-far from satisfying all the properties of $\mathcal{P}$, then for some i it is also $\delta(\epsilon)$-far from satisfying $\mathcal{P}_i$. Note that this statement is nontrivial only when $\mathcal{P}$ is infinite, as if $\mathcal{P}$ contains k properties, we can clearly take $\delta(\epsilon) = \epsilon/k$ (in fact, to get the case of finite k the properties need only to be closed under removal of edges). Consider now the case when the properties are assumed to be hereditary properties which are not necessarily monotone. Now it is not at all clear that a similar statement holds even for $k = 2$, as modifying a graph in order to turn it into a graph satisfying $\mathcal{P}_1$ may increase its distance from satisfying $\mathcal{P}_2$. Using Theorem 1, we can show that a similar result holds even for infinite sets of properties.

THEOREM 5. *For any (possibly infinite) set of hereditary graph properties $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots\}$, there is a function $\delta_{\mathcal{P}} : (0, 1) \mapsto (0, 1)$ with the following property: If a graph G is ϵ-far from satisfying all the properties of $\mathcal{P}$, then for some i the graph G is $\delta_{\mathcal{P}}(\epsilon)$-far from satisfying $\mathcal{P}_i$.*

2.3.4. An extremal result for all graph property. Confirming a conjecture of Erdős, it was shown in [33] that if a graph is ϵ-far from being k-colorable, then it contains a non-k-colorable subgraph of size that depends only on ϵ. In [11] this result was extended to any monotone graph property. As we have alluded to in subsection 2.1, the main technical result of this paper, Lemma 4.2, immediately implies that this result can be extended to any hereditary graph property. In fact, we can show that a similar result holds for *any* graph property.

THEOREM 6. *For every graph property $\mathcal{P}$, there is a function $W_{\mathcal{P}}(\epsilon)$ with the following property: If G is ϵ-far from satisfying $\mathcal{P}$, then G contains an induced subgraph of size at most $W_{\mathcal{P}}(\epsilon)$ which does not satisfy $\mathcal{P}$.*

2.4. Comparison to previous results. We next survey the previous results on graph property testing, which were shown to be testable with one-sided error. As all these properties are hereditary, their testability with one-sided error follows as a special case of Theorem 1.

- ***H*-*free.*** For every fixed graph H, let $\mathcal{P}_H$ be the property of not containing a copy of H, and let $\mathcal{P}^*_H$ be the property of not containing an induced copy of H. The property $\mathcal{P}_H$ was (implicitly) shown to be testable in [3], and $\mathcal{P}^*_H$ was shown to be testable in [4].
- ***k*-*colorability.*** The k-colorability property was (implicitly) shown to be testable already in [33]. In [22], a simplified explicit tester was studied with a significantly better query complexity. This result was further improved by [6].
- *Induced vertex colorability.* The main technical step in the proof of the main result of [4] was in showing that for every *finite* set of k-colored graphs $\mathcal{K}$, one can test the property of a graph being vertex k-colorable with no induced colored graph from the set $\mathcal{K}$. Note that any such property is hereditary.
- *Induced edge colorability.* Following [4], further induced edge-colorability properties were studied in [16]. In this case we have a *finite* set of *k-edge*-colored graphs $\mathcal{K}$, and the property defined by $\mathcal{K}$ is that of having a k-edge-coloring with no induced colored graph from the set $\mathcal{K}$. Note that any such property is hereditary and that by Theorem 1 we can even take $\mathcal{K}$ to be an infinite family of edge-colored graphs.
- *Graph partition problems.* One of the main results of [22] is that any graph-partition problem is testable with *two-sided* error. A characterization of the graph-partition properties that are testable with one-sided error was obtained in [24]. This characterization (essentially) follows as a special case of Theorem 2, as what it (implicitly) states is that a partition problem is testable with one-sided error if and only if it is hereditary.
- *Monotone graph properties.* Very recently, the authors have shown in [11] that any monotone graph property is testable with one-sided error (a graph property is monotone if it is closed under removal of vertices *and* edges; therefore, any monotone property is in particular hereditary). Though this family of graph properties is very general and contains many interesting graph properties such as k-colorability, being H-free and certain Ramsey properties, it fails to include many interesting hereditary nonmonotone properties such as those that were discussed in subsection 2.1.
- *One-sided vs. two-sided testers.* The first author has shown [24, Appendix D], that if a hereditary graph property is testable with two-sided error, then it is also testable with one-sided error (but not necessarily with the same query complexity). By Theorem 1, this transformation becomes obsolete, as Theorem 1 directly asserts that any hereditary graph property is testable with one-sided error.

- *Bounded first-order graph properties.* Theorem 4 extends the main result of [4], where the first-order graph property can contain only a single predicate A_i.

It is important to note that Theorems 1 and 2 do not assert the existence of one-sided error testers, which are as efficient as the ad hoc testers that were designed for every specific property in the above-mentioned papers. For example, the query complexity of the tester for k-colorability that follows as a special case of Theorem 1 is significantly larger than the query complexity which is guaranteed by the main result of [22] and [6]. These large bounds are obviously a consequence of the generality of Theorems 1 and 2. Furthermore, by Theorem 4 of [11], the upper bounds of Theorems 1 and 2 cannot be generally improved even for monotone graph properties. See the precise statement in [11].

2.5. Recent results. Recently, Lovász and Szegedy [27] found an alternative proof of Theorem 1 using the method of convergent graph sequences. Their result is slightly weaker than ours, as it does not give any explicit upper bound on the query complexity of testing even simple hereditary properties such as triangle-freeness. Rödl and Schacht [34] have generalized Theorem 1 and showed that any hereditary property of k-uniform hypergraphs is testable. This proof applies variants of the recently proved hypergraph regularity lemmas. In a joint work with Fischer and Newman [2], we have managed to give a combinatorial characterization of the testable graph properties (recall that in the present paper we mainly deal with one-sided error testers). This characterization implies that the regularity lemma is in some sense essential to graph property testing. Finally, the main techniques developed in this paper have been applied in another recent study [13] of the family of hereditary graph properties.

2.6. Organization. Our main tool in the proof of Theorem 1 is a novel application of a powerful variant of Szemerédi's regularity lemma proved in [4]. In section 3 we introduce the basic notions of regularity and state the regularity lemmas that we use and some of their standard consequences. The proof of Theorem 1 is quite involved technically, and thus we give in section 4 an overview of it. In this section we also prove Theorem 6. The ideas of this proof, especially the usage of the notion of colored-homomorphism, may be useful for handling other problems involving induced subgraphs.[2] In section 5 we give the full proof of Theorem 1 as well as the proof of Theorem 2. The proofs of Theorems 3, 4, and 5 appear in section 6. In section 7, we describe several possible extensions and open problems that this paper suggests. Throughout the paper, whenever we relate, for example, to a function $f_{3.1}$, we mean the function f defined in Lemma/Claim/Theorem 3.1. We would like to mention that readers that are not familiar with applications of the regularity lemma may find it useful to refer to [11] prior to reading this paper.

3. Regularity lemma background.

In this section we discuss the basic notions of regularity and some of the basic applications of regular partitions, and we state the regularity lemmas that we use in the proof of Theorem 1. See [26] for a comprehensive survey on the regularity lemma. We start with some basic definitions. For every two nonempty disjoint vertex sets A and B of a graph G, we define $e(A, B)$ to be the number of edges of G between A and B. The *edge density* of the pair is defined as $d(A, B) = e(A, B)/|A||B|$.

[2] As pointed to us by one of the referees, similar homomorphisms were also used in [31].

Definition 3.1 (γ-regular pair). *A pair (A, B) is* γ-regular *if for any two subsets $A' \subseteq A$ and $B' \subseteq B$, satisfying $|A'| \geq \gamma|A|$ and $|B'| \geq \gamma|B|$, the inequality $|d(A', B') - d(A, B)| \leq \gamma$ holds.*

A very useful lemma that we use in this paper is Lemma 3.2, which helps us find many induced copies of some fixed graph F, whenever a family of vertex sets are pairwise regular "enough" and their densities correspond to the edge set of F. Several versions of this lemma were previously proved in papers using the regularity lemma. See [4] for one such proof.

Lemma 3.2. *For every real $0 < \eta < 1$ and integer $f \geq 1$, there exist $\gamma = \gamma_{3.2}(\eta, f)$ and $\delta = \delta_{3.2}(\eta, f)$ with the following property. Suppose $U_1, \ldots, U_f$ is an f-tuple of disjoint vertex sets of a graph G such that for every $1 \leq i < j \leq f$ the pair (U_i, U_j) is γ-regular. Let F be a graph on f vertices $v_1, \ldots, v_f$ such that whenever $(v_i, v_j) \in E(F)$ we have $d(U_i, U_j) \geq \eta$, and whenever $(v_i, v_j) \notin E(F)$ we have $d(U_i, U_j) \leq 1 - \eta$. Then at least $\delta \prod_{i=1}^{f} |U_i|$ of the f-tuples $u_1 \in U_1, \ldots, u_f \in U_f$ span an induced copy of F, where each u_i plays the role of v_i.*

Comment 3.3. *Observe that the functions $\gamma_{3.2}(\eta, f)$ and $\delta_{3.2}(\eta, f)$ may and will be assumed to be monotone nonincreasing in f. Also, for ease of future definitions (in particular the one given in* (4)*) we set $\gamma_{3.2}(\eta, 0) = \delta_{3.2}(\eta, 0) = 1$ for any $0 < \eta < 1$.*

Note that in terms of regularity, Lemma 3.2 requires all the pairs (U_i, U_j) to be γ-regular. However, and this will be very important later in the paper, the requirements in terms of density are not very restrictive. In particular, if $\eta \leq d(U_i, U_j) \leq 1 - \eta$, then we do not care if (i, j) is an edge of F.

A partition $\mathcal{A} = \{V_i \mid 1 \leq i \leq k\}$ of the vertex set of a graph is called an *equipartition* if $|V_i|$ and $|V_j|$ differ by no more than 1 for all $1 \leq i < j \leq k$ (so in particular each V_i has one of two possible sizes). The regularity lemma of Szemerédi can be formulated as follows.

Lemma 3.4 (see [37]). *For every m and $\epsilon > 0$, there exists a number $T = T_{3.4}(m, \epsilon)$ with the following property: Any graph G on $n \geq T$ vertices has an equipartition $\mathcal{A} = \{V_i \mid 1 \leq i \leq k\}$ of $V(G)$ with $m \leq k \leq T$ for which all pairs (V_i, V_j), but at most $\epsilon\binom{k}{2}$ of them, are ϵ-regular.*

The function $T_{3.4}(m, \epsilon)$ may and is assumed to be monotone nondecreasing in m and monotone nonincreasing in ϵ. Another lemma which will be very useful in this paper is Lemma 3.5. Some versions of this lemma appear in various papers applying the regularity lemma. See [4] for one such proof.

Lemma 3.5. *For every l and γ, there exists $\delta = \delta_{3.5}(l, \gamma)$, such that for every graph G with $n \geq \delta^{-1}$ vertices there exist disjoint vertex sets $W_1, \ldots, W_l$ satisfying the following:*

1. *$|W_i| \geq \delta n$.*
2. *All $\binom{l}{2}$ pairs are γ-regular.*
3. *Either all pairs have densities at least $\frac{1}{2}$, or all pairs have densities less than $\frac{1}{2}$.*

Comment 3.6. *Observe that the function $\delta_{3.5}(l, \gamma)$ may and will be assumed to be monotone nonincreasing in l and monotone nondecreasing in γ. Therefore, for ease of future applications we will assume that for all l and γ we have $\delta_{3.5}(l, \gamma) \leq 1/2$.*

Our main tool in the proof of Theorem 1 in addition to Lemmas 3.2 and 3.5 is Lemma 3.8, proved in [4]. This lemma can be considered a variant of the standard regularity lemma, where one can use a function that defines ϵ as a function of the size of the partition, rather than having to use a fixed ϵ as in Lemma 3.4. We denote such functions by $\mathcal{E}$ throughout the paper. To state the lemma, we need the following definition.

Definition 3.7 (the function $W_{\mathcal{E},m}$). *Let $\mathcal{E}(r) : \mathbb{N} \mapsto (0,1)$ be an arbitrary monotone nonincreasing function and m be an arbitrary positive integer. We define the function $W_{\mathcal{E},m} : \mathbb{N} \mapsto (0,1)$ inductively as follows: $W_{\mathcal{E},m}(1) = T_{3.4}(m, \mathcal{E}(0))$. For any integer $i > 1$, put $R = W_{\mathcal{E},m}(i-1)$ and define*

$$W_{\mathcal{E},m}(i) = T_{3.4}(R, \mathcal{E}(R)/R^2) \ . \tag{2}$$

Lemma 3.8 (see [4]). *For every integer m and monotone nonincreasing function $\mathcal{E} : \mathbb{N} \mapsto (0,1)$, define*

$$S = S_{3.8}(m, \mathcal{E}) = W_{\mathcal{E},m}(100/\mathcal{E}(0)^4) \ .$$

For any graph $G = (V,E)$ on $n \geq S$ vertices, there exists an equipartition $\mathcal{A} = \{V_i \mid 1 \leq i \leq k\}$ of $V(G)$ as well as an equipartition $\mathcal{B} = \{U_i \mid 1 \leq i \leq k\}$ of a subset of vertices $U \subseteq V(G)$, which satisfy the following:

1. *$m \leq k \leq S$.*
2. *$U_i \subseteq V_i$ for all $i \geq 1$, and $|U_i| \geq n/S$.*
3. *In the equipartition $\mathcal{B}$, all pairs are $\mathcal{E}(k)$-regular.*
4. *All but at most $\mathcal{E}(0)\binom{k}{2}$ of the pairs $1 \leq i < j \leq k$ are such that $|d(V_i, V_j) - d(U_i, U_j)| < \mathcal{E}(0)$.*

Comment 3.9. *For technical reasons (see the proof in* [4]*), Lemma* 3.8 *requires that for any $r > 0$ the function $\mathcal{E}(r)$ will satisfy $\mathcal{E}(r) \leq \min\{\mathcal{E}(0)/4, 1/4r^2\}$. However, we can always assume without loss of generality that $\mathcal{E}$ satisfies this condition because if it does not, then we can apply Lemma 3.8 with $\mathcal{E}'$, which is defined as $\mathcal{E}'(r) = \min\{\mathcal{E}(r), \mathcal{E}(0)/4, 1/4r^2\}$. We will thus disregard this technicality.*

The main power of Lemma 3.8 is that for *any* function $\mathcal{E}$ it allows us to find k sets of vertices $U_1, \ldots, U_k$ of size $\Omega(n)$ such that all pairs (U_i, U_j) are $\mathcal{E}(k)$-regular. Note that in Lemma 3.4 we have no "control" of the relation between the number of sets and the regularity measure between them, as we first fix the regularity measure ϵ, and then get via the lemma k sets of vertices, where k can be very large in terms of ϵ. In Lemma 3.8, k can also be very large, but the lemma guarantees that so will be the regularity measure between the sets, via the function $\mathcal{E}$.

One of the difficulties in the proof of Theorem 2 is in showing that all the constants that are used in the course of the proof can be upper bounded by functions depending only on ϵ. The following observation will thus be useful.

Proposition 3.10. *For any fixed $\mathcal{E} : \mathbb{N} \mapsto (0,1)$ and integer m that is bounded by a function of ϵ only, the integer $S = S_{3.8}(m, \mathcal{E})$ can be upper bounded by a function of ϵ only.*[3]

It should be noted that the dependency of the function $T_{3.4}(m, \epsilon)$ on ϵ is a tower of exponents of height polynomial in $1/\epsilon$ (see the proof in [26]). Thus, even for moderate functions $\mathcal{E}$ the integer S has a huge dependency on ϵ, which is a tower of towers of exponents of height polynomial in $1/\epsilon$.

One of the main results of [4] is that for every *finite* set of graphs $\mathcal{F}$ the property of not containing any member of $\mathcal{F}$ as an induced subgraph can be tested with one-sided error and with query complexity depending only on ϵ. The proof technique in [4],

[3] In our application of Lemma 3.8, the function $\mathcal{E}(r)$ will (implicitly) depend on the error parameter ϵ and on the fixed property $\mathcal{P}$ being tested. For example, we will set $\mathcal{E}(r) = f(r, \mathcal{P}, \epsilon)$ for some function f. However, that will not change the fact that for a fixed property $\mathcal{P}$ the integer $S_{3.8}(m, \mathcal{E})$ can be bounded from above by a function only of ϵ.

which applies Lemmas 3.2, 3.5, and 3.8 critically, relies on the fact that the family of graphs is finite. The main step in the proof of Theorem 1 is in extending the above to *infinite* families of graphs. To this end, we use the main idea of [11], as well as a new type of homomorphism, in order to prove this stronger result. As in [11], the main idea of the proof is to apply Lemma 3.8 with a suitable function $\mathcal{E}(r)$. However, as it turns out, dealing with hereditary properties, which are not necessarily monotone, is considerably more involved. The techniques we apply in the next section, in particular the notion of colored-homomorphism, may be useful in dealing with other problems involving induced subgraphs.

4. Overview of the proof of Theorem 1.

The proof of Theorem 1 is very technical and rather long and appears in its entirety in section 5. In this section we try to give an overview of the proof, while keeping out most of the (unnecessary) technical details. We start with an equivalent formulation of Theorem 1. To this end, we introduce a convenient way of handling hereditary graph properties.

DEFINITION 4.1 (forbidden induced subgraphs). *For a hereditary graph property $\mathcal{P}$, define $\mathcal{F} = \mathcal{F}_{\mathcal{P}}$ to be the set of graphs which are minimal with respect to not satisfying property $\mathcal{P}$. In other words, a graph F belongs to $\mathcal{F}$ if it does not satisfy $\mathcal{P}$, but any graph obtained from F by removing a vertex satisfies $\mathcal{P}$.*

For a (possibly infinite) family of graph $\mathcal{F}$, a graph G is said to be *induced $\mathcal{F}$-free* if it contains no induced copy of any graph $F \in \mathcal{F}$. Note that for any hereditary graph property $\mathcal{P}$ there is a family of graphs $\mathcal{F} = \mathcal{F}_{\mathcal{P}}$ such that a graph satisfies $\mathcal{P}$ if and only if it is induced $\mathcal{F}$-free. For $\mathcal{F}$, one can simply take the family of forbidden induced subgraphs as in Definition 4.1. For example, when $\mathcal{P}$ is the property of being chordal (see subsection 2.1), then $\mathcal{F}_{\mathcal{P}}$ is the set of cycles of length at least 4. As another example, note that if $\mathcal{P}$ is the property of being bipartite, then $\mathcal{F}_{\mathcal{P}}$ is the family of odd cycles. Observe that $\mathcal{F}$ may contain *infinitely* many graphs. Clearly, for any family $\mathcal{F}$ the property of being induced $\mathcal{F}$-free is hereditary; thus, the hereditary graph properties are precisely the graph properties, which are equivalent to being induced $\mathcal{F}$-free for some family $\mathcal{F}$. For ease of presentation, it will be more convenient to derive Theorem 1 from the following (essentially equivalent[4]) lemma, whose proof is the main technical step in this paper.

LEMMA 4.2 (main technical result). *For every (possibly infinite) family of graphs $\mathcal{F}$, there are functions $N_{\mathcal{F}}(\epsilon)$, $f_{\mathcal{F}}(\epsilon)$, and $\delta_{\mathcal{F}}(\epsilon)$ such that the following holds for any $\epsilon > 0$: If a graph G on $n \geq N_{\mathcal{F}}(\epsilon)$ vertices is ϵ-far from being induced $\mathcal{F}$-free, then G contains δn^f induced copies of a graph $F \in \mathcal{F}$ of size f, where $f \leq f_{\mathcal{F}}(\epsilon)$ and $\delta \geq \delta_{\mathcal{F}}(\epsilon)$.*

Before continuing with the overview of the proof of Theorem 1, we note that the above lemma immediately implies Theorem 6. Indeed, given any graph property $\mathcal{P}$, let $\mathcal{F}$ be the family of graphs not satisfying $\mathcal{P}$. Observe that if a graph is ϵ-far from satisfying $\mathcal{P}$, then it is also ϵ-far from being induced $\mathcal{F}$-free, and thus by Lemma 4.2 it contains an induced subgraph $F \in \mathcal{F}$ of size at most $f_{\mathcal{F}}(\epsilon)$, and by our choice of $\mathcal{F}$ the graph F does not satisfy $\mathcal{P}$. Therefore, as the function $W_{\mathcal{P}}(\epsilon)$ in the statement of Theorem 6 we can take the function $f_{\mathcal{F}}(\epsilon)$.

4.1. Homomorphism and colored-homomorphism.

For the proof of Lemma 4.2, we will need a new type of homomorphism which is suitable for handling induced subgraph.

[4] See section 5 for a discussion about the subtle difference due to the possible necessity of testing some properties nonuniformly.

Definition 4.3 (colored-homomorphism). *Let K be a complete graph whose vertices are colored black or white and whose edges are colored black, white, or grey (neither the vertex coloring nor the edge coloring is assumed to be proper in the standard sense). A* colored-homomorphism *from a graph F to a graph K is a mapping $\varphi : V(F) \mapsto V(K)$, which satisfies the following:*

1. *If $(u, v) \in E(F)$, then either $\varphi(u) = \varphi(v) = t$ and t is colored black, or $\varphi(u) \neq \varphi(v)$ and $(\varphi(u), \varphi(v))$ is colored black or grey.*
2. *If $(u, v) \notin E(F)$, then either $\varphi(u) = \varphi(v) = t$ and t is colored white, or $\varphi(u) \neq \varphi(v)$ and $(\varphi(u), \varphi(v))$ is colored white or grey.*

If there is a colored-homomorphism from a graph F to a colored complete graph K, we write for brevity $F \mapsto_c K$. Some explanation is in place as to the meaning of the colors in the above definition. To this end, it is instructive to compare the definition of a colored-homomorphism to the standard notion of homomorphism.

Definition 4.4 (homomorphism). *A homomorphism from a graph F to a graph K is a mapping $\varphi : V(F) \mapsto V(K)$, which maps edges to edges; namely, $(v, u) \in E(F)$ implies $(\varphi(v), \varphi(u)) \in E(K)$.*

For brevity, we denote by $F \mapsto K$ the fact that there is a homomorphism from F to K. The fact that $F \mapsto K$ simply means that we can partition the vertex set of F into $k = |V(K)|$ subsets $V_1, \dots, V_k$, such that each V_i is edgeless and if $(i, j) \notin E(K)$, then none of the vertices of F that belong to V_i is connected to any of the vertices of F that belong to V_j. In particular, note that $F \mapsto K_k$ if and only if F is k-colorable (where K_k is a clique of size k). The standard notion of homomorphism is sufficient for dealing with not necessarily induced subgraphs, as was carried out in [11]. The reason is that having a homomorphism to a graph K is "closed under removal of vertices and edges" in the sense that if $F \mapsto K$ and F' is a subgraph of F, then $F' \mapsto K$. When one wants to handle *induced* subgraphs, it soon turns out that a standard homomorphism is not sufficient, as it does not supply enough information about F. The reason for that is that a standard homomorphism has no requirement about the nonedges of the graph. Returning to the colored-homomorphism from Definition 4.3, suppose we interpret the colors of K as follows: A white edge of K represents a nonedge, a black edge of K represents an existing edge, and a grey edge represents a "do not care." As for the vertex colors, we think of a black vertex as a complete graph and a white vertex as an edgeless graph. Thus, the fact that $F \mapsto_c K$, where K is a colored complete graph of size k, is equivalent to the following: There is a partition of $V(F)$ into k subsets $V_1, \dots, V_k$ such that each V_i is either complete or edgeless, where V_i is complete if $i \in V(K)$ is black and edgeless if $i \in V(K)$ is white. Also, if (i, j) is colored white, then none of the vertices of F that belong to V_i is connected to any of the vertices of F that belong to V_j. Similarly, if (i, j) is colored black, then all the vertices of F that belong to V_i are connected to all the vertices of F that belong to V_j. Finally, if (i, j) is colored grey, then there is no restriction on pairs ($v \in V_i$, $u \in V_j$) (or in our "formal" notation, we "do not care" if ($v \in V_i$, $u \in V_j$) is an edge of F). It is clear that a colored-homomorphism carries a lot more information about the structure of F than a standard homomorphism.

Our definition of colored-homomorphism should also be thought of with Lemma 3.2 in mind. Note that in this lemma we require only $d(U_i, U_j) \geq \eta$ when $(i, j) \in E(F)$ and $d(U_i, U_j) \leq 1 - \eta$ when $(i, j) \notin E(F)$. In particular, if $\eta \leq d(U_i, U_j) \leq 1 - \eta$, then we "do not care" whether $(i, j) \in E(F)$. In fact, as the details of the proof of Lemma 4.2 reveal, the possibility of having grey edges in the coloring of K in the definition of the colored-homomorphism is unavoidable (at least in our proof). Note that as far as Lemma 3.2 is concerned, we need only the edge coloring in the colored-homomorphism. The details below supply some explanation for the need of the vertex coloring.

We now turn to discuss the relation between the standard regularity lemma (Lemma 3.4), the stronger regularity lemma (Lemma 3.8), and colored-homomorphism. We stress that some of the explanations we give below are not completely accurate and are given in order to explain the main ideas of the proof. The formal proof appears in section 5. Given $\epsilon > 0$ and a graph G, Lemma 3.4 returns an equipartition of $V(G)$ of size k. Let the *regularity graph* of G, denoted $R = R(G)$, be the following graph. R is a graph on k vertices, where vertices i and j are connected if and only if (V_i, V_j) is a dense regular pair (with the appropriate parameters). In some sense, the regularity graph is an approximation of the original graph, up to ϵn^2 modifications. This approximation was good enough when considering monotone properties in [11] (this notion of regularity graph is standard when applying Lemma 3.4), but it is not good enough when dealing with induced graphs, which is the case we consider here. The reason is that R approximates only the dense pairs of the equipartition, while it carries no restriction or information on the sparse pairs in this equipartition. This is somewhat analogous to the fact that a standard homomorphism is not good enough for dealing with induced subgraphs. Just like we defined colored-homomorphism, we introduce *colored regularity graphs* as follows: Let R be a complete graph on k vertices. Color (i, j) black if (V_i, V_j) is a very dense pair, white if (V_i, V_j) is a very sparse pair, and grey if (V_i, V_j) is neither very dense nor very sparse (we omit the precise definition of "very"). Note that a colored regularity graph carries a lot more information about G. Note also how this definition relates to a colored-homomorphism.

4.2. Proof strategy. Our overall strategy for the proof of Lemma 4.2 can be described as follows: Given a graph G, which is ϵ-far from being induced $\mathcal{F}$-free, we will construct another "well-structured" graph $\tilde{G}$ by making fewer than ϵn^2 edge modifications. This will guarantee that $\tilde{G}$ spans an induced copy of some $F \in \mathcal{F}$. One of the key ingredients of the proof will be that as G is close to $\tilde{G}$ and $\tilde{G}$ is "well structured" we will be able to infer that G also contains a copy of some graph $F' \in \mathcal{F}$ in a way that will allow us to infer that G actually contains *many* induced copies of some (possibly other) $F'' \in \mathcal{F}$. For simplicity, in the following overview, we will briefly argue how to find many induced copies of some graph $F \in \mathcal{F}$ in the graph $\tilde{G}$ rather than G. The way to argue why G must also contain many copies of some $F \in \mathcal{F}$ is that the densities of the partition of G and $\tilde{G}$ are very similar. See the next section for the full details.

Suppose a graph G is ϵ-far from being induced $\mathcal{F}$-free. We would want to apply Lemma 3.4 on G, then construct the colored regularity graph, and then argue that if we make few (fewer than ϵn^2) modifications in G, then the new graph $\tilde{G}$ contains an induced copy of a graph $F \in \mathcal{F}$. Furthermore, as we make very few changes, the colored regularity graph is also a "good" approximation of $\tilde{G}$. We would thus want to use Lemma 3.2, where for the sets $U_1, \dots, U_f$ we take the clusters $V_1, \dots, V_k$ of the equipartition in order to get that there are many induced copies of F in $\tilde{G}$. However, we are faced with the following two problems: (i) As $\mathcal{F}$ may be infinite, we do not know the size of the graph $F \in \mathcal{F}$ that we may expect to find in $\tilde{G}$. As Lemma 3.2 needs to know the size of F in advance, we do not know how small a γ we should choose in order to apply Lemma 3.4. (ii) Note that Lemma 3.2 allows the copies of F to have only one vertex in each of the sets U_i. However, the induced copy of the graph $F \in \mathcal{F}$ that we may find in $\tilde{G}$ may have many vertices in each cluster V_i. Note further that Lemma 3.4 does not guarantee anything about the graphs induced by each V_i.

The main idea of the proof is to overcome the first problem by applying Lemma 3.8 with a suitable function $\mathcal{E}$ that will guarantee that the partition is regular enough even for the largest graph $F \in \mathcal{F}$ we may expect to find in $\tilde{G}$. For the second problem, we apply Lemma 3.5 on each of the clusters V_i in order to find subsets $W_{i,1}, \ldots, W_{i,f} \subset V_i$. Note that by Lemma 3.2, if for all j', j'' we have $d(W_{i,j'}, W_{i,j''}) \geq 1/2$, then $W_{i,1}, \ldots, W_{i,f}$ span many cliques of size f, while if for all j', j'' we have $d(W_{i,j'}, W_{i,j''}) \leq 1/2$, then they span many independent sets of size f. This is the main reason for the vertex coloring of R; that is, we color vertex i of R black if the sets returned by Lemma 3.5 are very dense and white if they are sparse. We note that overcoming both problems mentioned above *simultaneously* adds another level of complication.

An important ingredient in the proof of Lemma 4.2 will be the following function. The reader should think of the graphs R considered below as the set of colored regularity graphs discussed above and the parameter r as representing the size of R.

Definition 4.5 (the family $\mathcal{F}_r$). *For any (possibly infinite) family of graphs $\mathcal{F}$ and any integer r, let $\mathcal{F}_r$ be the following set of colored complete graphs: A colored complete graph R belongs to $\mathcal{F}_r$ if and only if it has at most r vertices and there is at least one $F \in \mathcal{F}$ such that $F \mapsto_c R$.*

In the proof of Lemma 4.2, the set $\mathcal{F}_r$, defined above, will represent a subset of the colored regularity graphs of size at most r, namely, those R for which there is at least one $F \in \mathcal{F}$ such that $F \mapsto_c R$. We now arrive at the key definition used in the proof of Lemma 4.2.

Definition 4.6 (the function $\Psi_{\mathcal{F}}$). *For any family of graphs $\mathcal{F}$ and integer r for which $\mathcal{F}_r \neq \emptyset$, let*

$$\Psi_{\mathcal{F}}(r) = \max_{R \in \mathcal{F}_r} \min_{\{F \in \mathcal{F} : F \mapsto_c R\}} |V(F)|. \tag{3}$$

Define $\Psi_{\mathcal{F}}(r) = 0$ if $\mathcal{F}_r = \emptyset$. Therefore, $\Psi_{\mathcal{F}}(r)$ is monotone nondecreasing in r.

One of the key definitions in [11] is a function analogous to $\Psi_{\mathcal{F}}$ but with respect to standard homomorphism. In our case as well, $\Psi_{\mathcal{F}}$ is one of the main tools with which we apply Lemma 3.8. As by Lemma 3.4 we can upper bound the size of the regularity graph R, we can also upper bound the size of the smallest graph $F \in \mathcal{F}$ for which $F \mapsto_c R$.

As we have mentioned in the previous section, the main difficulty that prevents one from proving Theorem 1 using Lemma 3.2 is that one does not know a priori the size of the graph that one may expect to find in the equipartition. This leads us to define the following function:

$$\mathcal{E}(r) = \gamma_{3.2}\left(\frac{\epsilon}{6},\ \Psi_{\mathcal{F}}(r)\right) \cdot \delta_{3.5}\left(\Psi_{\mathcal{F}}(r),\ \gamma_{3.2}\left(\frac{\epsilon}{6},\ \Psi_{\mathcal{F}}(r)\right)\right). \tag{4}$$

We next try to explain why the above defined $\mathcal{E}(r)$ when applied with Lemma 3.8 is useful in resolving the two difficulties mentioned above. Recall that r stands for the size of the regularity graph returned by Lemma 3.8. If we apply Lemma 3.8 with the above $\mathcal{E}$, then by the first term in the definition of $\mathcal{E}$ we know that the sets U_i (recall the statement of Lemma 3.4) are regular enough to allow one to apply Lemma 3.2 with the largest member of $\mathcal{F}$, which we may need to work with. This is due to invoking $\Psi_{\mathcal{F}}(r)$. The reason we need the second term in the definition of $\mathcal{E}$ is that we intend to apply Lemma 3.5 on each of the sets U_i in order to obtain certain subsets of U_i. This term guarantees that even the subsets of U_i will be "regular enough" for our purposes (via Claim 5.1).

5. Proofs of main results.

We start with the proof of Lemma 4.2, which is the main technical step in the proof of Theorem 1. We then use Theorem 1 in order to prove Theorem 2. We assume the reader is familiar with the overview of the proof of Lemma 4.2 given in section 4. For the proof, we need the following simple fact, which states that large enough subsets of a regular pair are themselves somewhat regular.

Claim 5.1. *If (A,B) is a γ-regular pair with density η, and $A' \subseteq A$ and $B' \subseteq B$ satisfy $|A'| \geq \xi|A|$ and $|B'| \geq \xi|B|$ for some $\xi \geq \gamma$, then (A',B') is a* $\max\{2\gamma, \gamma/\xi\}$*-regular pair.*

Proof. As (A,B) is a γ-regular pair with density η, then by definition of a regular pair, for every pair of subsets of $A' \subseteq A$ with $|A'| \geq \xi|A| \geq \gamma|A|$ and $B' \subseteq B$ with $|B'| \geq \xi|B| \geq \gamma|B|$ we have $|d(A',B') - d(A,B)| \leq \gamma$. Note that if A' and B' are as above, then for every pair of subsets $A'' \subseteq A'$ and $B'' \subseteq B'$ satisfying $|A''| \geq \frac{\gamma}{\xi}|A'|$ and $|B''| \geq \frac{\gamma}{\xi}|B'|$ also satisfy $|A''| \geq \gamma|A|$ and $|B''| \geq \gamma|B|$. Therefore, by the γ-regularity of (A,B) we have $|d(A'',B'') - d(A,B)| \leq \gamma$. We thus conclude that $|d(A'',B'') - d(A',B')| \leq 2\gamma$. Hence, (A',B') is $\max\{2\gamma, \gamma/\xi\}$-regular. □

Proof of Lemma 4.2. Fix any family of graphs $\mathcal{F}$. Let $\Psi_{\mathcal{F}}(r)$ be the function from Definition 4.6 and define the following functions of r:

$$\alpha(r) = \alpha_{\mathcal{F}}(r) = \delta_{3.5}(\Psi_{\mathcal{F}}(r), \quad \gamma_{3.2}(\epsilon/6,\ \Psi_{\mathcal{F}}(r))), \tag{5}$$

$$\beta(r) = \beta_{\mathcal{F}}(r) = \alpha(r) \cdot \gamma_{3.2}(\epsilon/6,\ \Psi_{\mathcal{F}}(r)), \tag{6}$$

and

$$\mathcal{E}(r) = \mathcal{E}_{\mathcal{F}}(r) = \begin{cases} \epsilon/6, & r = 0, \\ \min\{\beta(r),\ \epsilon/6\}, & r \geq 1. \end{cases} \tag{7}$$

For the rest of the proof, set

$$S(\epsilon) = S_{\mathcal{F}}(\epsilon) = S_{3.8}(6/\epsilon, \mathcal{E}), \tag{8}$$

and note that as we define $S(\epsilon)$ in terms of $m = 6/\epsilon$ we get by Proposition 3.10 that $S(\epsilon)$ is indeed a function only of ϵ. We now set $N_{\mathcal{F}}(\epsilon)$ to be the following function of ϵ:

$$N = N_{\mathcal{F}}(\epsilon) = S(\epsilon) \tag{9}$$

(as we have just argued, $S(\epsilon)$ and therefore also N can be upper bounded by functions only of ϵ). We postpone the definition of $f_{\mathcal{F}}(\epsilon)$ and $\delta_{\mathcal{F}}(\epsilon)$ until the end of the proof.

In the rest of the proof, we consider any graph G on n vertices, with $n \geq N \geq S(\epsilon)$, which is ϵ-far from being induced $\mathcal{F}$-free. Given G, we can use Lemma 3.8 with $m = 6/\epsilon$ and $\mathcal{E}(r)$ as defined in (7) in order to obtain an equipartition $\mathcal{A}$ of $V(G)$ into $6/\epsilon \leq k \leq S(\epsilon)$ clusters $V_1, \ldots, V_k$ (this is possible by item 1 in Lemma 3.8). Throughout the rest of the proof, k will denote the size of the equipartition returned by Lemma 3.8. By item 2 of Lemma 3.8, for every $1 \leq i \leq k$ we have sets $U_i \subseteq V_i$ each of size at least $n/S(\epsilon)$. Also, by item 3 of Lemma 3.8, every pair of these sets is at least $\beta(k)$-regular (recall that $\mathcal{E}(k) \leq \beta(k)$). For each $1 \leq i \leq k$, apply Lemma 3.5 on the subgraph induced by G on each U_i with $\ell = \Psi_{\mathcal{F}}(k)$ and $\gamma = \gamma_{3.2}(\epsilon/6, \Psi_{\mathcal{F}}(k))$ in order to obtain the appropriate sets $W_{i,1}, \ldots, W_{i,\Psi_{\mathcal{F}}(k)} \subset U_i$, all of size at least $\alpha(k)|U_i|$

(recall the definition of $\alpha(r)$ in (5)). It is crucial to note that we apply Lemma 3.5 on each of the sets $U_1, \dots, U_k$ *after* we apply Lemma 3.8 on G, and thus we "know" the value of k. The following observation will be useful for the rest of the proof.

Claim 5.2. *All the pairs $(W_{i,i'}, W_{j,j'})$ are $\gamma_{3.2}(\epsilon/6, \Psi_{\mathcal{F}}(k))$-regular. Also, if $i \neq j$, then we also have $|d(W_{i,i'}, W_{j,j'}) - d(U_i, U_j)| \leq \epsilon/6$.*

Proof. Consider first pairs that belong to the same set U_i. In this case, the fact that any pair $(W_{i,i'}, W_{i,j'})$ is $\gamma_{3.2}(\epsilon/6, \Psi_{\mathcal{F}}(k))$-regular follows immediately from our choice of these sets, as we applied Lemma 3.5 on each set U_i with $\gamma = \gamma_{3.2}(\epsilon/6, \Psi_{\mathcal{F}}(k))$. Consider now pairs that belong to different sets U_i, U_j. As was mentioned above, any pair (U_i, U_j) is $\beta(k)$-regular. As each set $W_{i,j}$ satisfies $|W_{i,j}| \geq \alpha(k)|U_i|$, we get from Claim 5.1 and the definition of $\beta(k)$ that any pair $(W_{i,i'}, W_{j,j'})$ is at least $\max\{2\beta(k), \beta(k)/\alpha(k)\} \leq \gamma_{3.2}(\epsilon/6, \Psi_{\mathcal{F}}(k))$-regular (here we use the fact that $\alpha(k) \leq 1/2$, which is guaranteed by Comment 3.6). Finally, as each of the sets $W_{i,j}$ satisfies $|W_{i,j}| \geq \alpha(k)|U_i| \geq \beta(k)|U_i| \geq \mathcal{E}(k)|U_i|$ we get from the fact that each pair (U_i, U_j) is $\mathcal{E}(k)$-regular that $|d(W_{i,i'}, W_{j,j'}) - d(U_i, U_j)| \leq \mathcal{E}(k) \leq \epsilon/6$, thus completing the proof. □

Recall that our goal is to show that G contains many induced copies of some graph $F \in \mathcal{F}$. To this end, we would like to apply Lemma 3.2 on some appropriately chosen subset of the sets $W_{i,j}$ defined above. As by Claim 5.2 all the pairs of sets $W_{i,j}$ are regular (we will later infer that they are regular enough for our purposes), we just have to find sets whose densities will correspond to the edge set of some graph $F \in \mathcal{F}$ (recall the statement of Lemma 3.2). To this end, we define a graph $\tilde{G}$ that will help us in choosing the sets $W_{i,j}$. The graph $\tilde{G}$ is obtained from G by adding and removing the following edges in the following order:

1. For $1 \leq i < j \leq k$ such that $|d(V_i, V_j) - d(U_i, U_j)| > \epsilon/6$, for all $v \in V_i$ and $v' \in V_j$ the pair (v, v') becomes an edge if $d(U_i, U_j) \geq \frac{1}{2}$ and becomes a nonedge if $d(U_i, U_j) < \frac{1}{2}$.
2. For $1 \leq i < j \leq k$ such that $d(U_i, U_j) < \frac{1}{3}\epsilon$, all edges between V_i and V_j are removed. For all $1 \leq i < j \leq k$ such that $d(U_i, U_j) > 1 - \frac{1}{3}\epsilon$, all nonedges between V_i and V_j become edges.
3. If for a fixed i all densities of pairs from $W_{i,1}, \dots, W_{i,l}$ are less than $\frac{1}{2}$, all edges within the vertices of V_i are removed. Otherwise, all the above densities are at least $\frac{1}{2}$ (by the choice of $W_{i,1}, \dots, W_{i,l}$ through Lemma 3.5), in which case all nonedges within V_i become edges.

In what follows we denote by $d(A, B)$ and $\tilde{d}(A, B)$ the edge density of the pair (A, B) in G and $\tilde{G}$, respectively. The following claim states several relations between the densities of G and $\tilde{G}$.

Claim 5.3. *For any i and $i' < j'$, we either have $\tilde{d}(W_{i,i'}, W_{i,j'}) = 1$ and $d(W_{i,i'}, W_{i,j'}) \geq \frac{1}{2}$ or $\tilde{d}(W_{i,i'}, W_{i,j'}) = 0$ and $d(W_{i,i'}, W_{i,j'}) \leq \frac{1}{2}$. Also, for any $i < j$ and any i', j' precisely one of the following holds:*

(1) *$\tilde{d}(V_i, V_j) = 1$ and $d(W_{i,i'}, W_{j,j'}) \geq \epsilon/6$.*

(2) *$\tilde{d}(V_i, V_j) = 0$ and $d(W_{i,i'}, W_{j,j'}) \leq 1 - \epsilon/6$.*

(3) *$\epsilon/6 \leq \tilde{d}(V_i, V_j) \leq 1 - \epsilon/6$ and $\epsilon/6 \leq d(W_{i,i'}, W_{j,j'}) \leq 1 - \epsilon/6$.*

Proof. The proof follows easily from the three steps for obtaining $\tilde{G}$ from G. The first assertion of the claim (concerning the relation between $\tilde{d}(W_{i,i'}, W_{i,j'})$ and $d(W_{i,i'}, W_{i,j'})$) follows directly from the third step of obtaining $\tilde{G}$. As for the second assertion (concerning the relation between $\tilde{d}(V_i, V_j)$ and $d(W_{i,i'}, W_{j,j'})$), assume the first step was applied to a pair (V_i, V_j). In this case either $\tilde{d}(V_i, V_j) = 1$ and $d(U_i, U_j) \geq 1/2$ or $\tilde{d}(V_i, V_j) = 0$ and $d(U_i, U_j) \leq 1/2$. By Claim 5.2 we get that in the former case for any i', j' we have $d(W_{i,i'}, W_{j,j'}) \geq 1/2 - \epsilon/6 \geq \epsilon/6$, while in the latter $d(W_{i,i'}, W_{j,j'}) \leq 1/2 + \epsilon/6 \leq 1 - \epsilon/6$, as needed. Note that if the first step was applied to a pair (V_i, V_j), then the second step has no effect, and thus either item (1) or (2) will hold at the end of the process. Assume the second step was applied to a pair (V_i, V_j). In this case either $\tilde{d}(V_i, V_j) = 1$ and $d(U_i, U_j) \geq 1 - \epsilon/3$ or $\tilde{d}(V_i, V_j) = 0$ and $d(U_i, U_j) \leq \epsilon/3$. Again, by Claim 5.2, we get that in the former case $d(W_{i,i'}, W_{j,j'}) \geq 1 - \epsilon/3 - \epsilon/6 \geq \epsilon/6$, while in the latter $d(W_{i,i'}, W_{j,j'}) \leq \epsilon/3 + \epsilon/6 \leq 1 - \epsilon/6$. If neither of the two steps was applied to (V_i, V_j), then we initially had $|d(V_i, V_j) - d(U_i, U_j)| \leq \epsilon/6$ and $\epsilon/3 \leq d(U_i, U_j) \leq 1 - \epsilon/3$. Thus, item (3) holds, as in this case we have $\epsilon/6 \leq d(V_i, V_j) = \tilde{d}(V_i, V_j) \leq 1 - \epsilon/6$ and by Claim 5.2 for any i', j' we have $\epsilon/6 \leq d(W_{i,i'}, W_{j,j'}) \leq 1 - \epsilon/6$. □

CLAIM 5.4. *The graphs G and $\tilde{G}$ differ by fewer than ϵn^2 edges.*

Proof. As the number of pairs $v \in V_i$, $v' \in V_j$ is n^2/k^2, and by item 4 of Lemma 3.8 the number of pairs $1 \leq i < j \leq k$ for which $|d(V_i, V_j) - d(U_i, U_j)| > \epsilon/6 = \mathcal{E}(0)$ is at most $\mathcal{E}(0)\binom{k}{2} = \frac{1}{6}\epsilon\binom{k}{2}$, in the first step we changed fewer than $\frac{1}{6}\epsilon\binom{k}{2}\frac{n^2}{k^2} \leq \frac{1}{6}\epsilon n^2$ edges. In the second stage, if $d(U_i, U_j) < \frac{1}{3}\epsilon$, then by the modifications made in the first step, we have $d(V_i, V_j) < \frac{1}{2}\epsilon$. Similarly, if $d(U_i, U_j) > 1 - \frac{1}{3}\epsilon$, then by the modifications made in the first step, we have $d(V_i, V_j) > 1 - \frac{1}{2}\epsilon$. Thus, in this step we make at most $\binom{k}{2}\frac{1}{2}\epsilon(n^2/k^2) \leq \frac{1}{2}\epsilon n^2$ modifications. Finally, in the third step we make at most $k\binom{n/k}{2} \leq n^2/k$ modifications. As we apply Lemma 3.8 with $m = 6/\epsilon$, we have $n^2/k \leq \frac{1}{6}\epsilon n^2$. Altogether, we make fewer than ϵn^2 modifications. □

We now turn to use the notion of colored-homomorphism, which was introduced in section 4. For the rest of the proof, let R be the following colored complete graph on k vertices. We color $i \in V(R)$ white if V_i is edgeless in $\tilde{G}$. Otherwise (i.e., V_i is a complete graph in $\tilde{G}$, by the third step in obtaining $\tilde{G}$ from G) we color v_i black. If $\tilde{d}(V_i, V_j) = 0$, we color (i, j) white; if $\tilde{d}(V_i, V_j) = 1$, we color (i, j) black; otherwise (i.e., $\epsilon/6 \leq \tilde{d}(V_i, V_j) \leq 1 - \epsilon/6$, by Claim 5.3) we color (i, j) grey. Our goal in the following two claims is to identify a graph $F \in \mathcal{F}$, which we will later show to be abundant in G.

CLAIM 5.5. *$\tilde{G}$ spans an induced copy of a graph $F' \in \mathcal{F}$. Moreover, $F' \mapsto_c R$.*

Proof. As G is by assumption ϵ-far from being induced $\mathcal{F}$-free, and by Claim 5.4 $\tilde{G}$ is obtained from G by making fewer than ϵn^2 modifications (of adding and removing edges), $\tilde{G}$ spans an induced copy of a graph $F' \in \mathcal{F}$. We claim that there is a colored-homomorphism from F' to R. Indeed, consider a mapping $\varphi : V(F') \mapsto V(R)$ which maps all the vertices of F' that belong to V_i to vertex i of R. We claim that this is a colored-homomorphism from F' to R. Suppose first that (u, v) is an edge of F'. If u and v belong to the same vertex set V_i, then V_i must be complete in $\tilde{G}$. By definition of φ, they are both mapped to $i \in V(R)$ and by our coloring of R, vertex i is colored black. If $u \in V_i$ and $v \in V_j$, then it cannot be the case that $\tilde{d}(V_i, V_j) = 0$, and hence $(i, j) \in E(R)$ was not colored white. Similarly, if (u, v) is not an edge of F', then if u and v belong to the same vertex set V_i, then V_i must be edgeless. Hence, vertex i is colored white. If $u \in V_i$ and $v \in V_j$, then it cannot be the case that $d'(V_i, V_j) = 1$, and hence $(i, j) \in E(R)$ was not colored black. We thus get that φ satisfies the definition of a colored-homomorphism. □

CLAIM 5.6. *There is a graph $F \in \mathcal{F}$ of size $f \leq \Psi_{\mathcal{F}}(k)$ for which $F \mapsto_c R$.*

Proof. By Claim 5.5, there is a graph $F' \in \mathcal{F}$ for which $F' \mapsto_c R$. Therefore, R belongs to $\mathcal{F}_k$ (recall Definition 4.5 and the fact that R is of size k). It thus follows from the definition of $\Psi_{\mathcal{F}}$ (see Definition 4.6) that $\mathcal{F}$ contains a graph of size at most $\Psi_{\mathcal{F}}(k)$ such that $F \mapsto_c R$. □

The reader may want to recall at this stage that in order to apply Lemma 3.2 with respect to a graph on f vertices we need f distinct vertex sets. The following proposition will enable us to apply Lemma 3.2 on an appropriately chosen f set of vertices in order to infer that G contains many induced copies of F.

PROPOSITION 5.7. *Let F be the graph from Claim* 5.6 *and denote its vertex set by $\{1,\dots,f\}$ with $f \le \Psi_{\mathcal{F}}(k)$. Let $\varphi : V(F) \mapsto V(R)$ be the colored homomorphism from F to R, which is guaranteed to exist by Claim* 5.6*, and put $t_i = \varphi(i)$ for every $i \in V(F)$. The following hold with respect to the sets $W_{t_1,1},\dots,W_{t_f,f}$:*

- *If $(i,j) \in E(F)$, then $d(W_{t_i,i}, W_{t_j,j}) \ge \epsilon/6$.*
- *If $(i,j) \notin E(F)$, then $d(W_{t_i,i}, W_{t_j,j}) \le 1-\epsilon/6$.*

Proof. First, note that we choose the sets as $W_{t_1,1},\dots,W_{t_f,f}$ in order to make sure that we do not choose the same $W_{i,i'}$ twice, because we may need to use several sets $W_{i,j}$ from the same set U_i (in the case that $t_i = t_j$ for some $i \neq j$). Also, observe that as $f \le \Psi_{\mathcal{F}}(k)$ and we obtained through Lemma 3.5 $\ell = \Psi_{\mathcal{F}}(k)$ sets $W_{i,j}$ from each U_i, we can indeed choose the sets in the above manner, even if all the chosen sets $W_{i,j}$ belong to the same U_i.

Assume that $(i,j) \in E(F)$. As φ is a colored homomorphism from F to R, we conclude that either $\varphi(i) = \varphi(j) = t$ and $t \in V(R)$ is colored black or $\varphi(i) = t \neq t' = \varphi(j)$ and $(t,t') \in E(R)$ is colored black or grey. By the way we colored R in the paragraph preceding Claim 5.5, this means that either $\varphi(i) = \varphi(j) = t$ and V_t is a complete graph in $\tilde{G}$ or $\varphi(i) = t \neq t' = \varphi(j)$ and $\tilde{d}(V_t, V_{t'}) \ge \epsilon/6$. Finally, by Claim 5.3 this means that in both cases $d(W_{t_i,i}, W_{t_j,j}) \ge \epsilon/6$. The case of $(i,j) \notin E(F)$ is analogous. □

The proof of Lemma 4.2 now follows easily from the above proposition. Consider the sets $W_{t_1,1},\dots,W_{t_f,f}$ as in Proposition 5.7. By Claim 5.2 any pair of these sets is at least $\gamma_{3.2}(\epsilon/6, \Psi_{\mathcal{F}}(k))$-regular in G. Moreover, by Proposition 5.7, these $f \le \Psi_{\mathcal{F}}(k)$ sets satisfy in G (*not* in $\tilde{G}$) the edge requirements of Lemma 3.2, which are needed in order to infer that they span many induced copies of F (recall that F has at most $\Psi_{\mathcal{F}}(k)$ vertices). Thus, Lemma 3.2 ensures that $W_{t_1,1},\dots,W_{t_f,f}$ span in G (*not* in $\tilde{G}$) at least

$$\delta_{3.2}(\epsilon/6, \Psi_{\mathcal{F}}(k)) \cdot \prod_{i=1}^{f} |W_{t_i,i}| \tag{10}$$

induced copies of F. We next show that we can take F as the graph in the statement of Lemma 4.2. To show this, we should define only the functions $f_{\mathcal{F}}(\epsilon)$ and $\delta_{\mathcal{F}}(\epsilon)$ (the function $N_{\mathcal{F}}(\epsilon)$ is defined in (9)). As $|U_i| \ge n/S(\epsilon)$ and $|W_{t_i,i}| \ge \alpha(k)|U_i|$, we conclude from (10) that G contains at least

$$\delta_{3.2}(\epsilon/6, \Psi_{\mathcal{F}}(k)) \cdot (\alpha(k)/S(\epsilon))^f \cdot n^f \tag{11}$$

induced copies of F. Thus, as $f \le \Psi_{\mathcal{F}}(k)$, $k \le S(\epsilon)$, and by the monotonicity properties of all the functions considered in the proof, we can replace k with $S(\epsilon)$ and f with $\Psi_{\mathcal{F}}(S(\epsilon))$ and thus define

$$f_{\mathcal{F}}(\epsilon) = \Psi_{\mathcal{F}}(S(\epsilon)). \tag{12}$$

Similarly, we can replace k and f in (11) in order to define

$$\delta_{\mathcal{F}}(\epsilon) = \frac{\delta_{3.2}(\epsilon/6, \Psi_{\mathcal{F}}(S(\epsilon)))}{(S(\epsilon)/\alpha(S(\epsilon)))^{\Psi_{\mathcal{F}}(S(\epsilon))}}. \tag{13}$$

This completes the proof of Lemma 4.2. □

Before proving Theorem 1 we briefly discuss the notions of uniform and nonuniform testing, which were defined and studied in [11] and [10]. We give here only a rough overview of the result of [10]. A tester is defined in [10] as *nonuniform* if it knows ϵ in advance, and therefore it should be able to distinguish between graphs that satisfy $\mathcal{P}$ from those that are ϵ-far from satisfying it (only for that specific ϵ). A tester is *uniform* if it can accept ϵ as part of the input. The main result of [10] is that there are monotone graph properties which have nonuniform one-sided testers but cannot be tested by a uniform (one-sided or two-sided) tester. The main reason is that there are monotone and decidable properties (in fact even in *coNP*) with the property that the query complexity of testing them cannot be computed given only the value of ϵ. It thus follows that we cannot design uniform testers for all the hereditary graph properties.

Note that in (9), (12), and (13) the only function that may be noncomputable is $\Psi_{\mathcal{F}}$. Thus, whenever this function is computable so are the three functions of Lemma 4.2. As the proof of Theorem 1 suggests (see below), once these functions are computable, the tester is uniform. Finally, we note that for any reasonable graph property, and in particular those that were discussed in subsection 2.1, $\Psi_{\mathcal{F}}$ is indeed computable (not necessarily very efficiently). Thus, these properties are testable in the usual sense. We thus assume henceforth that $\mathcal{F}$ is such that the functions $N_{\mathcal{F}}(\epsilon)$, $f_{\mathcal{F}}(\epsilon)$, and $\delta_{\mathcal{F}}(\epsilon)$ are computable. Note, however, that even if they are not computable, we still get a nonuniform tester for any (decidable) hereditary graph property.

Proof of Theorem 1. We show that any hereditary property can be tested with one-sided error even by an oblivious tester. Fix any hereditary graph property $\mathcal{P}$, and let $\mathcal{F}$ be the family of forbidden induced subgraphs of $\mathcal{P}$ as in Definition 4.1. Let $N_{\mathcal{F}}(\epsilon)$, $f_{\mathcal{F}}(\epsilon)$, and $\delta_{\mathcal{F}}(\epsilon)$ be the functions of Lemma 4.2 and assume they are computable. To design our one-sided error tester for $\mathcal{P}$, we just need to note that if a graph on n vertices contains at least δn^f induced copies of a graph F on f vertices, then sampling $2/\delta$ sets of f vertices each, which is a total of $2f/\delta$, finds an induced copy of F with probability at least $2/3$.

Given a graph G, the one-sided error tester for $\mathcal{P}$ works as follows: It asks the oracle for a subgraph of G induced by a randomly chosen set of $\max\{N_{\mathcal{F}}(\epsilon), 2f_{\mathcal{F}}(\epsilon)/\delta_{\mathcal{F}}(\epsilon)\}$ vertices. It declares G to be a graph satisfying $\mathcal{P}$ if and only if the induced subgraph on S satisfies $\mathcal{P}$. Clearly, if G satisfies $\mathcal{P}$, then as $\mathcal{P}$ is hereditary the algorithm accepts G with probability 1. If G is ϵ-far from satisfying $\mathcal{P}$ and G has less that $N_{\mathcal{F}}(\epsilon)$ vertices, the algorithm answers correctly with probability 1, as in this case S spans G. If G has more than $N_{\mathcal{F}}(\epsilon)$ vertices, then by Lemma 4.2 there is a member of $\mathcal{F}$ of size $f \leq f_{\mathcal{F}}(\epsilon)$ such that G spans $\delta_{\mathcal{F}}(\epsilon)n^f$ induced copies of F. By the observation from the preceding paragraph, S spans an induced copy of F with probability at least $2/3$. As $F \in \mathcal{F}$ and $\mathcal{P}$ is hereditary, we get that with probability at least $2/3$, the graph spanned by S does not satisfy $\mathcal{P}$. Hence, the tester rejects G with probability at least $2/3$. Also, its query complexity is always a function only of ϵ. □

Proof of Theorem 2. Let $\mathcal{P}$ be a semihereditary property and let $\mathcal{H}$ be the hereditary graph property as in Definition 2.2. We next show that $\mathcal{P}$ has an oblivious one-sided error tester. As $\mathcal{H}$ is hereditary, we get from Theorem 1 and the fact that its proof actually gives an oblivious tester for $\mathcal{H}$ that there is a function $Q_{\mathcal{H}}(\epsilon)$ such that $\mathcal{H}$ can be tested by an oblivious one-sided error tester with query complexity $Q_{\mathcal{H}}(\epsilon)$. The oblivious tester T we design for testing $\mathcal{P}$ works as follows: Its query complexity is $Q(\epsilon) = \max\{M(\epsilon/2), Q_{\mathcal{H}}(\epsilon/2)\}$ (the function M is part of Definition 2.2). After getting from the oracle the randomly chosen induced subgraph, which we denote by G', the tester T proceeds as follows: If G' is of size strictly smaller than $Q(\epsilon)$, the algorithm accepts if and only if G' satisfies $\mathcal{P}$. If G' is of size at least $Q(\epsilon)$, the algorithm accepts if and only if G' satisfies $\mathcal{H}$.

We turn to show that T is indeed an oblivious one-sided error tester for $\mathcal{P}$. We first observe that T satisfies the definition of an oblivious tester. We also note that if the input graph is of size less than $Q(\epsilon)$, then we accept the input if and only if it satisfies $\mathcal{P}$ because by the definition of an oblivious tester this means that the input graph was of size less than $Q(\epsilon)$, and therefore the oracle returned the entire input graph. Let us now consider an input of size at least $Q(\epsilon)$ and recall that $Q(\epsilon) \geq M(\epsilon/2)$. If this input satisfies $\mathcal{P}$, then by the first item of Definition 2.2 it also satisfies $\mathcal{H}$, and as in this case we accept if and only if G' satisfies $\mathcal{H}$, this means that T accepts the input. Hence, T has one-sided error. Suppose now that the input is ϵ-far from satisfying $\mathcal{P}$. This means that after adding/deleting $\frac{1}{2}\epsilon n^2$ edges, the input is still $\frac{\epsilon}{2}$-far from satisfying $\mathcal{P}$. By item 2 of Definition 2.2 and as in this case the input must be of size at least $M(\epsilon/2)$, this means that after adding/deleting $\frac{1}{2}\epsilon n^2$ edges, the input still contains an induced subgraph not satisfying $\mathcal{H}$. In other words, this means that the input is at least $\frac{\epsilon}{2}$-far from satisfying $\mathcal{H}$. As $Q(\epsilon) \geq Q_{\mathcal{H}}(\epsilon/2)$, we infer that with probability at least 2/3 the graph G' spans an induced subgraph not satisfying $\mathcal{H}$, and therefore G' does not satisfy $\mathcal{H}$ (as it is hereditary). As in this case T accepts if and only if G' satisfies $\mathcal{H}$, this means that T will reject an input that is ϵ-far from satisfying $\mathcal{P}$ with probability at least 2/3.

Assume now that property $\mathcal{P}$ has a one-sided error oblivious tester T. Our goal is to show the existence of a hereditary property $\mathcal{H}$ as in Definition 2.2. Let $\mathcal{F}$ be the following family of graphs: A graph F on $|V(F)|$ vertices belongs to $\mathcal{F}$ (i) if for some $\epsilon > 0$ the query complexity of T satisfies $Q(\epsilon) = |V(F)|$ (recall that the query complexity of T is a function of ϵ only); and (ii) if for this ϵ the sample of vertices spans a graph isomorphic to F, then T rejects the input with positive probability. We claim that we can take $\mathcal{H}$ in Definition 2.2 to be the property of being induced $\mathcal{F}$-free.

To establish the first item of Definition 2.2, it is enough to show that there is no graph G satisfying $\mathcal{P}$ which spans an induced subgraph isomorphic to a graph $F \in \mathcal{F}$. Suppose such a G exists, and consider the execution of T on G with an ϵ for which $Q(\epsilon) = |V(F)|$. By definition of $\mathcal{F}$ we get that T asks for a random subgraph of G of size $|V(F)|$ and that if T gets a graph isomorphic to F, it rejects G with positive probability. As we assume that G spans an induced copy of a graph isomorphic to F, this means that T has a nonzero probability of rejecting G, contradicting our assumption that T is one-sided.

To establish the second item of Definition 2.2, we claim that we can take $M(\epsilon) = Q(\epsilon)$. Indeed, consider a graph G on at least $Q(\epsilon)$ vertices that is ϵ-far from satisfying $\mathcal{P}$. As T is a tester for $\mathcal{P}$, it should reject G with nonzero probability. By definition of an oblivious tester and as G has at least $Q(\epsilon)$ vertices, this means that G must contain an induced subgraph F, of size precisely $Q(\epsilon)$, with the property that if T gets F from the oracle, then it rejects G. By definition of $\mathcal{F}$ this means that $F \in \mathcal{F}$. Hence, we can take F itself to be the graph not satisfying $\mathcal{H}$. □

6. Proofs of additional results.

In this section we give the proofs of Theorems 3, 4, and 5. We start with the proof of Theorem 3. Note that when considering the notion of ϵ-far$_{del}$ there is no sense in considering hereditary properties which are not satisfied by some independent set, as in this case any graph with even a single independent set (say, of size 3) is arbitrarily far from satisfying the property. Moreover, finding this independent set requires $\Omega(n^2)$ queries.

Our main tool for the proof of Theorem 3 is the following result, which is a strengthened version of a result of Frankl and Füredi in [20] and can be deduced, for example, from the main result of [38].

Theorem 7 (see [20], [38]). *For any graph $F = (R, T)$ with $|T| = t > 0$ edges, there is a constant $\delta = \delta(F)$ with the following property: For any integer n, there is a graph $G_n = (V, E)$ on n vertices, which consists of $(1 - n^{-\delta})\binom{n}{2}/t$ induced copies of F, such that no two copies of F share an edge.*

Proof of Theorem 3. By the discussion above we may assume that $\mathcal{P}$ has at least one forbidden induced subgraph $F = (R, T)$ and that F is not an independent set. Put $t = |T|$ and for any n let G_n be the graph whose existence is guaranteed by Theorem 7. As all these graphs consist of $(1 - n^{-\delta})\binom{n}{2}/t > n^2/4t$ induced copies of F, where none of the copies shares an edge, these graphs are all at least $\frac{1}{4t}$-far$_{del}$ from being induced F-free. Hence, they are also at least $\frac{1}{4t}$-far$_{del}$ from satisfying $\mathcal{P}$. On the other hand, as we assume that any clique satisfies $\mathcal{P}$, and G contains $(1 - n^{-\delta})\binom{n}{2}$ edges, any randomized algorithm with query complexity much smaller than n^{δ} cannot test$_{del}$ property $\mathcal{P}$. The reason is that the algorithm has a negligible probability of distinguishing between the graphs G_n, which are $\frac{1}{4t}$-far$_{del}$ from satisfying $\mathcal{P}$, and a clique of size n, which by assumption satisfies $\mathcal{P}$. □

Suppose we define ϵ-far$_{add}$ and testable$_{add}$ but now allowing only edge additions. One can easily see that simple modifications of the proof of Theorem 3 imply that the same lower bound can be proved for testing$_{add}$ any hereditary property which is not closed under edge additions and which is satisfied by any edgeless graph.

We continue with the proof of Theorem 4. As most of the technical details are very similar to those appearing in [4], we discuss only the main idea needed to obtain the extension of the result of [4]. We start with a useful result of [4].

Definition 6.1 (indistinguishability). *Two graph properties $\mathcal{P}$ and $\mathcal{Q}$ are called* indistinguishable *if for every $\epsilon > 0$ there exists $N = N(\epsilon)$ satisfying the following; A graph on $n \geq N$ vertices satisfying one of the properties is never ϵ-far from satisfying the other.*

Lemma 6.2 (see [4]). *If $\mathcal{P}$ and $\mathcal{Q}$ are indistinguishable graph properties, then $\mathcal{P}$ is testable if and only if $\mathcal{Q}$ is testable.*

We next define an extension of the notion of colorability. A similar notion was used in [4], where $\mathcal{F}$ was restricted to be *finite*.

Definition 6.3 ($\mathcal{F}$-colorability). *Suppose we are given an integer c, and a (possibly infinite) family (with repetitions) $\mathcal{F}$ of graphs, each of which is provided with a c-coloring (i.e., a function from its vertex set to $\{1, \dots, c\}$ which is not necessarily a proper c-coloring in the usual sense). A c-coloring of a graph G is called an* $\mathcal{F}$-coloring *if no member of $\mathcal{F}$ appears as an induced subgraph of G with an identical coloring. A graph G is called* $\mathcal{F}$-colorable *if there exists an $\mathcal{F}$-coloring of it.*

Note that for any family of colored graphs $\mathcal{F}$ (finite or infinite), being $\mathcal{F}$-colorable is a hereditary graph property. We thus get the following from Theorem 1.

Lemma 6.4. *For any family of colored graphs $\mathcal{F}$, being $\mathcal{F}$-colorable is testable.*

Note that by Theorem 1, being $\mathcal{F}$-colorable is in fact testable with one-sided error, but we do not need this stronger assertion here. The following lemma shows the relevance of the notion of $\mathcal{F}$-colorability for the proof of Theorem 4.

LEMMA 6.5. *For every first order property $\mathcal{P}$ of the form*

$$\exists x_1, \ldots, x_t \bigwedge_{i=1}^{\infty} \forall y_1, \ldots, y_i \; A_i(x_1, \ldots, x_t, y_1, \ldots, y_i),$$

there exists a (possibly infinite) family $\mathcal{F}$ of $(2^{t+\binom{t}{2}} + 1)$-colored graphs such that the property $\mathcal{P}$ is indistinguishable from the property of being $\mathcal{F}$-colorable.

The proof of the above lemma uses ideas very similar to those used to prove Lemma 2.2 in [4] and is thus omitted. We briefly mention that one can use the same technique of [4] along with the fact that one is allowed to put in $\mathcal{F}$ *infinitely* many forbidden colored subgraphs. The proof of Theorem 4 is immediate from Lemmas 6.2, 6.4, and 6.5.

We conclude this section with the proof of Theorem 5.

Proof of Theorem 5. For each of the hereditary properties $\mathcal{P}_i$, let $\mathcal{F}_i$ be the family of forbidden induced subgraphs of $\mathcal{P}_i$ as in Definition 4.1, and let $\mathcal{F} = \mathcal{F}_1 \bigcup \mathcal{F}_2 \bigcup \mathcal{F}_3 \bigcup \ldots$. Clearly, a graph G satisfies all the properties of $\mathcal{P}$ if and only if it is induced $\mathcal{F}$-free. Consider a graph G which is ϵ-far from satisfying all the properties of $\mathcal{P}$. In this case G is also ϵ-far from being induced $\mathcal{F}$-free; hence, by Lemma 4.2, there is a graph $F \in \mathcal{F}$ of size $f = f_{\mathcal{F}}(\epsilon)$ such that G contains $\delta_{\mathcal{F}}(\epsilon)n^f$ induced copies of F. Note that adding or removing an edge from G destroys at most $\binom{n}{f-2} \leq n^{f-2}$ induced copies of F. Thus, one must add or delete at least $\delta_{\mathcal{F}}(\epsilon)n^2$ edges to G in order to turn it into a graph containing no induced copy of F. Let i be such that $F \in \mathcal{F}_i$. We may now infer that G is $\delta_{\mathcal{F}}(\epsilon)$-far from satisfying $\mathcal{P}_i$. Finally, note that as $\mathcal{F}$ is determined by $\mathcal{P}$, we can also say that G is $\delta_{\mathcal{P}}(\epsilon)$-far from satisfying $\mathcal{P}_i$. □

7. Concluding remarks and open problems.

- Our main result in this paper can be considered a characterization of the natural graph properties that are testable with one-sided error. Thus, a natural and interesting open problem related to this paper is to complete the characterization of the graph properties that are testable with one-sided error by arbitrary testers, and not just oblivious ones.
- Theorem 1 asserts that any hereditary property is testable with one-sided error. However, the upper bounds on the query complexity, which this theorem guarantees, are huge. Even for rather simple properties, these bounds are towers of towers of exponents of height polynomial in $1/\epsilon$. Some specific properties, such as k-colorability, have far more efficient testers, whose query complexity is polynomial in $1/\epsilon$ (see [6]). For others, like being H-free (that is, containing no copy of H as a (not necessarily induced) subgraph), it is known that whenever H is not bipartite, there is no tester (one-sided or two-sided) whose query complexity is polynomial in $1/\epsilon$ (see [1], [9]). Recall that a hereditary property $\mathcal{P}$ is equivalent to being $\mathcal{F}_{\mathcal{P}}$-free for a possibly infinite family of graphs $\mathcal{F}_{\mathcal{P}}$. The hardness of testing hereditary properties for which $\mathcal{F}_{\mathcal{P}}$ is finite is (relatively) well understood, as it follows from the main result of [7] that if $\mathcal{F}_{\mathcal{P}}$ has a graph on at least five vertices, then there is no tester (one-sided or two-sided) for $\mathcal{P}$ whose query complexity is polynomial in $1/\epsilon$. When $\mathcal{F}_{\mathcal{P}}$ is infinite, the situation is much more complicated, and there are no general results which guarantee or rule out the possibility of designing testers with query complexity polynomial in $1/\epsilon$. In particular, a natural intriguing and probably challenging problem is the following:

  Which hereditary graph properties can be tested with $\boldsymbol{poly(1/\epsilon)}$ *queries?*

As a special case of this problem, it seems interesting to study the query complexity needed to test the natural graph properties that were discussed in subsection 2.1.

- Theorem 2 gives a precise characterization of the graph properties that have oblivious one-sided testers. As we have explained in section 1, any natural property that can be tested can be tested by an oblivious tester. It may thus be simpler, but still very interesting, to resolve the following problem:
  Which graph properties have (possibly two-sided) oblivious testers?
  Note that the definition of an oblivious tester implicitly assumes that the query complexity of such a tester is a function only of ϵ.
- Fischer and Newman [18] have recently shown that every testable graph property is also estimable; namely, for any such property one can estimate how far a given graph is from satisfying the property (in this paper this quantity is denoted by ϵ) while making a constant number of queries. Combining Theorem 1 and the result of [18], we get that any hereditary property is estimable.

Acknowledgment. We would like to thank the referees for many helpful comments that improved the presentation of our results.

REFERENCES

[1] N. Alon, *Testing subgraphs in large graphs*, Random Structures Algorithms, 21 (2002), pp. 359–370.

[2] N. Alon, E. Fischer, I. Newman, and A. Shapira, *A combinatorial characterization of the testable graph properties: It's all about regularity*, in STOC'06: Proceedings of the 38th Annual ACM Symposium on Theory of Computing, 2006, pp. 251–260.

[3] N. Alon, R. A. Duke, H. Lefmann, V. Rödl, and R. Yuster, *The algorithmic aspects of the regularity lemma*, J. Algorithms, 16 (1994), pp. 80–109.

[4] N. Alon, E. Fischer, M. Krivelevich, and M. Szegedy, *Efficient testing of large graphs*, Combinatorica, 20 (2000), pp. 451–476.

[5] N. Alon, M. Krivelevich, I. Newman, and M. Szegedy, *Regular languages are testable with a constant number of queries*, SIAM J. Comput., 30 (2001), pp. 1842–1862.

[6] N. Alon and M. Krivelevich, *Testing k-colorability*, SIAM J. Discrete Math., 15 (2002), pp. 211–227.

[7] N. Alon and A. Shapira, *A characterization of easily testable induced subgraphs*, Combin., Probab. Comput., 15 (2006), pp. 791–805.

[8] N. Alon and A. Shapira, *Testing satisfiability*, J. Algorithms, 47 (2003), pp. 87–103.

[9] N. Alon and A. Shapira, *Testing subgraphs in directed graphs*, J. Comput. System Sci., 69 (2004), pp. 353–382.

[10] N. Alon and A. Shapira, *A separation theorem in property-testing*, Combinatorica, to appear.

[11] N. Alon and A. Shapira, *Every monotone graph-property is testable*, SIAM J. Comput., to appear.

[12] N. Alon and A. Shapira, *Homomorphisms in graph property testing*, in Topics in Discrete Mathematics, Springer, Berlin, 2006, pp. 281–313.

[13] N. Alon and U. Stav, *What Is the Furthest Graph from a Hereditary Property?*, manuscript, 2006.

[14] M. Blum, M. Luby, and R. Rubinfeld, *Self-testing/correcting with applications to numerical problems*, J. Comput. System Sci., 47 (1993), pp. 549–595.

[15] T. M. Chan, *Polynomial-time approximation schemes for packing and piercing fat objects*, J. Algorithms, 46 (2003), pp. 178–189.

[16] E. Fischer, *Testing graphs for colorability properties*, Random Structures Algorithms, 26 (2005), pp. 289–309.

[17] E. Fischer, *The art of uninformed decisions: A primer to property testing*, Bull. Eur. Assoc. Theor. Comput. Sci. EATCS, 75 (2001), pp. 97–126.

[18] E. Fischer and I. Newman, *Testing versus estimation of graph properties*, in STOC'05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing, 2005, pp. 138–146.

[19] E. Fischer, I. Newman, and J. Sgall, *Functions that have read-twice constant width branching programs are not necessarily testable*, Random Structures Algorithms, 24 (2004), pp. 175–193.

[20] P. Frankl and Z. Füredi, *Colored packings of sets*, in Combinatorial Design Theory, North–Holland, Amsterdam, 1987, pp. 165–177.

[21] O. Goldreich, *Combinatorial property testing (a survey)*, in Randomization Methods in Algorithm Design, P. Pardalos, S. Rajasekaran, and J. Rolim, eds., AMS, Providence, RI, 1998, pp. 45–59.

[22] O. Goldreich, S. Goldwasser, and D. Ron, *Property testing and its connection to learning and approximation*, J. ACM, 45 (1998), pp. 653–750.

[23] O. Goldreich and D. Ron, *Property testing in bounded degree graphs*, Algorithmica, 32 (2002), pp. 302–343.

[24] O. Goldreich and L. Trevisan, *Three theorems regarding testing graph properties*, Random Structures Algorithms, 23 (2003), pp. 23–57.

[25] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, London, Toronto, 1980.

[26] J. Komlós and M. Simonovits, *Szemerédi's regularity lemma and its applications in graph theory*, in Combinatorics, Paul Erdős is Eighty, Vol. II, D. Miklós, V. T. Sós, and T. Szőnyi, eds., János Bolyai Mathematical Society, Budapest, Hungary, 1996, pp. 295–352.

[27] L. Lovász and B. Szegedy, *Graph Limits and Testing Hereditary Graph Properties*, manuscript, 2006.

[28] T. A. McKee and F. R. McMorris, *Topics in Intersection Graph Theory*, SIAM, Philadelphia, 1999.

[29] I. Newman, *Testing of functions that have small width branching programs*, in Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science, 2000, pp. 251–258.

[30] M. Parnas and D. Ron, *Testing the diameter of graphs*, Random Structures Algorithms, 20 (2002), pp. 165–183.

[31] H. J. Promel and A. Steger, *Excluding induced subgraphs.* III. *A general asymptotic*, Random Structures Algorithms, 3 (1992), pp. 19–31.

[32] J. L. Ramírez-Alfonsín and B. A. Reed, eds., *Perfect Graphs*, Wiley, Chichester, 2001.

[33] V. Rödl and R. Duke, *On graphs with small subgraphs of large chromatic number*, Graphs Combin., 1 (1985), pp. 91–96.

[34] V. Rödl and M. Schacht, *Property testing in hypergraphs and the removal lemma*, in STOC'07: Proceedings of the 39th Annual ACM Symposium on Theory of Computing, to appear.

[35] D. Ron, *Property testing*, in Handbook of Randomized Computing, Vol. II, P. M. Pardalos, S. Rajasekaran, J. Reif, and J. D. P. Rolim, eds., Kluwer Academic Publishers, Dordrecht, The Netherlands, 2001, pp. 597–649.

[36] R. Rubinfeld and M. Sudan, *Robust characterizations of polynomials with applications to program testing*, SIAM J. Comput., 25 (1996), pp. 252–271.

[37] E. Szemerédi, *Regular partitions of graphs*, in Problémes combinatoires et théorie des graphes, J. C. Bermond, J. C. Fournier, M. Las Vergnas, and D. Sotteau, eds., CNRS, Paris, 1978, pp. 399–401.

[38] V. Vu, *New bounds on nearly perfect matchings in hypergraphs: Higher codegrees do help*, Random Structures Algorithms, 17 (2000), pp. 29–63.

SIAM J. Comput.
Vol. 37, No. 6, pp. 1728–1776

AN ALGORITHMIC VERSION OF THE HYPERGRAPH REGULARITY METHOD*

P. E. HAXELL[†], B. NAGLE[‡], AND V. RÖDL[§]

Abstract. Extending the Szemerédi regularity lemma for graphs, P. Frankl and V. Rödl [*Random Structures Algorithms*, 20 (2002), pp. 131–164] established a 3-graph regularity lemma triple systems $\mathcal{G}_n$ admit bounded partitions of their edge sets, most classes of which consist of regularly distributed triples. Many applications of this lemma require a companion counting lemma [B. Nagle and V. Rödl, *Random Structures Algorithms*, 23 (2003), pp. 264–332] allowing one to find and enumerate subhypergraphs of a given isomorphism type in a "dense and regular" environment created by the 3-graph regularity lemma. Combined applications of these lemmas are known as the 3-graph regularity method. In this paper, we provide an algorithmic version of the 3-graph regularity lemma which, as we show, is compatible with a counting lemma. We also discuss some applications.

Key words. algorithmic regularity lemma for hypergraphs, counting lemma for hypergraphs

AMS subject classifications. Primary, 05C65; Secondary, 05D05, 68R05

DOI. 10.1137/060652385

1. Introduction.

Szemerédi's regularity lemma [40] is one of the most important tools in combinatorics, with applications ranging across combinatorial number theory, extremal graph theory, and theoretical computer science (see [25, 26] for excellent surveys of applications). This lemma hinges on the notion of ε-regularity. A bipartite graph $H = (X \cup Y, E)$ is (d, ε)-*regular* if for every $X' \subseteq X$, $|X'| > \varepsilon|X|$, and $Y' \subseteq Y$, $|Y'| > \varepsilon|Y|$, we have $|d_H(X', Y') - d| < \varepsilon$, where $d_H(X', Y') = |H[X', Y']|/(|X'||Y'|)$ is the *density* of the bipartite graph $H[X', Y']$ induced on the sets X' and Y'. (In this paper, graphs and hypergraphs are identified with their edge sets.) We say H is ε-regular if it is (d, ε)-regular for some d. Szemerédi's regularity lemma is stated as follows.

Theorem 1.1 (Szemerédi's regularity lemma [40]). *For all $\varepsilon > 0$ and integers $t_0 \geq 1$, there exist integers $T_0 = T_0(\varepsilon, t_0)$ and $N_0 = N_0(t_0, \varepsilon)$ so that every graph G on $N > N_0$ vertices admits a partition of its vertex set $V(G) = V_1 \cup \cdots \cup V_t$, $t_0 \leq t \leq T_0$, satisfying*

1. *$V(G) = V_1 \cup \cdots \cup V_t$ is equitable: $|V_1| \leq \cdots \leq |V_t| \leq |V_1| + 1$;*
2. *$V(G) = V_1 \cup \cdots \cup V_t$ is ε-regular: all but $\varepsilon\binom{t}{2}$ pairs V_i, V_j, $1 \leq i < j \leq t$, are ε-regular.*

Much of the strength of Szemerédi's regularity lemma rests on the ability to embed fixed subgraphs within appropriate parts of an ε-regular partition, a phenomenon formally due to the following easily proved graph "counting lemma."

Fact 1.2 (counting lemma). *For all integers k and nonnegative d, there exists $\varepsilon_0 > 0$ so that for all $0 < \varepsilon < \varepsilon_0$, whenever $H = \bigcup_{1 \leq i < j \leq k} H^{ij}$ is a k-partite graph*

*Received by the editors February 17, 2006; accepted for publication (in revised form) January 31, 2007; published electronically March 26, 2008.

http://www.siam.org/journals/sicomp/37-6/65238.html

[†] Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON, Canada, N2L 3G1 (pehaxell@math.uwaterloo.ca). This author was partially supported by the NSERC.

[‡] Department of Mathematics and Statistics, University of South Florida, Tampa, FL 33620 (bnagle@math.usf.edu). This author was partially supported by NSF grant DMS 0639839.

[§] Department of Mathematics and Computer Science, Emory University, Atlanta, GA 30322 (rodl@mathcs.emory.edu). This author was partially supported by NSF grant DMS 0300529.

on $V_1 \cup \cdots \cup V_k$*,* $|V_1| = \cdots = |V_k| = n$*, where each* H^{ij}*,* $1 \le i < j \le k$*, is* (d, ε)*-regular, then the number of* k*-cliques in* H*,* $|\mathcal{K}_k^{(2)}(H)|$*, satisfies* $d^{\binom{k}{2}} n^k \left(1 - \varepsilon^{1/k}\right) \le |\mathcal{K}_k^{(2)}(H)| \le d^{\binom{k}{2}} n^k \left(1 + \varepsilon^{1/k}\right)$*.*

Combined applications of Theorem 1.1 and Fact 1.2 are known as the *graph regularity method* and will be discussed shortly.

The great importance of Szemerédi's regularity lemma has led to a search for extensions to k-uniform hypergraphs—for example, [4, 7, 12, 13, 14]. While these early generalizations did lead to some interesting applications, they did not seem to capture the full power of Szemerédi's lemma for graphs. In particular, they did not allow for the embedding of small subsystems within a regular structure. The first hypergraph regularity lemma shown to admit a corresponding counting lemma was due to Frankl and Rödl [11] for 3-uniform hypergraphs (we shall refer to 3-uniform hypergraphs as 3-*graphs* for short). In what follows, we refer to this regularity lemma for 3-uniform hypergraphs as the 3*R-lemma* for short. The 3R-lemma guarantees that any large 3-graph $\mathcal{G}$ admits a bounded partition of its triples, most classes of which are "regularly distributed." The corresponding 3-graph counting lemma was due to Frankl, Nagle, and Rödl [11, 27]. Joint applications of these lemmas are known as the 3-*graph regularity method*, which has been used in several hypergraph problems [6, 18, 20, 21, 28, 31, 32, 38] (see also [33, 34, 36]).

The original proof of Szemerédi's regularity lemma for graphs was not algorithmic. An algorithmic version of Szemerédi's lemma was later established in [1, 2] by Alon et al. (see also [10]), rendering constructive solutions to many problems where Szemerédi's lemma is applied (see [1] for applications). Correspondingly,

the object of this paper is to establish compatible algorithmic versions of the 3*R-lemma and the counting lemma.*

Results in this paper were announced in our earlier paper [8] (see the abstract of [8]), and outlined in the extended abstract [17]. We state our results precisely in section 2.

Extending the 3R-lemma, regularity lemmas and counting lemmas for k-uniform hypergraphs, also allowing the embedding of small substructures, were recently developed by Gowers [15, 16], Nagle, Rödl, and Schacht [29], and Rödl and Skokan [35]. Most likely, it would be possible to extend our current work to provide an algorithmic version of the general k-graph regularity method. It appears that our approach here has some similarities to that of Gowers [15]. In particular, we use the concept of a hypergraph having "minimally many" copies of $K_{2,2,2}^{(3)}$, where $K_{2,2,2}^{(3)}$ is the complete 3-partite 3-graph with two vertices in each class (we call this concept "(α, δ)-minimal"). In [15], Gowers uses a related concept (cf. "α-quasirandom"), although the arguments in our paper are quite different from those in [15].

To accomplish the object of this paper, one seeks to use the approach of Alon et al. [1] for the graph case. Extending their approach in this paper becomes technical for several reasons. For one, the 3R-lemma and counting lemma of [11, 27] involve more technical environments. More importantly, however, and as we discuss momentarily, the approach taken in [1, 10] is based on the implication that "C_4-minimality implies regularity," the analogy of which fails to be true in the context of the 3R-lemma (cf. (8) and our earlier work with Dementieva [8]). To handle this problem, our paper does the following:

1. formulates a variant of the regularity lemma from [11] in the upcoming Theorem 2.4;

2. modifies the approach of Alon et al. [1] to prove Theorem 2.4 constructively; and
3. formulates and proves a counting lemma in the upcoming Theorem 2.7 which is compatible with Theorem 2.4.

The work of this paper represents some departure from the original 3R-lemma of [11] and the original counting lemma from [27]. To motivate our work, section 1.1 reviews the original approach of Alon et al. for constructively proving Szemerédi's regularity lemma. Section 1.2 then emphasizes which parts from section 1.1 shall be the same in this paper and illustrates how we modify the remaining parts. In section 2, we then precisely state the main results of our paper and discuss an application. The remainder of the paper proves the main results discussed in section 2.

1.1. Constructive proof of the regularity lemma. The algorithmic version of Szemerédi's regularity lemma states that for each $\varepsilon > 0$ and t_0, for each graph G on $N > N_0(\varepsilon, t_0)$ vertices, an ε-regular t-equitable partition $V(G) = V_1 \cup \cdots \cup V_t$, $1 \leq t \leq T_0 = T_0(\varepsilon, t_0)$ (the existence of which is guaranteed by Theorem 1.1), can be constructed in time polynomial in N. In what immediately follows, we sketch the constructive proof of statement 2 of Theorem 1.1 (the essence of Theorem 1.1) where it is enough to consider the special case $t_0 = 1$.

The proof boils down to one central problem. Indeed, let $\varepsilon > 0$ be fixed and let bipartite graph H be given with bipartition $X \cup Y$ and positive density $d = |H|/(|X||Y|) > 0$. As we explain below, to prove Theorem 1.1 constructively, one is faced with the problem of

$$\text{(1)} \qquad \textit{efficiently deciding if } H \textit{ is } \varepsilon\textit{-regular},$$

a problem which, unfortunately, is co-NP-complete [1]. To handle (1), one circumvents the problem by counting the number $|\mathcal{K}_{2,2}^{(2)}(H)|$ (cf. (42)) of copies of $K_{2,2}^{(2)} = C_4$ appearing in H, an operation easily completed in time $|X|^2|Y|^2$. It is always the case that $|\mathcal{K}_{2,2}^{(2)}(H)| \geq d^4\binom{|X|}{2}\binom{|Y|}{2}(1 - o(1))$, where $o(1) \to 0$ as $\min\{|X|, |Y|\} \to \infty$. We now consider the relationship between ε-regularity and the number $|\mathcal{K}_{2,2}^{(2)}(H)|$.

On the one hand, it is not difficult to show that

$$\text{(2)} \qquad H \textit{ is } \varepsilon\textit{-regular} \quad \Longrightarrow \quad \left|\mathcal{K}_{2,2}^{(2)}(H)\right| \leq (d + f(\varepsilon))^4 \binom{|X|}{2}\binom{|Y|}{2}$$

for a fixed function $f(\varepsilon) > \varepsilon$ satisfying $f(\varepsilon) \to 0$ as $\varepsilon \to 0$. Moreover, the same proof gives

$$\text{(3)} \quad \left|\mathcal{K}_{2,2}^{(2)}(H)\right| > (d + f(\varepsilon))^4 \binom{|X|}{2}\binom{|Y|}{2} \quad \Longrightarrow \quad \exists\, O(|X||Y|)\textit{-time algorithm}$$
$$\textit{which builds } X' \subset X \textit{ and } Y' \subset Y,\ |X'| > \varepsilon|X| \textit{ and } |Y'| > \varepsilon|Y|,$$
$$\textit{such that } |d_H(X', Y') - d| > \varepsilon.$$

(We shall often refer to sets X', Y' as above as *witnesses of irregularity*.) On the other hand, convexity arguments show [1, 10]

$$\text{(4)} \qquad \left|\mathcal{K}_{2,2}^{(2)}(H)\right| \leq (d + g(\varepsilon))^4 \binom{|X|}{2}\binom{|Y|}{2} \quad \Longrightarrow \quad H \textit{ is } \varepsilon\textit{-regular}$$

for a fixed (positive) function $g(\varepsilon) < \varepsilon$. We note the following crucial remark.

Remark 1.3. For the fixed functions f and g in (2) and (4), respectively, suppose $|\mathcal{K}_{2,2}^{(2)}(H)| > (d+g(\varepsilon))^4\binom{|X|}{2}\binom{|Y|}{2}$. Then from (2), H is not $f^{-1}(g(\varepsilon))$-regular, where from $f(\varepsilon) > \varepsilon > g(\varepsilon)$ we necessarily have $f^{-1}(g(\varepsilon)) < g(\varepsilon) < \varepsilon$. As such, H may simultaneously be both ε-regular and $f^{-1}(g(\varepsilon))$-irregular.

We now use the implications in (2)–(4) to give the constructive proof of Theorem 1.1 from [1]. Indeed, with $\varepsilon > 0$ fixed and N-vertex graph G given, one iteratively builds a special sequence of partitions $\mathcal{P}_\ell :\ V(G) = V_1^{(\ell)} \cup \cdots \cup V_{t_\ell}^{(\ell)}$, $\ell \geq 1$, and defines for each $\ell \geq 1$ the corresponding *index* (cf. [39, 40])

$$\text{ind}\,\mathcal{P}_\ell = \frac{1}{N^2}\sum_{1\leq i<j\leq t_\ell} d_G^2\left(V_i^{(\ell)}, V_j^{(\ell)}\right)\left|V_i^{(\ell)}\right|\left|V_j^{(\ell)}\right|, \tag{5}$$

noting that ind $\mathcal{P}_\ell$ never exceeds one.

Suppose partitions $\mathcal{P}_1, \dots, \mathcal{P}_\ell$ have been constructed where $\mathcal{P}_\ell : V(G) = V_1^{(\ell)} \cup \cdots \cup V_{t_\ell}^{(\ell)}$. We seek to know if $\mathcal{P}_\ell$ is ε-regular (for if it is, then $\mathcal{P}_\ell$ is the partition desired). Recalling (1), however, we cannot, in general, efficiently decide the regularity status of $G[V_a^{(\ell)}, V_b^{(\ell)}]$, $1 \leq a < b \leq t_\ell$, for a given $\varepsilon > 0$. As such, we instead count $|\mathcal{K}_{2,2}^{(2)}(G[V_a^{(\ell)}, V_b^{(\ell)}])|$. With $d_{ab}^{(\ell)}$ denoting the density of $G[V_a^{(\ell)}, V_b^{(\ell)}]$ and with the functions f and g given in (2) and (4), respectively, we compute $|\mathcal{K}_{2,2}^{(2)}(G[V_a^{(\ell)}, V_b^{(\ell)}])|$ and record the following information:

1. if $|\mathcal{K}_{2,2}^{(2)}(G[V_a^{(\ell)}, V_b^{(\ell)}])| \leq (d_{ab}^{(\ell)} + g(\varepsilon))^4\binom{|V_a^{(\ell)}|}{2}\binom{|V_b^{(\ell)}|}{2}$, then $G[V_a^{(\ell)}, V_b^{(\ell)}]$ is ε-regular by (4);
2. if $|\mathcal{K}_{2,2}^{(2)}(G[V_a^{(\ell)}, V_b^{(\ell)}])| > (d_{ab}^{(\ell)} + g(\varepsilon))^4\binom{|V_a^{(\ell)}|}{2}\binom{|V_b^{(\ell)}|}{2}$, then $G[V_a^{(\ell)}, V_b^{(\ell)}]$ is not $f^{-1}(g(\varepsilon))$-regular (cf. Remark 1.3), and by (3), we may efficiently construct witnesses $\widetilde{V}_a^{(\ell)}$ and $\widetilde{V}_b^{(\ell)}$ of the $f^{-1}(g(\varepsilon))$-irregularity of $G[V_a^{(\ell)}, V_b^{(\ell)}]$.

Correspondingly, if

(1′) all but $\varepsilon\binom{t_\ell}{2}$ of the bipartite graphs $G[V_a^{(\ell)}, V_b^{(\ell)}]$, $1 \leq a < b \leq t_\ell$, have the "minimal number" of $K_{2,2}^{(2)}$'s, i.e., such that (4) is satisfied, then the partition $\mathcal{P}_\ell$ is ε-regular, as desired;

(2′) at least $f^{-1}(g(\varepsilon))\binom{t_\ell}{2} < \varepsilon\binom{t_\ell}{2}$ of the bipartite graphs $G[V_a^{(\ell)}, V_b^{(\ell)}]$, $1 \leq a < b \leq t_\ell$, do not have the minimal number of $K_{2,2}^{(2)}$'s, then by (3), we have constructed a system of witnesses $\widetilde{V}_a^{(\ell)}$ and $\widetilde{V}_b^{(\ell)}$, $\{a,b\} \in I^{(\ell)}$ for some "large" indexing set $I^{(\ell)} \subseteq \binom{[t_\ell]}{2}$, for the $f^{-1}(g(\varepsilon))$-irregularity of $G[V_a^{(\ell)}, V_b^{(\ell)}]$.

While (1′) signifies the end of the proof, (2′) puts one directly in the center of Szemerédi's proof [39, 40] of the regularity lemma.

Indeed, with the system of constructed witnesses $\widetilde{V}_a^{(\ell)}$ and $\widetilde{V}_b^{(\ell)}$, $\{a,b\} \in I^{(\ell)}$, as above, Szemerédi's original proof [39, 40] establishes the existence of a partition $\mathcal{P}_{\ell+1} : V(G) = V_1^{(\ell+1)} \cup \cdots \cup V_{t_{\ell+1}}^{(\ell+1)}$ for which

$$t_{\ell+1} \leq t_\ell 2^{t_\ell - 1} \tag{6}$$

and

$$\text{ind}\,\mathcal{P}_{\ell+1} \geq \text{ind}\,\mathcal{P}_\ell + \frac{(f^{-1}(g(\varepsilon)))^5}{2}. \tag{7}$$

Moreover, since the system of witnesses $\widetilde{V}_a^{(\ell)}$ and $\widetilde{V}_b^{(\ell)}$, $\{a,b\} \in I^{(\ell)}$, was already constructed, Szemerédi's proof [39, 40] can be easily made to construct the partition

$\mathcal{P}_{\ell+1}$ in time $O(N)$. As such, within $2/(f^{-1}(g(\varepsilon)))^5$ iterations ℓ of the process above (recall the index never exceeds one), one must have constructed an ε-regular partition $\mathcal{P}^{(\ell)}$, as desired.

1.2. Modifying the graph approach. We see from the outline of the previous section that the constructive proof of Szemerédi's regularity lemma hinges on the relationship between the ε-regularity of a bipartite graph H and the number of $K_{2,2}^{(2)}$'s of H. The implications in (2)–(4) assert that the property of ε-regularity in graph H is essentially equivalent to the parameter $|\mathcal{K}_{2,2}^{(2)}(H)|$ being asymtotically minimized over all bipartite graphs of the same density.

The concept of 3-graph regularity in [11] is so-called (δ, r)*-regularity* for 3-graphs $\mathcal{H}$. While we define this concept precisely in Definition 3.1 below, we say, for now, that it plays the role of ε-regularity in Szemerédi's regularity lemma and is, correspondingly, the central concept in the counting lemma from [11, 27]. Similarly to (1), it is a co-NP-complete problem to verify the property of (δ, r)-regularity, and thus as in (2)–(4), we seek an easily verifiable property which is essentially equivalent to (δ, r)-regularity. Following the outline of the previous section, a primary candidate would be to consider the number of $K_{2,2,2}^{(3)}$'s in a 3-graph $\mathcal{H}$, where we recall that $K_{2,2,2}^{(3)}$ is the complete 3-partite 3-graph with two vertices in each class. In Definition 2.2 below, we define the appropriate sense in which a 3-graph $\mathcal{H}$ contains "minimally many" copies of $K_{2,2,2}^{(3)}$ (viz. (α, δ)-minimality). Then, one hopes to transparently extend the graph approach above, going between (δ_1, r)-regularity and (α, δ_2)-minimality analogously to how we went between ε-regularity and "$K_{2,2}^{(2)}$-minimality" with (2)–(4).

The problem in extending the approach above is that, unlike the case of graphs where one has the equivalence established in (2)–(4),

(8) (δ_1, r)*-regularity is not equivalent to* (α, δ_2)*-minimality.*

In particular, our earlier work in [8] established that (with appropriately quantified constants)

(9) (δ_1, r)*-regularity implies* (α, δ_2)*-minimality*

but that

(10) (α, δ_2)*-minimality does not imply* (δ_1, r)*-regularity.*

(The details of (9) will be discussed in Proposition 3.2 below.) Then (10) implies that step (1′) of the graph outline above cannot be extended to the context of (δ_1, r)-regular hypergraphs. We now very roughly indicate the main idea we take in resolving this technicality. (The precise details are given in section 3.)

Unlike Frankl and Rödl's regularity lemma, we formulate our algorithmic regularity lemma in terms of (α, δ_2)-minimality, meaning that for a given 3-graph $\mathcal{H}$ on n vertices, we want to construct a partition $\mathcal{P}$, most "parts" of which have minimally many $K_{2,2,2}^{(3)}$'s. Although the concepts are (technically) different, the main idea is similar to Szemerédi's proof [39, 40]. Let $\mathcal{P}_i$ be a constructed partition for $\mathcal{H}$.

- If $\mathcal{P}_i$ already has most parts with minimally many $K_{2,2,2}^{(3)}$'s, then our algorithm terminates with $\mathcal{P}_i$, as desired.
- If $\mathcal{P}_i$ has many parts, each with too many $K_{2,2,2}^{(3)}$'s, then by (9), these parts cannot be (δ_1, r)-regular (and witnesses against the (δ_1, r)-regularity can be

found in time $O(n^5)$). Now, however, we are philosophically at the center of Frankl and Rödl's proof: *we may refine $\mathcal{P}_i$ to obtain a new partition $\mathcal{P}_{i+1}$ whose index (defined later) is significantly larger than that of $\mathcal{P}_i$.*

The rough sketch above indicates that we find a minimal partition $\mathcal{P}$ rather than a (δ_1, r)-regular one. A main result in this paper (the upcoming Theorem 2.7) asserts that

(11) *the concept of (α, δ_2)-minimality admits a corresponding counting lemma.*

As such, we do not need to find a (δ_1, r)-regular partition, as did Frankl and Rödl, in order to cooperatively apply a counting lemma.

2. Main results of paper.

In this section, we state the main results of this paper, an algorithmic 3R-lemma, Theorem 2.4, and a corresponding 3-graph counting lemma, Theorem 2.7. We conclude the section with an application of our theorems to a constructive hypergraph problem.

To state Theorems 2.4 and 2.7, we require some notation and concepts. We shall begin this section by presenting concepts needed to state Theorem 2.4.

2.1. 3R-partitions.

Given a 3-graph $\mathcal{G}$ with vertex set V, the 3R-lemma guarantees partitions of the vertices V and the pairs $\binom{V}{2}$, w.r.t. which $\mathcal{G}$ behaves "regularly." In what follows, we refer to such families of partitions as *3R-partitions.* We now describe the vertex and pair structure of 3R-partitions.

Let V be a set with $|V| = N$ and let integers ℓ and t and reals $\gamma > 0$ and $\varepsilon > 0$ be given. An *$(\ell, t, \gamma, \varepsilon)$-partition* $\mathcal{P}$ of $\binom{V}{2}$ is an (auxiliary) partition $V = V_0 \cup V_1 \cup \cdots \cup V_t$ of V, together with a system of edge-disjoint bipartite graphs $\mathcal{B} = \{P_a^{ij} \colon 1 \le i < j \le t,\ 0 \le a \le \ell_{ij} \le \ell\}$, such that

1. $|V_0| < t$ and $|V_1| = |V_2| = \cdots = |V_t| = \lfloor \frac{N}{t} \rfloor \overset{\text{def}}{=} n$,
2. $\bigcup_{a=0}^{\ell_{ij}} P_a^{ij} = K(V_i, V_j)$ is a partition of the complete bipartite graph $K(V_i, V_j)$ for all $1 \le i < j \le t$, and
3. setting $\mathcal{B}_{\text{reg}}$ to be the collection of those bipartite graphs from $\mathcal{B}$ that are ε-regular, we have

$$\sum_{P_a^{ij} \in \mathcal{B}_{\text{reg}}} \left|P_a^{ij}\right| > (1 - \gamma) \binom{t}{2} n^2.$$

An $(\ell, t, \gamma, \varepsilon)$-partition $\mathcal{P}$ is said to be *equitable* if for all but $\gamma\binom{t}{2}$ pairs $1 \le i < j \le t$, $|P_0^{ij}| \le \gamma n^2$ and P_a^{ij} has density $d_{P_a^{ij}}(V_i, V_j)$ satisfying $|d_{P_a^{ij}}(V_i, V_j) - \ell^{-1}| < \varepsilon$ for all $a = 1, \ldots, \ell_{ij}$.

To describe the "triple structure" of 3R-partitions requires still more definitions and notation. For a fixed set V, let an $(\ell, t, \gamma, \varepsilon)$-partition $\mathcal{P}$ of V be given. Any 3-partite graph $P \subseteq \mathcal{B}$ of the form

(12)
$$P = P_a^{ij} \cup P_b^{jk} \cup P_c^{ik}, \quad 1 \le i < j < k \le t, \quad 0 \le a \le \ell_{ij}, \quad 0 \le b \le \ell_{jk}, \quad 0 \le c \le \ell_{ik},$$

is called a *triad.* Denote by $\text{Triad}(\mathcal{P})$ the set of all such triads P. For $P \in \text{Triad}(\mathcal{P})$, let $\mathcal{K}_3^{(2)}(P)$ denote the system of *triangles* of P:

$$\mathcal{K}_3^{(2)}(P) = \left\{ \{x, y, z\} \in \binom{V}{3} : \{x, y, z\} \text{ induces a triangle } K_3^{(2)} \text{ in } P \right\}.$$

Now, let $\mathcal{G}$ be a 3-graph on vertex set $V = V(\mathcal{G})$, where V has $(\ell, t, \gamma, \varepsilon)$-partition $\mathcal{P}$. For $P \in \text{Triad}(\mathcal{P})$, we write $\mathcal{G}_P = \mathcal{G} \cap \mathcal{K}_3^{(2)}(P)$ and define the *density* of $\mathcal{G}_P$ w.r.t. P as $\alpha_P = d_{\mathcal{G}_P}(P) = |\mathcal{G}_P|/|\mathcal{K}_3^{(2)}(P)|$. Set

$$\mathcal{K}_{2,2,2}^{(3)}(\mathcal{G}_P) = \left\{ J \in \binom{V}{6} : \ J \text{ induces a copy of } K_{2,2,2}^{(3)} \text{ in } \mathcal{G}_P \right\},$$

where $K_{2,2,2}^{(3)}$ is the complete 3-partite 3-graph with 2 vertices in each class.

2.2. Algorithmic regularity lemma. Our regularity lemma hinges on the concept of (α, δ)-*minimality* which is defined for the following environment.

Setup 2.1. *Let $P = P^{12} \cup P^{23} \cup P^{13}$ be a 3-partite graph and $\mathcal{H}$ be a 3-partite 3-graph satisfying the following conditions:*

1. *$\mathcal{H}$ and P have common 3-partition $V = V(P) = V_1 \cup V_2 \cup V_3$, $|V_1| = |V_2| = |V_3| = n$;*
2. *P^{ij} is (ℓ^{-1}, ε)-regular for each $1 \le i < j \le 3$; and*
3. *$\mathcal{H} \subseteq \mathcal{K}_3^{(2)}(P)$ and $d_{\mathcal{H}}(P) = \alpha$.*

We note that Setup 2.1 models the environment of a "typical" triad $P_a^{ij} \cup P_b^{jk} \cup P_c^{ik}$ from an equitable $(\ell, t, \gamma, \varepsilon)$-partition $\mathcal{P}$ of $\binom{V}{2}$, where, here, $V = V(\mathcal{G})$ is the vertex set of a 3-graph $\mathcal{G}$ and $\mathcal{H}$ plays the role of $\mathcal{G} \cap \mathcal{K}_3^{(2)}(P)$.

We will confirm in section 4 that, with ε sufficiently small, Setup 2.1 ensures

$$\left| \mathcal{K}_{2,2,2}^{(3)}(\mathcal{H}) \right| \ge \frac{\alpha^8}{\ell^{12}} \binom{n}{2}^3 \left(1 - \varepsilon^{1/10}\right) \tag{13}$$

(see upcoming Proposition 4.1). The following definition is therefore motivated.

Definition 2.2 ((α, δ)-minimality). *For $\delta > 0$, we say 3-graph $\mathcal{H}$, as in Setup 2.1, is* (α, δ)-minimal w.r.t. P *if*

$$\left| \mathcal{K}_{2,2,2}^{(3)}(\mathcal{G}) \right| \le \frac{\alpha^8}{\ell^{12}} \binom{n}{2}^3 (1 + \delta).$$

If $\mathcal{H}$ is not (α, δ)-minimal w.r.t. P, then we say $\mathcal{H}$ is (α, δ)-excessive *w.r.t. P.*

We now define a notion of "minimality" for an $(\ell, t, \gamma, \varepsilon)$-partition $\mathcal{P}$ of $\mathcal{G}$. For α_0, $\delta > 0$, we first set

$$\text{Triad}_{(\alpha_0,\delta)\text{-exc}}(\mathcal{P}) = \Big\{ P \in \text{Triad}(\mathcal{P}) : \mathcal{G}_P = \mathcal{G} \cap \mathcal{K}_3^{(2)}(P) \text{ is } (\alpha_P, \delta)\text{-excessive w.r.t. } P, \text{ where } d_{\mathcal{G}_P}(P) = \alpha_P \ge \alpha_0 \Big\}. \tag{14}$$

We say partition $\mathcal{P}$ is (α_0, δ)-*minimal* w.r.t. $\mathcal{G}$ if

$$\sum_{P \in \text{Triad}_{(\alpha_0,\delta)\text{-exc}}(\mathcal{P})} \left| \mathcal{K}_3^{(2)}(P) \right| < \delta N^3 \tag{15}$$

and (α_0, δ)-*excessive* w.r.t. $\mathcal{G}$ otherwise. For future reference, we make the following remark.

Remark 2.3. As defined, every triad $P = P_a^{ij} \cup P_b^{jk} \cup P_c^{ik} \in \text{Triad}_{(\alpha_0,\delta)\text{-exc}}(\mathcal{P})$ necessarily satisfies that each of P_a^{ij}, P_b^{jk}, P_c^{ik} is $(1/\ell, \varepsilon)$-regular. In particular, the set

Triad($\mathcal{P}$) splits into three classes: those which are minimal, those which are excessive, and those which contain an irregular bipartite graph.

We arrive at our first main theorem.

THEOREM 2.4 (algorithmic regularity lemma). *For all reals $\alpha_0, \delta, \gamma > 0$, integers t_0 and ℓ_0, and functions $\boldsymbol{\varepsilon} : \mathbb{N}^+ \to (0,1)$, there exist integers T_0, L_0, and N_0 so that every 3-graph $\mathcal{G}$ on $N > N_0$ vertices admits an equitable and (α_0, δ)-minimal $(\ell, t, \gamma, \boldsymbol{\varepsilon}(\ell))$-partition $\mathcal{P}$ where $\ell_0 \leq \ell \leq L_0$ and $t_0 \leq t \leq T_0$. Moreover, there exists an algorithm which produces the partition $\mathcal{P}$ in time $O(N^6)$.*

In our proof of Theorem 2.4, the running time $O(N^6)$ will be easy to see. It is not, however, optimal. Indeed, combining ideas from [23, 24] with some from the current paper, one can prove a running time of $O(N^4)$. It seems likely that a running time of $O(N^3)$ should be possible. However, as refinements to the running time are not our focus in this paper, we do not discuss the issue here.

One can also prove the following form of Theorem 2.4 which may be slightly more convenient in some applications.

THEOREM 2.5. *For all $\alpha_0, \delta > 0$, integers t_0 and ℓ_0 and functions $\boldsymbol{\varepsilon} : \mathbb{N}^+ \to (0,1)$, there exist integers T_0, L_0, and N_0 so that for every 3-graph $\mathcal{G}$ on vertex set V, $|V| = N > N_0$, there exist integers $\ell_0 \leq \ell \leq L_0$ and $t_0 \leq t \leq T_0$ and a partition $\mathcal{P}$ of $\binom{V}{2}$ with the following properties:*

1. *$\mathcal{P}$ has auxiliary partition $V = V_1 \cup \cdots \cup V_t$ split as evenly as possible, i.e., $|V_1| \leq \cdots \leq |V_t| \leq |V_1| + 1$.*
2. *For each $1 \leq i < j \leq t$, we have $K(V_i, V_j) = \bigcup_{1 \leq a \leq \ell} P_a^{ij}$, where for each $1 \leq a \leq \ell$, the graph P_a^{ij} is $(\ell^{-1}, \boldsymbol{\varepsilon}(\ell))$-regular.*
3. *For all but $\delta t^3 \ell^3$ triads $P \in$ Triad($\mathcal{P}$) of density $d_{\mathcal{G}_P}(P) = \alpha_P \geq \alpha_0$, we have that P is (α_P, δ)-minimal w.r.t. $\mathcal{G}_P$.*

Moreover, there exists an algorithm which produces partition $\mathcal{P}$ in time $O(N^6)$.

While we only sketch the details, Theorem 2.5 follows from Theorem 2.4 by employing ideas considered in [29]. In particular, one can, in time $O(N^2)$, alter a partition $\mathcal{P}_{2.4}$ produced by Theorem 2.4 to obtain a partition $\mathcal{P}_{2.5}$ promised by Theorem 2.5. Indeed, first equitably distribute vertices of the "garbage class" V_0 of $\mathcal{P}_{2.4}$ into $V_1 \cup \cdots \cup V_t$. It is easy to show that this process interferes with the regularity/minimality of $\mathcal{P}_{2.4}$ by at most a measure of $o(1) \to 0$ as $N \to \infty$. Now, fix $1 \leq i < j \leq t$, where we suppose there are $r_{ij} \leq \ell_{ij} \leq \ell$ many $(\ell^{-1}, \boldsymbol{\varepsilon}(\ell))$-regular bipartite graphs P_a^{ij}, $1 \leq a \leq \ell_{ij}$. It is not hard to show that the union of these r_{ij} graphs is itself $(r_{ij}/\ell, r_{ij}\boldsymbol{\varepsilon}(\ell))$-regular, and therefore, $(r_{ij}/\ell, \ell\boldsymbol{\varepsilon}(\ell))$-regular. It then easily follows that the complement of this union (which contains P_0^{ij} and the $\ell_{ij} - r_{ij}$ many $(\ell^{-1}, \boldsymbol{\varepsilon}(\ell))$-irregular bipartite graphs P_a^{ij}) is $(1 - r_{ij}/\ell, \ell\boldsymbol{\varepsilon}(\ell))$-regular. As such, one "randomly" slices this complement into $\ell - r_{ij}$ new bipartite graphs R_a^{ij}, each of which is (almost surely) $(\ell^{-1}, 3\ell\boldsymbol{\varepsilon}(\ell))$-regular (see, [29, Lemma 30, p. 129]). Moreover, in time $O(N^2)$, this random slicing may be algorithmically derandomized using Lemma 3.8 of [19, p. 144]. Repeating this procedure for all $1 \leq i < j \leq t$ produces the partition $\mathcal{P}_{2.5}$ promised by Theorem 2.5. (Note that triads P which were originally (α_P, δ)-minimal w.r.t. $\mathcal{G}_P$ are unaffected by the process above since each bipartite graph of P was already $(\ell^{-1}, \boldsymbol{\varepsilon}(\ell))$-regular (cf. Remark 2.3).)

2.3. Counting lemma. We prove a hypergraph counting lemma compatible with our algorithmic regularity lemma, Theorem 2.4. In what follows, for a hypergraph $\mathcal{G}$ on vertex set V and an integer k, let $\mathcal{K}_k^{(3)}(\mathcal{G})$ denote the system of k-cliques in $\mathcal{G}$:

$$\mathcal{K}_k^{(3)}(\mathcal{G}) = \left\{ K \in \binom{V}{k} : \ K \text{ induces a clique } K_k^{(3)} \text{ of size } k \text{ in } \mathcal{G} \right\}.$$

Our counting lemma takes place in the following environment.

Setup 2.6. *Suppose*

1. $P = \bigcup_{1 \le i < j \le k} P^{ij}$ *is a k-partite graph and* $\mathcal{H} = \bigcup_{1 \le h < i < j \le k} \mathcal{H}^{hij} \subseteq \mathcal{K}_3^{(2)}(P)$ *is a k-partite 3-graph, each with k-partition* $V(P) = V(\mathcal{H}) = V_1 \cup \cdots \cup V_k$, $|V_1| = \cdots = |V_k| = n$.
2. *Each graph* P^{ij}, $1 \le i < j \le k$, *is* $(1/\ell, \varepsilon)$*-regular.*
3. *Each 3-graph* $\mathcal{H}^{hij} \subseteq \mathcal{K}_3^{(2)}(P^{hi} \cup P^{ij} \cup P^{hj})$, $1 \le h < i < j \le k$, *is* (α, δ)*-minimal w.r.t.* $P^{hi} \cup P^{ij} \cup P^{hj}$, *i.e.,* $d_{\mathcal{H}^{hij}}(P^{hi} \cup P^{ij} \cup P^{hj}) = \alpha$ *and* $|\mathcal{K}_{2,2,2}^{(3)}(\mathcal{H}^{hij})| \le \frac{\alpha^8}{\ell^{12}} \binom{n}{2}^3 (1 + \delta)$.

In the environment of Setup 2.6 and with appropriately chosen constants, we estimate $|\mathcal{K}_k^{(3)}(\mathcal{H})|$ with the following counting lemma.

Theorem 2.7 (counting lemma). *For all integers k and $\alpha > 0$ there exists $\delta_0 > 0$ so that for all $0 < \delta < \delta_0$ and integers ℓ there exists $\varepsilon > 0$ so that, with n sufficiently large, whenever* $\mathcal{H} = \bigcup_{1 \le h < i < j \le k} \mathcal{H}^{hij}$ *and* $P = \bigcup_{1 \le i < j \le k} P^{ij}$ *are as in Setup* 2.6 *with these constants,*

$$\left| \mathcal{K}_k^{(3)}(\mathcal{H}) \right| = \frac{\alpha^{\binom{k}{3}}}{\ell^{\binom{k}{2}}} n^k \left(1 \pm \delta^{\frac{1}{120k}}\right).$$

We mention that our error term is not optimal and is taken in this paper for convenience.

2.4. Application: Constructive fractional packings. In [1, 10] and, more recently, [3], a number of applications are given using the algorithmic version of Szemerédi's regularity lemma. It is likely that the work of the current paper allows some of these applications to be extended to a 3-uniform hypergraph setting. In what follows, however, we discuss an application of our work to a different problem, which concerns constructive fractional packings.

Let fixed 3-graph $\mathcal{F}$ be given. For a 3-graph $\mathcal{G}$, let $\binom{\mathcal{G}}{\mathcal{F}}$ denote the set of copies $\mathcal{F}_0$ of $\mathcal{F}$ contained in $\mathcal{G}$. An *$\mathcal{F}$-packing* of $\mathcal{G}$ is a collection of pairwise edge-disjoint elements of $\binom{\mathcal{G}}{\mathcal{F}}$. We denote by $\nu_{\mathcal{F}}(\mathcal{G})$ the maximum size of an $\mathcal{F}$-packing of $\mathcal{G}$. A *fractional $\mathcal{F}$-packing* of $\mathcal{G}$ is any function $\psi : \binom{\mathcal{G}}{\mathcal{F}} \to [0, 1]$ such that for every fixed edge $e \in \mathcal{G}$, we have $\sum_{\binom{\mathcal{G}}{\mathcal{F}} \ni \mathcal{F}_0 \ni e} \psi(\mathcal{F}_0) \le 1$. Then, $\nu_{\mathcal{F}}^*(\mathcal{G})$ is defined to be the maximum value of $\sum_{\mathcal{F}_0 \in \binom{\mathcal{G}}{\mathcal{F}}} \psi(\mathcal{F}_0)$ taken over all fractional $\mathcal{F}$-packings of $\mathcal{G}$. It is clear that $\nu_{\mathcal{F}}^*(\mathcal{G}) \ge \nu_{\mathcal{F}}(\mathcal{G})$ holds for all 3-graphs $\mathcal{G}$.

While computing $\nu_{\mathcal{F}}(\mathcal{G})$ is an NP-hard problem (cf. [9]), computing $\nu_{\mathcal{F}}^*(\mathcal{G})$ is known to be a linear programming problem (and hence can be done in polynomial time). Extending a result of the first and third author [19] for graphs, the current authors proved in [18] that

$$\nu_{\mathcal{F}}^*(\mathcal{G}) - \nu_{\mathcal{F}}(\mathcal{G}) = o(|V(\mathcal{G})|^3) \tag{16}$$

holds for all 3-graphs $\mathcal{G}$.

Theorems 2.4 and 2.7 may be combined to give the following constructive extension of (16).

Theorem 2.8. *For all fixed 3-graphs $\mathcal{F}$ and constants $\varepsilon > 0$, there exists N_0 so that for all 3-graphs $\mathcal{G}$ on $N > N_0$ vertices, an $\mathcal{F}$-packing of $\mathcal{G}$ of size $\nu_{\mathcal{F}}(\mathcal{G}) - \varepsilon N^3$ can be constructed in polynomial time.*

Proving Theorem 2.8 requires attention to a few technical details. We defer its proof to a forthcoming paper.

2.5. Organization of paper. The remainder of this paper is organized as follows. In section 3, we prove Theorem 2.4, the regularity lemma. In section 4, we present some facts on (α, δ)-minimality we need to prove Theorem 2.7, the counting lemma. In section 5, we prove Theorem 2.7. In section 7, we show how the upcoming Proposition 3.2 follows from our earlier work, Lemma 5.8 of [8].

3. Proof of Theorem 2.4.

Our proof of Theorem 2.4 follows the outline set forth in sections 1.1 and 1.2, but uses the hypergraph language established in this paper as well as in Frankl and Rödl [11]. We begin by reviewing some concepts from [11], beginning with the crucial concept of (δ, r)-regularity (compare this with the concept of our paper, (α, δ)-minimality; see Definition 2.2).

Definition 3.1. *Let $\delta > 0$ and integer r be given. We say 3-graph $\mathcal{H}$ is (δ, r)-*regular *w.r.t. graph P if for any sequence $\mathbf{Q}_r = (Q_1, \dots, Q_r)$ of subgraphs $Q_s \subseteq P$, $1 \leq s \leq r$,*

$$\left| \bigcup_{s=1}^{r} \mathcal{K}_3^{(2)}(Q_s) \right| > \delta \left| \mathcal{K}_3^{(2)}(P) \right| \quad \Longrightarrow \quad |d_{\mathcal{H}}(\mathbf{Q}_r) - d_{\mathcal{H}}(P)| < \delta,$$

where

$$d_{\mathcal{H}}(\mathbf{Q}_r) = \frac{\left| \mathcal{H} \cap \bigcup_{s=1}^{r} \mathcal{K}_3^{(2)}(Q_s) \right|}{\left| \bigcup_{s=1}^{r} \mathcal{K}_3^{(2)}(Q_s) \right|}$$

is the density *of $\mathbf{Q}_r$ w.r.t. $\mathcal{H}$. If $\mathcal{H}$ is not (δ, r)-regular w.r.t. P, we say $\mathcal{H}$ is (δ, r)-*irregular *w.r.t. P, and in this case, any vector $\mathbf{Q}_r = (Q_1, \dots, Q_r)$ violating the regularity condition above is said to be a* witness *of the (δ, r)-irregularity of $\mathcal{H}$ w.r.t. P.*

(In the definition above, when $\bigcup_{s=1}^{r} \mathcal{K}_3^{(2)}(Q_s) = \emptyset$, we shall define $d_{\mathcal{H}}(\mathbf{Q}_r) = \emptyset$.)

The paper [11] also extends the notion of the graph "index" (cf. (5)) to hypergraphs. For a 3-graph $\mathcal{G}$ with $(\ell, t, \gamma, \varepsilon)$-partition $\mathcal{P}$, define the *index* of $\mathcal{P}$ w.r.t. $\mathcal{G}$ as

$$\operatorname{ind} \mathcal{P} = \frac{1}{N^3} \sum_{P \in \mathrm{Triad}(\mathcal{P})} d_{\mathcal{G}_P}^2(P) \left| \mathcal{K}_3^{(2)}(P) \right|. \tag{17}$$

Similarly to (5), it is easy to see that $\operatorname{ind} \mathcal{P} \leq 1$.

We need one final preparation before proceeding to the proof of Theorem 2.4. As in (14), where we defined the family $\mathrm{Triad}_{(\alpha_0,\delta)\text{-exc}}(\mathcal{P}_s)$ of all "excessive" triads, we define

$$\mathrm{Triad}_{(\delta,r)\text{-irr}}(\mathcal{P}) = \left\{ P \in \mathrm{Triad}(\mathcal{P}) : \mathcal{G}_P = \mathcal{G} \cap \mathcal{K}_3^{(2)}(P) \text{ is } (\delta, r)\text{-irregular w.r.t. } P \right\} \tag{18}$$

as the family of all "irregular" triads. (By our definitions, a triad $P \in \mathrm{Triad}_{(\delta,r)\text{-irr}}(\mathcal{P})$ is allowed to have a bipartite graph P_a^{ij} which is not (ℓ^{-1}, ε)-regular (where $1 \leq i < j \leq t$ and $1 \leq a \leq \ell_{ij}$), while a triad $P \in \mathrm{Triad}_{(\alpha_0,\delta)\text{-exc}}(\mathcal{P})$ is not.)

Now, to prove Theorem 2.4 according to the outline from the introduction, we need Propositions 3.2 and 3.4, and we begin by presenting the former. Proposition 3.2 below generalizes the common implication of (2) and (3) (which said that, for graphs, ε-regularity implies $K_{2,2}^{(2)}$-minimality). Proposition 3.2 asserts that if $\mathcal{H}$ is (δ_A, r)-regular w.r.t. P, then it is also (α, δ_B)-minimal w.r.t. P (or, as we shall apply it, if $\mathcal{H}$ is (α, δ_B)-excessive w.r.t. P, then it is also (δ_A, r)-irregular w.r.t. P).

PROPOSITION 3.2 (excessiveness ⇒ irregularity). *For all α and δ_B, there exists $\delta_A > 0$ so that for all integers ℓ, there exist $\varepsilon > 0$ and integer r so that whenever $\mathcal{H}$ and P satisfy the hypothesis of Setup* 2.1 *with constants α, ℓ, and ε and n sufficiently large, then the following hold:*

1. *if $\mathcal{H}$ is (α, δ_B)-excessive w.r.t. P, then $\mathcal{H}$ is (δ_A, r)-irregular w.r.t. P;*
2. *moreover, one may construct, in time $O(n^5)$, a witness $\mathbf{Q}_r$ of the (δ_A, r)-irregularity of $\mathcal{H}$ w.r.t. P.*

Remark 3.3. Statement (1) of Proposition 3.2 can be inferred from Lemma 3.1.1, the counting lemma, of [27]. Indeed, let $\mathcal{H}$ and P be given as in Setup 2.1 with suitably chosen constants, where $\mathcal{H}$ is (δ, r)-regular w.r.t. P. The counting lemma implies that $\mathcal{H}$ contains (asymptotically) the same number of copies of any fixed subhypergraph (and, in particular, $K_{2,2,2}^{(3)}$) as is expected in the corresponding random 3-partite hypergraph. (Here, the corresponding random hypergraph is the binomial random subset of triangles from P including each element of $\mathcal{K}_3^{(2)}(P)$ independently with probability α.)

In this paper, we shall need statement 2 of Proposition 3.2. This statement was proved, in slightly different language, in Lemma 5.8 (Algorithm A) of [8]. For completeness, we prove that Proposition 3.2 follows from Lemma 5.8 (Algorithm A) of [8] and give this proof in section 7.

We remind the reader (cf. (8)) that the converse of Proposition 3.2 is, in general, not true (see [8] for details).

In our proof of the hypergraph regularity lemma, Theorem 2.4, we will also use Proposition 3.4 below. This proposition will extend the "index-increasing" step (7) stated in the introduction from graphs to hypergraphs. To motivate this proposition, note that we may count the number of copies of $K_{2,2,2}^{(3)}$ spanned in each triad P of an $(\ell, t, \gamma, \varepsilon)$-partition $\mathcal{P}$. For those triads P with "excessively many" copies, Proposition 3.2 asserts P is "irregular" and builds a corresponding witness $\mathbf{Q}_P$. If many of these triads P are found, in this way, to be irregular, then the following proposition will construct a new partition $\mathcal{P}'$ from $\mathcal{P}$ and the witnesses $\mathbf{Q}_P$, where $\mathcal{P}'$ has index nontrivially larger than that of $\mathcal{P}$. (In what follows, one may therefore think of the subfamily $\mathcal{T}^*$ as a class of suitably "excessive" triads.)

PROPOSITION 3.4 (inflating the index). *Let constants δ and γ be given as well as functions $\boldsymbol{\varepsilon} : \mathbb{N}^+ \to (0, 1)$ and $\boldsymbol{r} : \mathbb{N}^+ \to \mathbb{N}^+$ and integers ℓ_{old} and t_{old}. There exist constants*

$$L_0 = L_0(\delta, \gamma, \boldsymbol{\varepsilon}, \boldsymbol{r}, \ell_{\text{old}}, t_{\text{old}}), \quad T_0 = T_0(\delta, \gamma, \boldsymbol{\varepsilon}, \boldsymbol{r}, \ell_{\text{old}}, t_{\text{old}}), \quad N_0 = N_0(\delta, \gamma, \boldsymbol{\varepsilon}, \boldsymbol{r}, \ell_{\text{old}}, t_{\text{old}})$$

so that the following holds.

Suppose $\mathcal{G}$ is a 3-graph on $N > N_0$ vertices with $(\ell_{\text{old}}, t_{\text{old}}, \gamma, \boldsymbol{\varepsilon}(\ell_{\text{old}}))$-partition $\mathcal{P}_{\text{old}}$, and suppose $\mathcal{T}^* = \text{Triad}^*_{(\delta, \boldsymbol{r}(\ell_{\text{old}}))\text{-irr}}(\mathcal{P}_{\text{old}})$ is a subfamily of the collection of all $(\delta, \boldsymbol{r}(\ell_{\text{old}}))$-irregular triads $\text{Triad}_{(\delta, \boldsymbol{r}(\ell_{\text{old}}))\text{-irr}}(\mathcal{P}_{\text{old}})$ (see the notation in (18)*), satisfying the following properties:*

1. *for each triad P of the subfamily $\mathcal{T}^*$, one is* given *witness $\mathbf{Q}_{\boldsymbol{r}(\ell_{\text{old}}), P}$ of the $(\delta, \boldsymbol{r}(\ell_{\text{old}}))$-irregularity of $\mathcal{G}_P = \mathcal{G} \cap \mathcal{K}_3^{(2)}(P)$ w.r.t. P;*

2.

$$\sum_{P\in\mathcal{T}^*}\left|\mathcal{K}_3^{(2)}(P)\right|\geq\delta N^3.$$

Then,

(a) *there exists an equitable* $(\ell_{\mathrm{new}},t_{\mathrm{new}},\gamma,\boldsymbol{\varepsilon}(\ell_{\mathrm{new}}))$*-partition* $\mathcal{P}_{\mathrm{new}}$ *of* $\binom{V}{2}$ *for which*

$$\mathrm{ind}\,\mathcal{P}_{\mathrm{new}}\geq\mathrm{ind}\,\mathcal{P}_{\mathrm{old}}+\frac{\delta^4}{2},\tag{19}$$

where $\ell_{\mathrm{old}}\leq\ell_{\mathrm{new}}\leq L_0$ *and* $t_{\mathrm{old}}\leq t_{\mathrm{new}}\leq T_0$*. Moreover,*

(b) *there exists an algorithm which in time* $O(N^2)$ *constructs the partition* $\mathcal{P}_{\mathrm{new}}$ *above from* $\mathcal{P}_{\mathrm{old}}$ *and the given collection of witnesses* $\{\mathbf{Q}_{\boldsymbol{r}(\ell_{\mathrm{old}}),P}:P\in\mathcal{T}^*\}$*.*

The proof of Proposition 3.4 is given in [11] with no focus on it being algorithmic. We shall not give a formal proof of Proposition 3.4, but we will now sketch a proof to indicate how its algorithmic part is obtained.

The approach in [11] is similar to Szemerédi's [40]. Consider the Venn diagram of the intersections of the witnesses $\mathbf{Q}_P=\mathbf{Q}_{\boldsymbol{r}(\ell_{\mathrm{old}}),P}$ over all $P\in\mathcal{T}^*$. In (1) of the hypothesis in Proposition 3.4, these witnesses are given to us. (In Szemerédi [40], these witnesses were subsets of vertices; here, the witnesses are (r-tuples of) subsets of pairs.) This Venn diagram has at most

$$2^{|\mathcal{T}^*|\boldsymbol{r}(\ell_{\mathrm{old}})}\leq2^{t_{\mathrm{old}}^3\ell_{\mathrm{old}}^3\boldsymbol{r}(\ell_{\mathrm{old}})}$$

regions (this number is independent of N), where each region is a bipartite graph. This Venn diagram defines a refinement $\mathcal{P}'_{\mathrm{old}}$ of $\mathcal{P}_{\mathrm{old}}$, so that $\mathcal{P}'_{\mathrm{old}}$ is itself a partition. (The index of $\mathcal{P}'_{\mathrm{old}}$ is larger than that of $\mathcal{P}_{\mathrm{old}}$ on account of the fact that in (2), we assumed "many" triangles were lost to irregular triads in $\mathcal{P}_{\mathrm{old}}$.) The bipartite graphs of $\mathcal{P}'_{\mathrm{old}}$ may not be "regular," so we apply Szemerédi's regularity lemma to each. (The regularity lemma is known to be algorithmic by the result of Alon et al. [1, 10].) The resulting (regular) bipartite graphs may have differing densities, so we "randomly slice" each into thinner "equidense" layers (this is the same idea we discussed earlier after stating Theorem 2.5). This random slicing is derandomized, however, in Lemma 6 (p. 17) of Haxell and Rödl [19]. (These latter two refinements of $\mathcal{P}'_{\mathrm{old}}$ are done at no real cost to the index of $\mathcal{P}'_{\mathrm{old}}$.) The process above produces the partition $\mathcal{P}_{\mathrm{new}}$. For the formal details of this outline, see Lemmas 3.9 and 3.10 of [11, pp. 145 and 149] and Lemma 6 of [19].

We now give the proof of Theorem 2.4 using Propositions 3.2 and 3.4 and following the outline of the introduction.

3.1. Proof of Theorem 2.4.

For quick reference on the proof that follows, we provide a flow-chart in Figure 1. In what immediately follows, we formally describe all parameters used in our argument.

3.1.1. Constants of Theorem 2.4.

Let $\alpha_0,\delta,\gamma>0$ be given as well as function $\boldsymbol{\varepsilon}:\mathbb{N}\to(0,1)$. For simplicity, let $\ell_0=1$ (most applications of the original Frankl–Rödl hypergraph regularity lemma (cf. Theorem 3.5 of [11]) take $\ell_0=1$). Let integer t_0 be given. Let us now briefly describe a few auxiliary constants we will need momentarily.

For α_0 and $\delta_B=\delta$, let

$$\delta_A^{(3.2)}=\delta_A^{(3.2)}(\alpha_0,\delta)\tag{20}$$

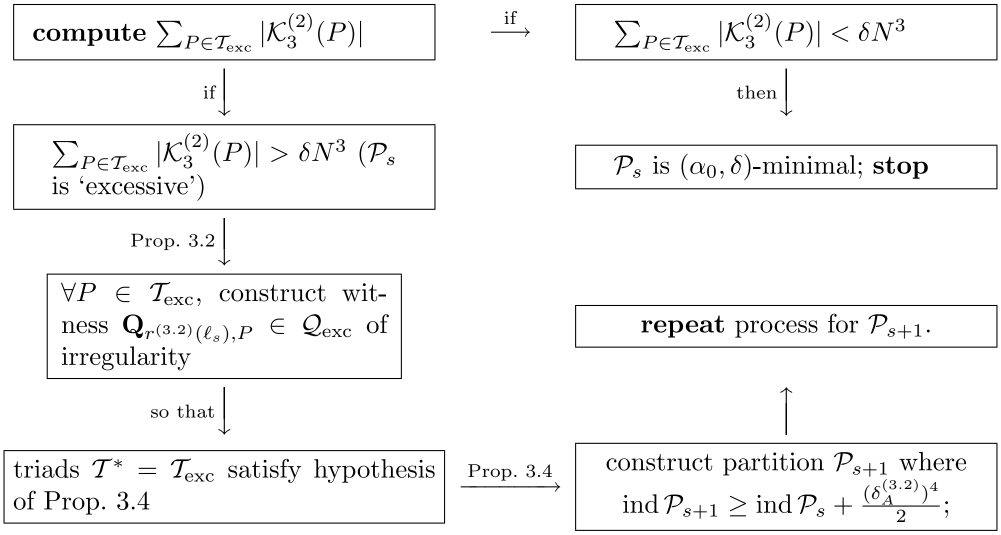

compute $\sum_{P\in\mathcal{T}_{\rm exc}}|\mathcal{K}_3^{(2)}(P)|$
if
$\sum_{P\in\mathcal{T}_{\rm exc}}|\mathcal{K}_3^{(2)}(P)| < \delta N^3$
if
then
$\sum_{P\in\mathcal{T}_{\rm exc}}|\mathcal{K}_3^{(2)}(P)| > \delta N^3$ ($\mathcal{P}_s$ is ‘excessive’)
$\mathcal{P}_s$ is (α_0,δ)-minimal; **stop**
Prop. 3.2
$\forall P \in \mathcal{T}_{\rm exc}$, construct witness $\mathbf{Q}_{r^{(3.2)}(\ell_s),P} \in \mathcal{Q}_{\rm exc}$ of irregularity
repeat process for $\mathcal{P}_{s+1}$.
so that
triads $\mathcal{T}^* = \mathcal{T}_{\rm exc}$ satisfy hypothesis of Prop. 3.4
Prop. 3.4
construct partition $\mathcal{P}_{s+1}$ where $\operatorname{ind}\mathcal{P}_{s+1} \geq \operatorname{ind}\mathcal{P}_s + \frac{(\delta_A^{(3.2)})^4}{2}$;

FIG. 1. *Flow chart for the proof of Theorem* 2.4. *Here,* $\mathcal{P}_s$ *is a given* $(\ell_s, t_s, \gamma, \boldsymbol{\varepsilon}(\ell_s))$*-partition,* $\delta_A^{(3.2)} \leq \delta$ *(cf.* (21)*), and* $\mathcal{T}^* = \mathcal{T}_{\rm exc}$ *is defined in* (28).

be the constant guaranteed by Proposition 3.2 where we may assume, without loss of generality, that

$$\delta_A^{(3.2)} \leq \delta. \tag{21}$$

For an integer variable $\ell \in \mathbb{N}$, let

$$\varepsilon^{(3.2)}(\ell) = \varepsilon^{(3.2)}(\alpha_0, \delta, \delta_A^{(3.2)}, \ell) \quad \text{and} \quad r^{(3.2)}(\ell) = r^{(3.2)}(\alpha_0, \delta, \delta_A^{(3.2)}, \ell) \tag{22}$$

be the *functions* guaranteed by Proposition 3.2. Without loss of generality, we shall assume the given function $\boldsymbol{\varepsilon}$ satisfies

$$\boldsymbol{\varepsilon}(\ell) \leq \varepsilon^{(3.2)}(\ell) \tag{23}$$

for every integer $\ell \in \mathbb{N}$.

Theorem 2.4 promises constants L_0, T_0, and N_0, which we shall now formally describe (although they will be easier to see later in context). With $\gamma > 0$ (given as above), $\delta_A^{(3.2)}$ (given in (20)), function $\boldsymbol{\varepsilon}(\ell) \leq \varepsilon^{(3.2)}(\ell)$ (given as in (23)), and $r^{(3.2)}(\ell)$ (given in (22)) fixed, and for arbitrary integers $\ell_{\rm old}$ and $t_{\rm old}$, Proposition 3.4 promises constants

$$L_0(\ell_{\rm old}, t_{\rm old}) = L_0^{(3.4)}(\gamma, \delta_A^{(3.2)}, \boldsymbol{\varepsilon}(\ell), r^{(3.2)}(\ell), \ell_{\rm old}, t_{\rm old}),$$

$$T_0(\ell_{\rm old}, t_{\rm old}) = T_0^{(3.4)}(\gamma, \delta_A^{(3.2)}, \boldsymbol{\varepsilon}(\ell), r^{(3.2)}(\ell), \ell_{\rm old}, t_{\rm old}),$$

and

$$N_0(\ell_{\rm old}, t_{\rm old}) = N_0^{(3.4)}(\gamma, \delta_A^{(3.2)}, \boldsymbol{\varepsilon}(\ell), r^{(3.2)}(\ell), \ell_{\rm old}, t_{\rm old}).$$

We successively define constants $L_0^{(i)}$, $T_0^{(i)}$, and $N_0^{(i)}$, $0 \leq i \leq 2/(\delta_A^{(3.2)})^4$, as follows: with already given constants $\ell_0 = 1$ and t_0, set

$$L_0^{(0)} = L_0(\ell_0 = 1, t_0), \quad T_0^{(0)} = T_0(\ell_0 = 1, t_0), \quad N_0^{(0)} = N_0(\ell_0 = 1, t_0).$$

For $1 \le i \le 2/(\delta_A^{(3.2)})^4$, set

(24)
$$L_0^{(i)} = L_0(L_0^{(i-1)}, T_0^{(i-1)}), \quad T_0^{(i)} = T_0(L_0^{(i-1)}, T_0^{(i-1)}), \quad N_0^{(i)} = N_0(L_0^{(i-1)}, T_0^{(i-1)}).$$

Then, the constants L_0, T_0, and N_0 of Theorem 2.4 are given by

$$L_0 = L_0^{(i_*)}, \quad T_0 = T_0^{(i_*)}, \quad N_0 = N_0^{(i_*)}, \tag{25}$$

where

$$i_* = \left\lfloor \frac{2}{(\delta_A^{(3.2)})^4} \right\rfloor .$$

This concludes our discussion of the constants.

3.1.2. The argument. Let 3-graph $\mathcal{G}$ be given on sufficiently large vertex set V, $|V| = N$. We are going to construct, in time $O(N^6)$, an (α_0, δ)-minimal and equitable $(\ell, t, \gamma, \boldsymbol{\varepsilon}(\ell))$ partition $\mathcal{P}$ for $\mathcal{G}$ for which $1 \le \ell \le L_0$ and $t_0 \le t \le T_0$ for L_0 and T_0 given in (25). The main idea of the proof is outlined in the introduction as well as the flow-chart in Figure 1.

Start with the partition $\mathcal{P}_1$, taken as $K(V_0, V_1, \dots, V_{t_0})$, where $V(\mathcal{G}) = V_0 \cup V_1 \cup \dots \cup V_{t_0}$ is any vertex partition with $|V_0| < t_0$ and $|V_1| = \dots = |V_{t_0}|$ (so that $\mathcal{B}$ consists of the $\binom{t_0}{2}$ complete bipartite graphs $K[V_i, V_j]$, $1 \le i < j \le t_0$). Then $\mathcal{P}_1$ is an equitable $(\ell_0 = 1, t_0, \gamma, \boldsymbol{\varepsilon}(\ell_0))$-partition since, in fact, it is an equitable $(\ell_0 = 1, t_0, 0, \varepsilon)$-partition for any $\varepsilon > 0$.

For an integer $1 \le s < 2/(\delta_A^{(3.2)})^4$ (this upper bound will become clearer within the context of the proof), assume $\mathcal{P}_1, \dots, \mathcal{P}_s$ are constructed partitions where $\mathcal{P}_s$ is an equitable $(\ell_s, t_s, \gamma, \boldsymbol{\varepsilon}(\ell_s))$-partition of $\binom{V}{2}$ where

$$1 \le \ell_s \le L_0^{(s-1)} \quad \text{and} \quad t_0 \le t_s \le T_0^{(s-1)} \tag{26}$$

for the constants $L_0^{(s-1)}$ and $T_0^{(s-1)}$ defined in (24). The main idea here is similar to that used by Szemerédi [40]. We shall either verify that $\mathcal{P}_s$ is (α_0, δ)-minimal or else construct a new partition $\mathcal{P}_{s+1}$ whose index (cf. (17)) is larger than that of $\mathcal{P}_s$. Moreover, we show that these steps can be carried out in time $O(N^6)$.

Our first step in the algorithm is to compute, for the partition $\mathcal{P}_s$ above, the sum (cf. (15))

$$\sum_{P \in \mathrm{Triad}_{(\alpha_0,\delta)\text{-exc}}(\mathcal{P}_s)} \left| \mathcal{K}_3^{(2)}(P) \right| . \tag{27}$$

The central operation in computing this sum consists of identifying

$$\mathcal{T}_{\mathrm{exc}} \overset{\mathrm{def}}{=} \mathrm{Triad}_{(\alpha_0,\delta)\text{-exc}}(\mathcal{P}_s) \subseteq \mathrm{Triad}(\mathcal{P}_s). \tag{28}$$

Indeed, for each of the $\binom{t_s}{3}\ell_s^3$ triads $P \in \mathrm{Triad}(\mathcal{P}_s)$, we count the number of $K_{2,2,2}^{(3)}$'s appearing in $\mathcal{G}_P = \mathcal{G} \cap \mathcal{K}_3^{(2)}(P)$. As such, computing (27) has complexity $O(N^6)$.

Upon computing the sum in (27), two outcomes can occur (cf. (28)):

$$\sum_{P \in \mathcal{T}_{\mathrm{exc}}} \left| \mathcal{K}_3^{(2)}(P) \right| < \delta N^3 \quad \overset{(15)}{\Longrightarrow} \quad \mathcal{P}_s \text{ is } (\alpha_0, \delta)\text{-minimal} \tag{29}$$

or

$$\sum_{P\in\mathcal{T}_{\mathrm{exc}}}\left|\mathcal{K}_3^{(2)}(P)\right|\geq\delta N^3\quad\overset{(15)}{\Longrightarrow}\quad\mathcal{P}_s\text{ is }(\alpha_0,\delta)\text{-excessive.}\tag{30}$$

If we determine that (29) occurs, then we are done. Indeed, $\mathcal{P}_s$ is the (α_0,δ)-minimal and equitable $(\ell_s,t_s,\gamma,\boldsymbol{\varepsilon}(\ell_s))$-partition we wanted to construct. This situation is the analogue of step $(1')$ in the introduction.

Otherwise, we determine that (30) occurs and we are in a situation similar to step $(2')$ in the outline of the introduction where we increased the graph index in (7). Here, we want to show that from the large sum in (30), we may construct a new and equitable $(\ell_{s+1},t_{s+1},\gamma,\boldsymbol{\varepsilon}(\ell_{s+1}))$-partition $\mathcal{P}_{s+1}$ of $\binom{V}{2}$ whose index is nontrivially larger than ind $\mathcal{P}_s$. Moreover, we want to show that the new parameters ℓ_{s+1} and t_{s+1} satisfy $\ell_{s+1}\leq L_0^{(s)}$ and $t_{s+1}\leq T_0^{(s)}$ (cf. (24)). Proposition 3.4 is precisely the tool to do this.

Before we can apply Proposition 3.4 to the partition $\mathcal{P}_s$, we need to show that its hypothesis is met. To that end, set

$$\mathcal{T}^*=\mathrm{Triad}^*_{(\delta_A^{(3.2)},r^{(3.2)}(\ell_s))\text{-irr}}(\mathcal{P}_s)=\mathrm{Triad}_{(\alpha_0,\delta)\text{-exc}}(\mathcal{P}_s)=\mathcal{T}_{\mathrm{exc}}.\tag{31}$$

Proposition 3.2 will guarantee that for each excessive triad $P\in\mathcal{T}_{\mathrm{exc}}$,

(32) *one may construct, in time $O((N/t_s)^5)$, a witness $\mathbf{Q}_{\boldsymbol{r}(\ell_s),P}$*
of the $(\delta_A^{(3.2)},r^{(3.2)}(\ell_s))$-irregularity of $\mathcal{G}_P=\mathcal{G}\cap\mathcal{K}_3^{(2)}(P)$ w.r.t. P,

so that, in particular (cf. (18)),

$$\mathcal{T}^*=\mathcal{T}_{\mathrm{exc}}\subseteq\mathrm{Triad}_{(\delta_A^{(3.2)},r^{(3.2)}(\ell_s))\text{-irr}}(\mathcal{P}_s).$$

Indeed, fix triad $P=P_a^{ij}\cup P_b^{jk}\cup P_c^{ik}\in\mathrm{Triad}_{(\alpha_0,\delta)\text{-exc}}(\mathcal{P}_s)$, where $1\leq i<j<k\leq t_s$ and $1\leq a,b,c\leq\ell_s$ (so that P has 3-partition $V_i\cup V_j\cup V_k$, where $\lfloor N/t_s\rfloor\leq|V_i|,|V_j|,|V_k|\leq\lceil N/t_s\rceil$). We intend to apply Proposition 3.2 to graph P and hypergraph $\mathcal{H}=\mathcal{G}_P=\mathcal{G}\cap\mathcal{K}_3(P)$. Note that the constants involved with P and $\mathcal{G}_P$ meet the requirements of Proposition 3.2. Indeed, $P\in\mathcal{T}_{\mathrm{exc}}$ means that with $\delta_B=\delta$, $\mathcal{G}_P$ is (α_P,δ_B)-excessive w.r.t. P where $\alpha_P=|\mathcal{G}_P|/|\mathcal{K}_3^{(2)}(P)|\geq\alpha_0$ where α_0 was given in the beginning of the proof (cf. Definition 2.2 and Remark 2.3). Moreover, each of the bipartite graphs P_a^{ij}, P_b^{jk}, P_c^{ik} is $(1/\ell_s,\boldsymbol{\varepsilon}(\ell_s))$-regular, where by (23), $\boldsymbol{\varepsilon}(\ell_s)\leq\varepsilon^{(3.2)}(\ell_s)$, where $\varepsilon^{(3.2)}(\ell_s)$ is chosen in (22) sufficiently small to apply Proposition 3.2. Therefore, statement (2) of Proposition 3.2 guarantees a $O((N/t_s)^5)$-time algorithm which constructs a witness $\mathbf{Q}_{r^{(3.2)}(\ell_s),P}$ of the $(\delta_A^{(3.2)},r^{(3.2)}(\ell_s))$-irregularity of $\mathcal{G}_P$ w.r.t. P.

Continuing, (32) implies that statement (1) in the hypothesis of Proposition 3.4 is met by the partition $\mathcal{P}_s$. To see that statement (2) in the hypothesis of Proposition 3.4 is also met, we return to the large sum in (30) (cf. (31)) to see

$$\sum_{P\in\mathcal{T}^*}\left|\mathcal{K}_3^{(2)}(P)\right|\geq\delta N^3\overset{(21)}{\geq}\delta_A^{(3.2)}N^3.$$

Thus, Proposition 3.4 applies to the partition $\mathcal{P}_s$.

Proposition 3.4 constructs, in time $O(N^2)$, an equitable $(\ell_{s+1}, t_{s+1}, \gamma, \boldsymbol{\varepsilon}(\ell_{s+1}))$-partition $\mathcal{P}_{s+1}$ of $\binom{V}{2}$ for which

$$\text{ind}\,\mathcal{P}_{s+1} \geq \text{ind}\,\mathcal{P}_s + \frac{\left(\delta_A^{(3.2)}\right)^4}{2} \tag{33}$$

and for which

$$\ell_{s+1} \leq L_0(\ell_s, t_s) \overset{(26)}{\leq} L_0(L_0^{(s-1)}, T_0^{(s-1)}) \overset{(24)}{=} L_0^{(s)}$$

and

$$t_0 \leq t_{s+1} \leq T_0(\ell_s, t_s) \overset{(26)}{\leq} T_0(L_0^{(s-1)}, T_0^{(s-1)}) \overset{(24)}{=} T_0^{(s)}. \tag{34}$$

Now, the proof of Theorem 2.4 is essentially complete. Indeed, by (33), we can repeat the constructive procedure above at most $2/(\delta_A^{(3.2)})^4$ times (the index cannot exceed one), in which case, for some iteration, we must have constructed an (α_0, δ)-minimal and equitable $(\ell, t, \gamma, \boldsymbol{\varepsilon}(\ell))$-partition $\mathcal{P}$, where $\ell \leq L_0$ and $t_0 \leq t \leq T_0$ for L_0 and T_0 defined in (25).

As a final note, it is easy to see that the construction above is completed in time $O(N^6)$. Indeed, the application of Proposition 3.2, which constructs witnesses $\mathbf{Q}_{r^{(3.2)}(\ell_s),P}$ for each $P \in \mathcal{T}^* = \mathcal{T}_{\text{exc}}$, contributes $O(N^5)$ complexity. The application of Proposition 3.4, which constructs the new partition $\mathcal{P}_{s+1}$, contributes $O(N^2)$ complexity. As such, the complexity $O(N^6)$ of the algorithm is generated by counting copies of $K_{2,2,2}^{(3)}$ in (27) (by which we determine whether or not partitions $\mathcal{P}$ are (α_0, δ)-minimal). This completes the proof of Theorem 2.4.

4. Properties of (α, δ)-minimality.

It is well known that, in certain hypergraph contexts, having the (asymptotic) minimum number of copies of $K_{2,2,2}^{(3)}$ implies various other properties (see [5, 22]). In our proof of the counting lemma, Theorem 2.7, we shall need some of these properties for the context of Setup 2.1[1] (these properties are stated as upcoming the Propositions 4.1–4.6). At the end of this section, we sketch the proofs of the upcoming Propositions 4.1–4.6.

Recall that we asserted in (13) that a hypergraph $\mathcal{H}$ (with graph P) as in Setup 2.1 will contain at least $\sim (\alpha^8/\ell^{12})\binom{n}{2}^3$ many copies of $K_{2,2,2}^{(3)}$. If $\mathcal{H}$ has at most $\sim (\alpha^8/\ell^{12})\binom{n}{2}^3$ many such copies, then we defined (in Definition 2.2) $\mathcal{H}$ to be (α, δ)-minimal w.r.t. P. In this section, we also wish to consider a notion of minimality for the frequency of the subhypergraph $K_{1,2,2}^{(3)}$, where $K_{1,2,2}^{(3)}$ is the complete 3-uniform 3-partite hypergraph whose vertex partition classes have sizes 1, 2, and 2. To that end, with $\mathcal{H}$ and P given as in Setup 2.1, let

$$\mathcal{K}_{1,2,2}^{(3)}(\mathcal{H}) = \Bigg\{ (\{x\}, \{a,b\}, \{u,v\}) :\ x \in V_1,\ \{a,b\} \in \binom{V_2}{2},\ \{u,v\} \in \binom{V_3}{2},$$
$$\{x\} \cup \{a\} \cup \{b\} \cup \{u\} \cup \{v\} \text{ induces } K_{1,2,2}^{(3)} \text{ in } \mathcal{H} \Bigg\}.$$

[1] Setup 2.1 allows for the possibility that $1/\ell \ll \delta$, while the opposite situation $\delta \ll 1/\ell$ is considered in [5, 22]. The proofs we sketch for Propositions 4.1–4.6 are similar, nonetheless, to the proofs given in [5, 22].

Note that this family is different from, say,

$$\mathcal{K}^{(3)}_{2,2,1}(\mathcal{H}) = \left\{ (\{x,y\},\{a,b\},\{u\}) :\ \{x,y\} \in \binom{V_1}{2},\ \{a,b\} \in \binom{V_2}{2},\ u \in V_3, \right.$$
$$\left. \{x\} \cup \{y\} \cup \{a\} \cup \{b\} \cup \{u\} \text{ induces } K^{(3)}_{2,2,1} \text{ in } \mathcal{H} \right\},$$

and different from $\mathcal{K}^{(3)}_{2,1,2}(\mathcal{H})$, which is defined similarly. We proceed to state Propositions 4.1–4.6, and conclude this section with (sketches of) their proofs.

Proposition 4.1. *With given constants α, δ, and ℓ, sufficiently small $\varepsilon = \varepsilon(\alpha,\delta,\ell) > 0$, and sufficiently large $n = n(\alpha,\delta,\ell,\varepsilon)$, suppose $\mathcal{H}$ and P are as in Setup* 2.1. *Then*

$$\left|\mathcal{K}^{(3)}_{1,2,2}(\mathcal{H})\right| \geq \frac{\alpha^4}{\ell^8} n \binom{n}{2}^2 \left(1 - \varepsilon^{1/10}\right) \quad \textit{and} \quad \left|\mathcal{K}^{(3)}_{2,2,2}(\mathcal{H})\right| \geq \frac{\alpha^8}{\ell^{12}} \binom{n}{2}^3 \left(1 - \varepsilon^{1/10}\right).$$

The following definition is now motivated (cf. Definition 2.2).

Definition 4.2. *With $\mathcal{H}$ and P given as in Setup* 2.1, *we say $\mathcal{H}$ is $(\alpha,\delta)_1$-*minimal *w.r.t. P if*

$$\left|\mathcal{K}^{(3)}_{1,2,2}(\mathcal{H})\right| \leq \frac{\alpha^4}{\ell^8} n \binom{n}{2}^2 (1 + \delta).$$

Remark 4.3. For a clear distiction in this section, we shall refer to the original (α,δ)-minimality as $(\alpha,\delta)_2$-*minimality.*

The following proposition relates $(\alpha,\delta)_2$-minimality with $(\alpha,\delta)_1$-minimality.

Proposition 4.4. *With given constants α, δ, and ℓ, sufficiently small $\varepsilon = \varepsilon(\alpha,\delta,\ell) > 0$, and sufficiently large $n = n(\alpha,\delta,\ell,\varepsilon)$, suppose $\mathcal{H}$ and P are as in Setup* 2.1. *If $\mathcal{H}$ is $(\alpha,\delta)_2$-minimal w.r.t. P, then $\mathcal{H}$ is also $(\alpha,\delta)_1$-minimal w.r.t. P. In particular, if $\mathcal{H}$ is $(\alpha,\delta)_2$-minimal w.r.t. P, then all of the following inequalities hold:*

$$\left|\mathcal{K}^{(3)}_{1,2,2}(\mathcal{H})\right|, \left|\mathcal{K}^{(3)}_{2,1,2}(\mathcal{H})\right|, \left|\mathcal{K}^{(3)}_{2,2,1}(\mathcal{H})\right| \leq \frac{\alpha^4}{\ell^8} n \binom{n}{2}^2 (1 + \delta). \tag{35}$$

We sketch the proof of Proposition 4.4 at the end of this section.

To present Propositions 4.5 and 4.6, we require some notation. With $\mathcal{H}$ and P given as in Setup 2.1 and $x \in V$ an arbitrary vertex, let

$$L_x = \Big\{ \{u,v\} \in P :\ \{x,u,v\} \in \mathcal{H} \Big\} \tag{36}$$

denote the *link* of x. For vertices $x, y \in V$, let

$$L_{xy} = L_x \cap L_y \tag{37}$$

denote the *colink* of x and y. For Propositions 4.5 and 4.6, we also consider the following supplemental notation. For $x \in V$, let

$$P_x = \Big\{ \{u,v\} \in P :\ \{x,u,v\} \in \mathcal{K}^{(2)}_3(P) \Big\} = P[N_P(x)]$$

be the subgraph of P induced on the P-neighborhood of x. For vertices $x, y \in V_1$, let

$$P_{xy} = P_x \cap P_y = P[N_P(x,y)]. \tag{38}$$

In the context of Setup 2.1, it is well known (cf. Fact 1.4 from [26]) from the (ℓ^{-1},ε)-regularity of P^{12}, P^{23}, and P^{13} that all but $4\varepsilon n$ vertices $x\in V_1$ ($8\varepsilon\binom{n}{2}$ pairs $\{x,y\}\in\binom{V_1}{2}$) satisfy

$$\text{(39)}\quad \deg_{P^{1j}}(x)=|N_{P^{1j}}(x)|=\left(\frac{1}{\ell}\pm\varepsilon\right)n,\quad j=2,3,$$
$$\left(\deg_{P^{1j}}(x,y)=|N_{P^{1j}}(x,y)|=\left(\frac{1}{\ell}\pm\varepsilon\right)^2 n,\quad j=2,3\right).$$

As such, with ε sufficiently small (say, $0<\varepsilon\le 1/(2\ell)^2$), all but $4\varepsilon n$ vertices $x\in V_1$ ($8\varepsilon\binom{n}{2}$ pairs $\{x,y\}\in\binom{V_1}{2}$) satisfy

$$\text{(40)}\quad |P_x|=\left(\frac{1}{\ell}\pm\varepsilon\right)\deg_{P^{12}}(x)\cdot\deg_{P^{13}}(x)=\left(\frac{1}{\ell}\pm\varepsilon\right)^3 n^2=\frac{n^2}{\ell^3}\left(1\pm 4\ell\varepsilon\right),$$
$$\left(|P_{xy}|=\left(\frac{1}{\ell}\pm\varepsilon\right)\deg_{P^{12}}(x,y)\cdot\deg_{P^{13}}(x,y)=\left(\frac{1}{\ell}\pm\varepsilon\right)^5 n^2=\frac{n^2}{\ell^5}\left(1\pm 6\ell\varepsilon\right)\right),$$

which follows from the (ℓ^{-1},ε)-regularity of P^{23}. In particular, one can prove (cf. Fact 1.5 from [26]) that all but $4\varepsilon n$ vertices $x\in V_1$ (pairs $\{x,y\}\in\binom{V_1}{2}$) satisfy that

$$\text{(41)}\qquad P_x \text{ is } (\ell^{-1},2\ell\varepsilon)\text{-regular}\quad (P_{xy} \text{ is } (\ell^{-1},4\ell^2\varepsilon)\text{-regular}).$$

We now present Proposition 4.5.

PROPOSITION 4.5. *With given constants α, δ, and ℓ, sufficiently small $\varepsilon=\varepsilon(\alpha,\delta,\ell)>0$, and sufficiently large $n=n(\alpha,\delta,\ell,\varepsilon)$, whenever $\mathcal{H}$ and P are as in Setup* 2.1*, the following hold:*

1. *If $\mathcal{H}$ is $(\alpha,\delta)_1$-minimal w.r.t. P, then all but $3\delta^{1/3}n$ vertices $x\in V_1$ satisfy*
$$|L_x|=\frac{\alpha}{\ell^3}n^2\left(1\pm 3\delta^{1/3}\right)\overset{(40)}{=}\alpha|P_x|\left(1\pm 4\delta^{1/3}\right).$$
2. *If $\mathcal{H}$ is $(\alpha,\delta)_2$-minimal w.r.t. P, then all but $3\delta^{1/3}n^2$ pairs $\{x,y\}\in\binom{V_1}{2}$ satisfy*
$$|L_{xy}|=\frac{\alpha^2}{\ell^5}n^2\left(1\pm 3\delta^{1/3}\right)\overset{(40)}{=}\alpha^2|P_{xy}|\left(1\pm 4\delta^{1/3}\right).$$

The upcoming Proposition 4.6 establishes a "local characterization" of $(\alpha,\delta)_1$-minimality and $(\alpha,\delta)_2$-minimality, respectively. We use the following supplemental notation in our presentation. For a pair $1\le i<j\le 3$ and integers $s_1,s_2\in\{1,2\}$, let

$$\text{(42)}\qquad \mathcal{K}^{(2)}_{s_1,s_2}(P^{ij})=\left\{(S_1,S_2):\ S_1\in\binom{V_i}{s_1},\ S_2\in\binom{N_{P^{ij}}(S_1)}{s_2}\right\}.$$

Note that for each element $(S_1,S_2)\in\mathcal{K}^{(2)}_{s_1,s_2}(P^{ij})$, the set $S_1\cup S_2$ induces a copy of the complete bipartite graph $K^{(2)}_{s_1,s_2}$ in P^{ij}. In context, we shall use the standard fact that for $s_1,s_2\in\{1,2\}$,

$$\text{(43)}\qquad \left|\mathcal{K}^{(2)}_{s_1,s_2}\left(P^{ij}\right)\right|=\frac{1}{\ell^{s_1s_2}}\binom{n}{s_1}\binom{n}{s_2}\left(1\pm\varepsilon^{1/2}\right),$$

which follows from the (ℓ^{-1},ε)-regularity of P^{ij} (cf. Setup 2.1) whenever $\varepsilon > 0$ is sufficiently small and $n = n(\ell,\varepsilon)$ is sufficiently large.

PROPOSITION 4.6. *With given constants α, δ, and ℓ, sufficiently small $\varepsilon = \varepsilon(\alpha,\delta,\ell) > 0$, and sufficiently large integer $n = n(\alpha,\delta,\ell,\varepsilon)$, whenever $\mathcal{H}$ and P are as in Setup* 2.1 *with these constants, the following hold:*

1. (a) *if $\mathcal{H}$ is $(\alpha,\delta)_1$-minimal w.r.t. P, then all but $3\delta^{1/3}|\mathcal{K}^{(2)}_{1,2}(P^{12})|$ many elements $(\{x\},\{a,b\}) \in \mathcal{K}^{(2)}_{1,2}(P^{12})$ satisfy*

$$\deg_{L_x}(a,b) = \left(\frac{\alpha}{\ell}\right)^2 \frac{n}{\ell}\left(1 \pm 3\delta^{1/3}\right) \overset{(39)}{=} \left(\frac{\alpha}{\ell}\right)^2 \deg_{P^{13}}(x)\left(1 \pm 6\delta^{1/3}\right);$$

conversely,

(b) *if all but $\delta|\mathcal{K}^{(2)}_{1,2}(P^{12})|$ many elements $(\{x\},\{a,b\}) \in \mathcal{K}^{(2)}_{1,2}(P^{12})$ satisfy*

$$\deg_{L_x}(a,b) = \left(\frac{\alpha}{\ell}\right)^2 \frac{n}{\ell}\left(1 \pm \delta\right),$$

then $\mathcal{H}$ is $(\alpha,3\delta^{1/3})_1$-minimal w.r.t. P;

2. (a) *if $\mathcal{H}$ is $(\alpha,\delta)_2$-minimal w.r.t. P, then all but $3\delta^{1/3}|\mathcal{K}^{(2)}_{2,2}(P^{12})|$ many elements $(\{x,y\},\{a,b\}) \in \mathcal{K}^{(2)}_{2,2}(P^{12})$ satisfy*

$$\deg_{L_{xy}}(a,b) = \left(\frac{\alpha^2}{\ell}\right)^2 \frac{n}{\ell^2}\left(1 \pm 3\delta^{1/3}\right) \overset{(39)}{=} \left(\frac{\alpha^2}{\ell}\right)^2 \deg_{P^{13}}(x,y)\left(1 \pm 6\delta^{1/3}\right);$$

conversely,

(b) *if all but $\delta|\mathcal{K}^{(2)}_{2,2}(P^{12})|$ many elements $(\{x,y\},\{a,b\}) \in \mathcal{K}^{(2)}_{2,2}(P^{12})$ satisfy*

$$\deg_{L_{xy}}(a,b) = \left(\frac{\alpha^2}{\ell}\right)^2 \frac{n}{\ell^2}\left(1 \pm \delta\right),$$

then $\mathcal{H}$ is $(\alpha,3\delta^{1/3})_2$-minimal w.r.t. P.

4.1. Proofs. In all that follows, $\mathcal{H}$ and P are as in Setup 2.1. The constants α,δ,ℓ are fixed, and we shall take $\varepsilon = \varepsilon(\alpha,\delta,\ell) > 0$ sufficiently small and $n = n(\alpha,\delta,\ell,\varepsilon)$ sufficiently large whenever needed. We denote an application of the Cauchy–Schwarz inequality by "CS," and we denote by $o(1)$ a quantity vanishing as $n \to \infty$.

Proof of Proposition 4.1. Here, we prove only the promised lower bound for $|\mathcal{K}^{(3)}_{2,2,2}(\mathcal{H})|$. The proof of the corresponding lower bound for $|\mathcal{K}^{(3)}_{1,2,2}(\mathcal{H})|$ is similar (and can, in fact, be derived from the calculations below).

Summing over all $(\{x,y\},\{a,b\}) \in \mathcal{K}^{(2)}_{2,2}(P^{12})$ yields

$$\left|\mathcal{K}^{(3)}_{2,2,2}(\mathcal{H})\right| = \sum_{\mathcal{K}^{(2)}_{2,2}(P^{12})} \binom{\deg_{L_{xy}}(a,b)}{2} = \left(\frac{1}{2} - o(1)\right) \sum_{\mathcal{K}^{(2)}_{2,2}(P^{12})} \deg^2_{L_{xy}}(a,b) \tag{44}$$

$$\overset{\text{CS}}{\geq} \left(\frac{1}{2} - o(1)\right) \left|\mathcal{K}^{(2)}_{2,2}(P^{12})\right|^{-1} \left(\sum_{\mathcal{K}^{(2)}_{2,2}(P^{12})} \deg_{L_{xy}}(a,b)\right)^2$$

$$= \left(\frac{1}{2} - o(1)\right) \left|\mathcal{K}^{(2)}_{2,2}(P^{12})\right|^{-1} \left|\mathcal{K}^{(3)}_{2,2,1}(\mathcal{H})\right|^2 \overset{(43)}{\geq} \left(2 - \varepsilon^{1/3}\right) \frac{\ell^4}{n^4} \left|\mathcal{K}^{(3)}_{2,2,1}(\mathcal{H})\right|^2,$$

where we used $\varepsilon > 0$ sufficiently small and n sufficiently large. Summing over all $(\{a,b\},\{u\}) \in \mathcal{K}^{(2)}_{2,1}(P^{23})$ yields

$$\begin{aligned}\left|\mathcal{K}^{(3)}_{2,2,1}(\mathcal{H})\right| &= \sum_{\mathcal{K}^{(2)}_{2,1}(P^{23})} \binom{\deg_{L_u}(a,b)}{2} = \left(\frac{1}{2} - o(1)\right) \sum_{\mathcal{K}^{(2)}_{2,1}(P^{23})} \deg^2_{L_u}(a,b) \\ &\overset{\text{CS}}{\geq} \left(\frac{1}{2} - o(1)\right) \left|\mathcal{K}^{(2)}_{2,1}(P^{23})\right|^{-1} \left(\sum_{\mathcal{K}^{(2)}_{2,1}(P^{23})} \deg_{L_u}(a,b)\right)^2 \\ &\overset{(43)}{\geq} \left(1 - \varepsilon^{1/3}\right) \frac{\ell^2}{n^3} \left(\sum_{\mathcal{K}^{(2)}_{2,1}(P^{23})} \deg_{L_u}(a,b)\right)^2,\end{aligned} \tag{45}$$

where we used that $\varepsilon > 0$ is sufficiently small and n is sufficiently large. Note that

$$\begin{aligned}\sum_{\mathcal{K}^{(2)}_{2,1}(P^{23})} \deg_{L_u}(a,b) &= \sum_{\{x,u\}\in P^{13}} \binom{\deg_{L_x}(u)}{2} = \left(\frac{1}{2} - o(1)\right) \sum_{\{x,c\}\in P^{13}} \deg^2_{L_x}(u) \\ &\overset{\text{CS}}{\geq} \left(\frac{1}{2} - o(1)\right) |P^{13}|^{-1} \left(\sum_{\{x,u\}\in P^{13}} \deg_{L_x}(u)\right)^2 = \left(\frac{1}{2} - o(1)\right) |P^{13}|^{-1} |\mathcal{H}|^2.\end{aligned} \tag{46}$$

Now, the ε-regularity of P^{13} gives $|P^{13}| = (n^2/\ell)(1 \pm \varepsilon)$, and Setup 2.1 and Fact 1.2 ensure

$$|\mathcal{H}| = \alpha|\mathcal{K}_3(P)| = \alpha \frac{n^3}{\ell^3}\left(1 \pm \varepsilon^{1/3}\right) \tag{47}$$

so that, in view of (46) (and $\varepsilon > 0$ sufficiently small and n sufficiently large),

$$\sum_{\mathcal{K}^{(2)}_{2,1}(P^{23})} \deg_{L_u}(a,b) \geq \left(\frac{1}{2} - \varepsilon^{1/4}\right) \alpha^2 \frac{n^4}{\ell^5}. \tag{48}$$

Combining (44)–(48) and employing $\varepsilon > 0$ sufficiently small and n sufficiently large, we conclude

$$\left|\mathcal{K}^{(3)}_{2,2,2}(\mathcal{H})\right| \geq \frac{\alpha^8}{\ell^{12}} \binom{n}{2}^3 \left(1 - \varepsilon^{1/10}\right), \tag{49}$$

as promised. □

Proof of Proposition 4.4. Suppose $\mathcal{H}$ is $(\alpha,\delta)_2$-minimal w.r.t. P. We show $\mathcal{H}$ is also $(\alpha,\delta)_1$-minimal w.r.t. P. For this, note that it suffices to prove any of the inequalities in (35) since swapping the roles of V_1, V_2, V_3 preserves $(\alpha,\delta)_2$-minimality. We show $|\mathcal{K}^{(3)}_{2,2,1}(\mathcal{H})| \leq (\alpha^4/\ell^8) n \binom{n}{2}^2 (1+\delta)$. Indeed, the inequalities of (44) give

$$\begin{aligned}\left|\mathcal{K}^{(3)}_{2,2,1}(\mathcal{H})\right|^2 &\leq (2 + o(1)) \left|\mathcal{K}^{(2)}_{2,2}(P^{12})\right| \left|\mathcal{K}^{(3)}_{2,2,2}(\mathcal{H})\right| \\ &\overset{(43)}{\leq} (2 + o(1)) \frac{1}{\ell^4} \binom{n}{2}^2 \left(1 + \varepsilon^{1/2}\right) \left|\mathcal{K}^{(3)}_{2,2,2}(\mathcal{H})\right|.\end{aligned}$$

Since $\mathcal{H}$ is $(\alpha,\delta)_2$-minimal, i.e., $|\mathcal{K}^{(3)}_{2,2,2}(\mathcal{H})| \le (\alpha^8/\ell^{12})\binom{n}{2}^3(1+\delta)$, it now follows (with ε sufficiently small and n sufficiently large) that $|\mathcal{K}^{(3)}_{2,2,1}(\mathcal{H})| \le (\alpha^4/\ell^8)n\binom{n}{2}^2(1+\delta)$. □

In our proofs below, we shall use the following well-known "approximate" version of the Cauchy–Schwarz inequality (for a reference and a proof, see, for example, Proposition 1, [37, p. 5]).

FACT 4.7 (approximate Cauchy–Schwarz). *Let $\gamma > 0$ be given and suppose $a_1,\dots,a_s \ge 0$ satisfy*

1. $\sum_{i=1}^{s} a_i \ge as(1-\gamma)$,
2. $\sum_{i=1}^{s} a_i^2 < a^2 s(1+\gamma)$.

Then $\left|\left\{i \in [s] : |a-a_i| < 2\gamma^{1/3}a\right\}\right| > \left(1-2\gamma^{1/3}\right)s$.

Proof of Proposition 4.5. We sketch the proof of statement 1 (for single links) (the proof of statement 2 for colinks L_{xy} is similar). Assume that $\mathcal{H}$ is $(\alpha,\delta)_1$-minimal w.r.t. P. We show that all but $3\delta^{1/3}n$ vertices $x \in V_1$ satisfy $|L_x| = (\alpha/\ell^3)n^2(1 \pm 3\delta^{1/3})$. In our proof, we shall need the following inequalities, which are virtually identical to (45) and (46):

$$(50)\qquad \left|\mathcal{K}^{(3)}_{1,2,2}(\mathcal{H})\right| = \sum_{\mathcal{K}^{(2)}_{1,2}(P^{12})} \binom{\deg_{L_x}(a,b)}{2}$$
$$\overset{\text{CS}}{\ge} \left(\frac{1}{2}-o(1)\right)\left|\mathcal{K}^{(2)}_{1,2}(P^{12})\right|^{-1}\left(\sum_{\mathcal{K}^{(2)}_{1,2}(P^{12})} \deg_{L_x}(a,b)\right)^2$$

(where we sum over all $(\{x\},\{a,b\}) \in \mathcal{K}^{(2)}_{1,2}(P^{12})$) and

$$(51)\qquad \sum_{\mathcal{K}^{(2)}_{1,2}(P^{12})} \deg_{L_x}(a,b) = \sum_{\{x,u\}\in P^{13}} \binom{\deg_{L_x}(u)}{2} = \left(\frac{1}{2}-o(1)\right)\sum_{\{x,u\}\in P^{13}} \deg^2_{L_x}(u).$$

We want to use Fact 4.7 (with the a_i's being the terms $|L_x|$, $x \in V_1$). To that end, we need the following preparations. Observe that

$$\sum_{x\in V_1} |L_x| = |\mathcal{H}| \overset{(47)}{\ge} \alpha\frac{n^3}{\ell^3}\left(1-\varepsilon^{1/3}\right) \ge \left(\alpha\frac{n^2}{\ell^3}\right)n\left(1-3\delta\right).$$

To bound $\sum_{x\in V_1}|L_x|^2$, we separate the sum into two groups. Let V_1' denote those vertices $x \in V_1$ for which $\deg_{P^{13}}(x) = (\ell^{-1} \pm \varepsilon)n$. Then $|V_1 \setminus V_1'| < 2\varepsilon n$ and so

$$(52)\qquad \sum_{x\in V_1}|L_x|^2 = \sum_{x\in V_1'}|L_x|^2 + \sum_{x\in V_1\setminus V_1'}|L_x|^2 \le 2\varepsilon n^5 + \sum_{x\in V_1'}|L_x|^2.$$

We now bound $\sum_{x\in V_1'}|L_x|^2$. To that end, observe that

$$\sum_{\{x,u\}\in P^{13}} \deg^2_{L_x}(u) = \sum_{x\in V_1}\sum_{u\in N_{P^{13}}(x)} \deg^2_{L_x}(u) \ge \sum_{x\in V_1'}\sum_{u\in N_{P^{13}}(x)} \deg^2_{L_x}(u)$$
$$\overset{\text{CS}}{\ge} \sum_{x\in V_1'} \deg^{-1}_{P^{13}}(x)\left(\sum_{u\in N_{P^{13}}(x)} \deg_{L_x}(u)\right)^2$$
$$= \sum_{x\in V_1'} \deg^{-1}_{P^{13}}(x)|L_x|^2 \ge \frac{\ell}{n}(1+\ell\varepsilon)^{-1}\sum_{x\in V_1'}|L_x|^2,$$

where we used that each $x \in V_1'$ satisfies $\deg_{P^{13}}(x) = (\ell^{-1} \pm \varepsilon)n$. These inequalities, combined with (50) and (51) (and $\varepsilon > 0$ sufficiently small and n sufficiently large) then yield

$$\left(\sum_{x\in V_1'} |L_x|^2\right)^2 \le \left(8+\varepsilon^{1/2}\right)\frac{n^2}{\ell^2}\left|\mathcal{K}^{(2)}_{1,2}(P^{12})\right|\left|\mathcal{K}^{(3)}_{1,2,2}(\mathcal{H})\right| \overset{(43)}{\le} \left(4+\varepsilon^{1/3}\right)\frac{n^5}{\ell^4}\left|\mathcal{K}^{(3)}_{1,2,2}(\mathcal{H})\right|.$$

Since $\mathcal{H}$ is $(\alpha,\delta)_1$-minimal w.r.t. P, i.e., $|\mathcal{K}^{(3)}_{1,2,2}(\mathcal{H})| \le (\alpha^4/\ell^8)n\binom{n}{2}^2(1+\delta)$, we then see that (with $0 < \varepsilon \ll \delta$)

$$\sum_{x\in V_1'} |L_x|^2 \le \left(\alpha\frac{n^2}{\ell^3}\right)^2 n(1+2\delta). \tag{53}$$

Combining (52) and (53), we then see that (with $0 < \varepsilon \ll \delta$)

$$\sum_{x\in V_1} |L_x|^2 \le 2\varepsilon n^5 + \left(\alpha\frac{n^2}{\ell^3}\right)^2 n(1+2\delta) \le \left(\alpha\frac{n^2}{\ell^3}\right)^2 n(1+3\delta).$$

Now, to conclude the proof of (statement 1 of) Proposition 4.5, we set, as in Fact 4.7,

$$\gamma = 3\delta, \quad s = n, \quad a = \alpha\frac{n^2}{\ell^3}$$

and set the a_i's to be the terms $|L_x|$, $x \in V_1$. Fact 4.7 applies in that for all but $2(3\delta)^{1/3}n$ many vertices $x \in V_1$, we have

$$|L_x| = \alpha\frac{n^2}{\ell^3}\left(1 \pm 2(3\delta)^{1/3}\right).$$

Since $2 \cdot 3^{1/3} < 3$, we have shown that all but $3\delta^{1/3}n$ vertices $x \in V_1$ have $|L_x| = \alpha(n^2/\ell^3)(1 \pm 3\delta^{1/3})$, as promised by statement 1 of Proposition 4.5. □

Proof of Proposition 4.6. We sketch the proof of statement 2 (involving colinks L_{xy}) (the proof of statement 1 (involving links) is similar). To that end, we shall concentrate on the proof of statement 2(a), since the proof of statement 2(b) is quite standard. (Indeed, to briefly sketch it, suppose all but $\delta|\mathcal{K}^{(2)}_{2,2}(P^{12})|$ many copies $(\{x,y\},\{a,b\})$ of C_4 in P^{12} satisfy $\deg_{L_{xy}}(a,b) = (\alpha^2/\ell)^2(n/\ell^2)(1\pm\delta)$. Recall (cf. (44))

$$\left|\mathcal{K}^{(3)}_{2,2,2}(\mathcal{H})\right| = \sum_{\mathcal{K}^{(2)}_{2,2}(P^{12})} \binom{\deg_{L_{xy}}(a,b)}{2},$$

where our assumption is that we may control (from above) all but a δ proportion of the terms in the sum. Therefore, to establish an upper bound on $|\mathcal{K}^{(3)}_{2,2,2}(\mathcal{H})|$, we need to estimate the contribution of those at most δ proportion of terms over which we no not have such tight control. To this end, we use the ε-regularity of the graphs P^{12}, P^{23}, P^{13} to conclude that all but $\varepsilon^{1/3}n^4$ many terms $(\{x,y\},\{a,b\})$ satisfy $\deg_{L_{xy}}(a,b) \le 2n/\ell^4$, which essentially ends the argument.)

We now prove statement 2(a). To that end, assume $\mathcal{H}$ is $(\alpha,\delta)_2$-minimal w.r.t. P. We prove that for all but $3\delta^{1/3}|\mathcal{K}^{(2)}_{2,2}(P^{12})|$ many elements $(\{x,y\},\{a,b\}) \in \mathcal{K}^{(2)}_{2,2}(P^{12})$, we have

$$\deg_{L_{xy}}(a,b) = \left(\frac{\alpha^2}{\ell}\right)\frac{n}{\ell^2}\left(1 \pm 3\delta^{1/3}\right).$$

We want to use Fact 4.7 (with the a_i's being the terms $\deg_{L_{xy}}(a,b)$, $(\{x,y\},\{a,b\}) \in \mathcal{K}^{(2)}_{2,2}(P^{12})$). To that end, we need the following preparations. From (44), we infer that

$$\sum_{\mathcal{K}^{(2)}_{2,2}(P^{12})} \deg^2_{L_{xy}}(a,b) \le (2+o(1))\left|\mathcal{K}^{(3)}_{2,2,2}(\mathcal{H})\right|$$

so that, by the $(\alpha,\delta)_2$-minimality of $\mathcal{H}$ (and n sufficiently large), we have

$$\sum_{\mathcal{K}^{(2)}_{2,2}(P^{12})} \deg^2_{L_{xy}}(a,b) \le \alpha^8\frac{n^6}{4\ell^{12}}(1+2\delta) \overset{(43)}{\le} \left[\left(\frac{\alpha^2}{\ell}\right)^2\frac{n}{\ell^2}\right]^2\left|\mathcal{K}^{(2)}_{2,2}(P^{12})\right|(1+3\delta),$$

where in the last inequality we used $\varepsilon > 0$ sufficiently small. On the other hand,

$$\begin{aligned}\left(\sum_{\mathcal{K}^{(2)}_{2,2}(P^{12})} \deg_{L_{xy}}(a,b)\right)^2 &= \left|\mathcal{K}^{(3)}_{2,2,1}(\mathcal{H})\right|^2 \\ &\overset{(45)}{\ge} \left[\left(1-\varepsilon^{1/3}\right)\frac{\ell^2}{n^3}\left(\sum_{\mathcal{K}^{(2)}_{2,1}(P^{23})} \deg_{L_u}(a,b)\right)^2\right]^2 \\ &\overset{(48)}{\ge} \left[\left(1-\varepsilon^{1/3}\right)\frac{\ell^2}{n^3}\left(\left(\frac{1}{2}-\varepsilon^{1/4}\right)\alpha^2\frac{n^4}{\ell^5}\right)^2\right]^2 \ge \alpha^8\frac{n^{10}}{16\ell^{16}}\left(1-\varepsilon^{1/5}\right)\end{aligned}$$

(using $\varepsilon > 0$ sufficiently small), so that

$$\begin{aligned}\sum_{\mathcal{K}^{(2)}_{2,2}(P^{12})} \deg_{L_{xy}}(a,b) \ &\ge \ \left[\left(\frac{\alpha^2}{\ell}\right)^2\frac{n}{\ell^2}\right]\frac{n^4}{4\ell^4}\left(1-\varepsilon^{1/5}\right) \\ &\overset{(43)}{\ge} \left[\left(\frac{\alpha^2}{\ell}\right)^2\frac{n}{\ell^2}\right]\left|\mathcal{K}^{(2)}_{2,2}(P^{12})\right|\left(1-\varepsilon^{1/6}\right) \\ &\ge \left[\left(\frac{\alpha^2}{\ell}\right)^2\frac{n}{\ell^2}\right]\left|\mathcal{K}^{(2)}_{2,2}(P^{12})\right|(1-3\delta).\end{aligned}$$

Now to conclude the proof of (statement 2(a) of) Proposition 4.6, we set, as in Fact 4.7,

$$\gamma = 3\delta, \quad s = |\mathcal{K}^{(2)}_{2,2}(P^{12})|, \quad a = \left(\frac{\alpha^2}{\ell}\right)^2\frac{n}{\ell^2}$$

and set the a_i's to be the terms $\deg_{L_{xy}}(a,b)$, $(\{x,y\},\{a,b\}) \in \mathcal{K}^{(2)}_{2,2}(P^{12})$. Fact 4.7 applies to say that for all but $2(3\delta)^{1/3}|\mathcal{K}^{(2)}_{2,2}(P^{12})|$ many terms, we have

$$\deg_{L_{xy}}(a,b) = \left(\frac{\alpha^2}{\ell}\right)^2 \frac{n}{\ell^2}\left(1 \pm 2(3\delta)^{1/3}\right).$$

Since $2 \cdot 3^{1/3} < 3$, this proves statement 2(a) of Proposition 4.6. □

5. Proof of the counting lemma. In this section, we prove the counting lemma, Theorem 2.7. Our proof proceeds formally by induction on $k \geq 3$.

Base case. The base case $k = 3$ holds trivially. Indeed, $\mathcal{H}^{123} = \mathcal{H}$ has density $\alpha = |\mathcal{H}|/|\mathcal{K}^{(2)}_3(P)|$ w.r.t. $P = P^{12} \cup P^{23} \cup P^{13}$. With constant $\varepsilon \leq \delta$ sufficiently small w.r.t. ℓ^{-1}, Fact 1.2 renders

$$\left|\mathcal{K}^{(2)}_3(P)\right| = \frac{n^3}{\ell^3}\left(1 \pm \varepsilon^{1/3}\right) \quad \Longrightarrow \quad \left|\mathcal{K}^{(3)}_3(\mathcal{H})\right| = |\mathcal{H}| = \alpha\frac{n^3}{\ell^3}\left(1 \pm \varepsilon^{1/3}\right) = \alpha\frac{n^3}{\ell^3}\left(1 \pm \delta^{1/360}\right),$$

confirming the counting lemma for $k = 3$.

Induction step. We assume Theorem 2.7 holds up through $k - 1 \geq 3$ and consider Theorem 2.7 for $k \geq 4$. With appropriately defined constants (which we comment on momentarily), we consider $\mathcal{H} = \bigcup_{1 \leq h < i < j \leq k} \mathcal{H}^{hij}$ and $P = \bigcup_{1 \leq i < j \leq k} P^{ij}$ as in Setup 2.6 on k-partition $V = V_1 \cup \cdots \cup V_k$, $|V_1| = \cdots = |V_k| = n$, where each $\mathcal{H}^{hij}$, $1 \leq h < i < j \leq k$, is (α,δ)-minimal w.r.t. $P^{hi} \cup P^{ij} \cup P^{hj}$ and each P^{ij}, $1 \leq i < j \leq k$, is (ℓ^{-1},ε)-regular. We prove

$$\left|\mathcal{K}^{(3)}_k(\mathcal{H})\right| = \frac{\alpha^{\binom{k}{3}}}{\ell^{\binom{k}{2}}} n^k \left(1 \pm \delta^{\frac{1}{120k}}\right). \tag{54}$$

We now discuss the constants $k, \alpha, \delta_0, \delta, \ell, \varepsilon, n$ required to enable (54).

With k fixed (by induction), the counting lemma is quantified as

$$\forall \alpha,\ \exists \delta_0 : \forall \delta < \delta_0,\ \forall \ell,\ \exists \varepsilon : \quad \text{with } n \text{ sufficiently large.}$$

This quantification is consistent with the following hierarchy:

$$\frac{1}{k}, \alpha \gg \delta_0 > \delta \geq \min\left\{\delta, \frac{1}{\ell}\right\} \gg \varepsilon \gg \frac{1}{n}. \tag{55}$$

Rather than presenting a tedious determination of the constants $\alpha, \delta_0, \delta, \ell, \varepsilon, n$, we shall, in all calculations below, appeal to the hierarchy in (55).

To establish (54), we find it convenient to reformulate the counting lemma in terms of a slightly different language given by auxiliary bipartite graphs $\Lambda \subseteq \Pi$, which we now define.

CONSTRUCTION 5.1 (defining $\Lambda \subseteq \Pi$). *With k-partition $V_1 \cup \cdots \cup V_k$ of $\mathcal{H} = \bigcup_{1 \leq h < i < j \leq k} \mathcal{H}^{hij}$ and $P = \bigcup_{1 \leq i < j \leq k} P^{ij}$ given as in Setup 2.6, define auxiliary bipartite graphs $\Lambda \subseteq \Pi$ with bipartition $X \cup Y$ as follows:*

-
$$X = V_1 \quad \textit{and} \quad Y = \mathcal{K}^{(2)}_{k-1}(P[V_2,\dots,V_k]) = \mathcal{K}^{(2)}_{k-1}\Big(\bigcup_{1 < i < j \leq k} P^{ij}\Big),$$

where elements of Y, each denoted by K^-, correspond to the vertex sets of graph $(k-1)$-cliques $K^{(2)}_{k-1}$ in the $(k-1)$-partite graph $P[V_2,\dots,V_k] = \bigcup_{1<i<j\le k} P^{ij}$. Note that

$$|X| = n \quad \textit{and} \quad |Y| \overset{\text{Fact 1.2}}{=} \frac{n^{k-1}}{\ell^{\binom{k-1}{2}}}\left(1 \pm \varepsilon^{\frac{1}{k-1}}\right). \tag{56}$$

- *For $x \in X$ and $K^- \in Y$,*

$$\begin{aligned}\left\{x, K^-\right\} \in \Pi \iff \{x\} \cup K^- \in \mathcal{K}^{(2)}_k(P) &\iff K^- \in \mathcal{K}^{(2)}_{k-1}(P_x)\\ &\iff K^- \subset N_P(x),\end{aligned}$$

where $P_x = P\left[N_P(x)\right] = P\left[N_{P^{12}}(x), \dots, N_{P^{1k}}(x)\right]$ is the subgraph of P induced on the neighborhood $N_P(x) = N_{P^{12}}(x) \cup \cdots \cup N_{P^{1k}}(x)$ of x.

- *For $x \in X$ and $K^- \in Y$,*

$$\left\{x, K^-\right\} \in \Lambda \iff K^- \in \mathcal{K}^{(2)}_{k-1}(L_x) \iff \binom{K^-}{2} \subset L_x,$$

where $L_x = \{\{y,z\} \in P_x : \{x,y,z\} \in \mathcal{H}\} \subseteq P_x$ is the link graph of x (cf. (36)*).*

Note that since each $x \in V_1$ has $L_x \subseteq P_x$, it follows that $\Lambda \subseteq \Pi$. We also note the following useful but standard fact concerning graph Π.

Fact 5.2. *With ε given in* (55)*, the graph Π is $(\ell^{1-k}, \varepsilon^{1/3})$-regular.*

We omit the standard proof of Fact 5.2.

We now make a few straightforward observations establishing connections between graphs Λ and Π and the counting lemma, Theorem 2.7. For the purpose of stating these observations, set

$$Y_{\mathcal{H}} \overset{\text{def}}{=} \mathcal{K}^{(3)}_{k-1}(\mathcal{H}[V_2,\dots,V_k]) = \mathcal{K}^{(3)}_{k-1}\Big(\bigcup_{1<h<i<j\le k} \mathcal{H}^{hij}\Big)$$

and note that elements of $Y_{\mathcal{H}}$ correspond to vertex sets of hypergraph $(k-1)$-cliques $K^{(3)}_{k-1}$ in the $(k-1)$-partite 3-graph $\mathcal{H}[V_2,\dots,V_k] = \bigcup_{1<h<i<j\le k} \mathcal{H}^{hij}$. We observe the following.

Observations.

1. *Since $\mathcal{H} \subseteq \mathcal{K}^{(2)}_3(P)$ (cf. Setup* 2.6*), $Y_{\mathcal{H}} \subseteq Y$ is a subset of the vertices Y.*
2. *By our induction hypothesis on the counting lemma (for $k-1$),*

$$|Y_{\mathcal{H}}| = \left|\mathcal{K}^{(3)}_{k-1}(\mathcal{H}[V_2,\dots,V_k])\right| = \left|\mathcal{K}^{(3)}_{k-1}\Big(\bigcup_{1<h<i<j\le k} \mathcal{H}^{hij}\Big)\right|$$

$$= \frac{\alpha^{\binom{k-1}{3}}}{\ell^{\binom{k-1}{2}}} n^{k-1}\left(1 \pm \delta^{\frac{1}{120(k-1)}}\right) \overset{(55),(56)}{=} \alpha^{\binom{k-1}{3}}|Y|\left(1 \pm 2\delta^{\frac{1}{120(k-1)}}\right). \tag{57}$$

3.

$$\left|\mathcal{K}^{(3)}_k(\mathcal{H})\right| = \sum_{K^- \in Y_{\mathcal{H}}} \deg_\Lambda\left(K^-\right). \tag{58}$$

We see from (57) and (58) that, to prove the counting lemma, it suffices to analyze the terms in the sum (58). Proposition 5.3 does precisely this.

PROPOSITION 5.3 (key to counting lemma). *All but* $3\delta^{\frac{1}{111(k-1)}}|Y|$ *vertices* $K^- \in Y$ *satisfy*

$$\deg_\Lambda\left(K^-\right) = \frac{\alpha^{\binom{k-1}{2}}}{\ell^{k-1}}|X|\left(1 \pm 3\delta^{\frac{1}{111(k-1)}}\right) = \frac{\alpha^{\binom{k-1}{2}}}{\ell^{k-1}}n\left(1 \pm 3\delta^{\frac{1}{111(k-1)}}\right). \tag{59}$$

We defer the proof of Proposition 5.3 to section 5.2. We proceed now with the confirmation that the counting lemma follows from Proposition 5.3.

5.1. Proposition 5.3 $\Longrightarrow$ counting lemma. Combining Proposition 5.3 with (57) and (58), the proof of the counting lemma is virtually immediate. Indeed, Proposition 5.3 and (57) imply "almost all" (cf. $\delta \ll \alpha$ in (55)) of the (57) many terms in (58) have degree in (59). As such,

$$\left|\mathcal{K}_k^{(3)}(\mathcal{H})\right| = \sum_{K^-\in Y_\mathcal{H}} \deg_\Lambda\left(K^-\right) \sim \frac{\alpha^{\binom{k-1}{2}}}{\ell^{k-1}}n \times \frac{\alpha^{\binom{k-1}{3}}}{\ell^{\binom{k-1}{2}}}n^{k-1} = \frac{\alpha^{\binom{k}{3}}}{\ell^{\binom{k}{2}}}n^k, \tag{60}$$

where $\sim$ denotes an "essential" equality we make precise by accounting for the degrees of vertices $K^- \in Y_\mathcal{H}$ not satisfying (59).

To make (60) precise, we will use (59) for all $K^- \in Y_\mathcal{H}$ satisfying it, and for the remaining ones, we employ the natural upper bound $\deg_\Lambda(K^-) \leq \deg_\Pi(K^-)$ (recall $\Lambda \subseteq \Pi$). As such, Fact 5.2 implies

$$\begin{gathered}\textit{all but } 2\varepsilon^{1/3}|Y| \textit{ vertices } K^- \in Y \textit{ satisfy}\\ \deg_\Pi\left(K^-\right) = \left(\frac{1}{\ell^{k-1}} \pm \varepsilon^{1/3}\right)|X| \overset{(55)}{\leq} 2\frac{n}{\ell^{k-1}}.\end{gathered} \tag{61}$$

Now, set

$$Y_\Lambda = \left\{K^- \in Y : \text{ } K^- \text{ satisfies (59)}\right\} \quad \text{and} \quad Y_\Pi = \left\{K^- \in Y : \text{ } K^- \text{ satisfies (61)}\right\} \tag{62}$$

so that

$$|Y_\mathcal{H} \setminus Y_\Lambda| \leq |Y \setminus Y_\Lambda| \overset{\text{Prop. 5.3}}{\leq} 3\delta^{\frac{1}{111(k-1)}}|Y| \overset{(56)}{\leq} 6\delta^{\frac{1}{111(k-1)}}\frac{n^{k-1}}{\ell^{\binom{k-1}{2}}}, \tag{63}$$

$$|Y_\mathcal{H} \setminus Y_\Pi| \leq |Y \setminus Y_\Pi| \overset{(61)}{\leq} 2\varepsilon^{1/3}|Y| \overset{(56)}{\leq} 4\varepsilon^{1/3}\frac{n^{k-1}}{\ell^{\binom{k-1}{2}}}. \tag{64}$$

We now make (60) precise. Observe that

$$\begin{aligned}\left|\mathcal{K}_k^{(3)}\left(\mathcal{H}\right)\right| = \sum_{K^-\in Y_\Lambda\cap Y_\mathcal{H}} \deg_\Lambda\left(K^-\right) &+ \sum_{K^-\in (Y_\Pi\setminus Y_\Lambda)\cap Y_\mathcal{H}} \deg_\Lambda\left(K^-\right)\\ &+ \sum_{K^-\in Y_\mathcal{H}\setminus(Y_\Lambda\cup Y_\Pi)} \deg_\Lambda\left(K^-\right).\end{aligned} \tag{65}$$

To obtain the formula for $|\mathcal{K}_k^{(3)}(\mathcal{H})|$ promised in (54), we need to bound (65) from above and below. We do so now.

To see the lower bound, we use that every $K^- \in Y_\Lambda \cap Y_{\mathcal{H}}$ satisfies $\deg_\Lambda(K^-) \geq (\alpha^{\binom{k-1}{2}}/\ell^{k-1})n(1-3\delta^{111(k-1)})$ (cf. (59)) to obtain (from (65))

$$\begin{aligned}\left|\mathcal{K}_k^{(3)}(\mathcal{H})\right| &\geq \frac{\alpha^{\binom{k-1}{2}}}{\ell^{k-1}} n \left(1-3\delta^{\frac{1}{111(k-1)}}\right)|Y_\Lambda \cap Y_{\mathcal{H}}| \\ &= \frac{\alpha^{\binom{k-1}{2}}}{\ell^{k-1}} n \left(1-3\delta^{\frac{1}{111(k-1)}}\right)(|Y_{\mathcal{H}}| - |Y_{\mathcal{H}} \setminus Y_\Lambda|) \\ &\overset{(57),\,(63)}{\geq} \frac{\alpha^{\binom{k}{3}}}{\ell^{\binom{k}{2}}} n^k \left(1-3\delta^{\frac{1}{111(k-1)}}\right)\left[1-\delta^{\frac{1}{120(k-1)}} - 6\frac{\delta^{\frac{1}{111(k-1)}}}{\alpha^{\binom{k-1}{3}}}\right] \overset{(55)}{\geq} \frac{\alpha^{\binom{k}{3}}}{\ell^{\binom{k}{2}}} n^k \left(1-\delta^{\frac{1}{120k}}\right).\end{aligned}$$

To see the upper bound, we argue similarly, this time showing that the terms of (65) outside of $\sum_{K^- \in Y_\Lambda \cap Y_{\mathcal{H}}} \deg_\Lambda(K^-)$ contribute to only little error. Indeed,

$$\begin{aligned}\left|\mathcal{K}_k^{(3)}(\mathcal{H})\right| &\leq \sum_{K^- \in Y_\Lambda \cap Y_{\mathcal{H}}} \deg_\Lambda(K^-) + \sum_{K^- \in (Y_\Pi \setminus Y_\Lambda) \cap Y_{\mathcal{H}}} \deg_\Pi(K^-) + n\left|Y_{\mathcal{H}} \setminus (Y_\Lambda \cup Y_\Pi)\right| \\ &\overset{(59),\,(61)}{\leq} \frac{\alpha^{\binom{k-1}{2}}}{\ell^{k-1}} n \left(1+3\delta^{\frac{1}{111(k-1)}}\right)\left|Y_\Lambda \cap Y_{\mathcal{H}}\right| + 2\frac{n}{\ell^{k-1}}\left|\left(Y_\Pi \setminus Y_\Lambda\right) \cap Y_{\mathcal{H}}\right| \\ &\qquad + n\left|Y_{\mathcal{H}} \setminus \left(Y_\Lambda \cup Y_\Pi\right)\right| \\ &\leq \frac{\alpha^{\binom{k-1}{2}}}{\ell^{k-1}} n \left(1+3\delta^{\frac{1}{111(k-1)}}\right)|Y_{\mathcal{H}}| + 2\frac{n}{\ell^{k-1}}|Y_{\mathcal{H}} \setminus Y_\Lambda| + n|Y_{\mathcal{H}} \setminus Y_\Pi| \\ &\overset{(57),\,(63),\,(64)}{\leq} \frac{\alpha^{\binom{k}{3}}}{\ell^{\binom{k}{2}}} n^k \left(1+3\delta^{\frac{1}{111(k-1)}}\right)\left(1+\delta^{\frac{1}{120(k-1)}}\right) + 6\delta^{\frac{1}{111(k-1)}}\frac{n^k}{\ell^{\binom{k}{2}}} + 8\varepsilon^{1/3}\frac{n^k}{\ell^{\binom{k-1}{2}}} \\ &= \frac{\alpha^{\binom{k}{3}}}{\ell^{\binom{k}{2}}} n^k \left[\left(1+3\delta^{\frac{1}{111(k-1)}}\right)\left(1+\delta^{\frac{1}{120(k-1)}}\right) + 6\frac{\delta^{\frac{1}{111(k-1)}}}{\alpha^{\binom{k}{3}}} + 8\frac{\varepsilon^{1/3}\ell^{k-1}}{\alpha^{\binom{k}{3}}}\right] \\ &\overset{(55)}{\leq} \frac{\alpha^{\binom{k}{3}}}{\ell^{\binom{k}{2}}} n^k \left(1+\delta^{\frac{1}{120k}}\right).\end{aligned}$$

This completes the proof of the induction step for the counting lemma.

5.2. Proof of Proposition 5.3.

It remains to prove Proposition 5.3. We do so by proving the following stronger version of Proposition 5.3.

Proposition 5.3*.

- *All but* $\delta^{1/36}|X| = \delta^{1/36}n$ *vertices* $x \in X = V_1$ *satisfy*

(66)

$$\deg_\Lambda(x) = \left(\frac{\alpha}{\ell}\right)^{\binom{k-1}{2}}\left(\frac{n}{\ell}\right)^{k-1}\left(1 \pm \delta^{\frac{1}{36(k-1)}}\right) \overset{(56)}{=} \frac{\alpha^{\binom{k-1}{2}}}{\ell^{k-1}}|Y|\left(1 \pm 2\delta^{\frac{1}{36(k-1)}}\right).$$

In particular,

$$\sum_{K^- \in Y} \deg_\Lambda(K^-) \geq \left(\frac{\alpha^{\binom{k-1}{2}}}{\ell^{k-1}}|X|\right)|Y|\left(1-\delta^{\frac{1}{37(k-1)}}\right). \qquad (67)$$

- *All but* $\delta^{1/36}|X|^2 = \delta^{1/36}n^2$ *pairs* $x, y \in X = V_1$ *satisfy*

$$\deg_\Lambda(x,y) = \left(\frac{\alpha^2}{\ell}\right)^{\binom{k-1}{2}} \left(\frac{n}{\ell^2}\right)^{k-1}\left(1 \pm \delta^{\frac{1}{36(k-1)}}\right) \overset{(56)}{=} \left(\frac{\alpha^{\binom{k-1}{2}}}{\ell^{k-1}}\right)^2 |Y|\left(1 \pm 2\delta^{\frac{1}{36(k-1)}}\right). \tag{68}$$

In particular,

$$\sum_{K^-\in Y} \deg_\Lambda(K^-)^2 < \left(\frac{\alpha^{\binom{k-1}{2}}}{\ell^{k-1}}|X|\right)^2 |Y|\left(1 + 3\delta^{\frac{1}{37(k-1)}}\right). \tag{69}$$

It is easy to prove Proposition 5.3 from (67) and (69) of Proposition 5.3*. We do so using the approximate Cauchy–Schwarz inequality, Fact 4.7. Indeed, in Fact 4.7, set $a = (\alpha^{\binom{k-1}{2}}/\ell^{k-1})|X|$, $r = |Y|$ and $\gamma = 3\delta^{\frac{1}{37(k-1)}}$. The terms a_i, $1 \le i \le r$, correspond to terms $\deg_\Lambda(K^-)$, $K^- \in Y$. Fact 4.7 then gives that all but $2(3\delta^{\frac{1}{37(k-1)}})^{1/3}|Y| < 3\delta^{\frac{1}{111(k-1)}}|Y|$ terms $\deg_\Lambda(K^-)$, $K^- \in Y$, satisfy

$$\deg_\Lambda(K^-) = \frac{\alpha^{\binom{k-1}{2}}}{\ell^{k-1}}|X|\left(1 \pm 2\left(3\delta^{\frac{1}{37(k-1)}}\right)^{1/3}\right) = \frac{\alpha^{\binom{k-1}{2}}}{\ell^{k-1}}|X|\left(1 \pm 3\delta^{\frac{1}{111(k-1)}}\right),$$

where we used that $2 \cdot 3^{1/3} < 3$. This proves Proposition 5.3.

To prove Proposition 5.3*, we must show four things: the assertions of (66) and (68) and the implications (66) $\Longrightarrow$ (67) and (68) $\Longrightarrow$ (69). The assertions of (66) and (68) follow from Lemma 5.4 given at the beginning of this section, but the work requires some effort to prepare. The implications are, on the other hand, easy, and so we proceed with those first. We shall then immediately return to the task of proving the assertions of (66) and (68).

5.2.1. Proof that (66) $\Longrightarrow$ (67). Observe that

$$\sum_{K^-\in Y} \deg_\Lambda(K^-) = \sum_{x\in X} \deg_\Lambda(x).$$

Now, denote by X_{good} the set of vertices $x \in X$ for which (66) holds. Then,

$$\sum_{K^-\in Y} \deg_\Lambda(K^-) = \sum_{x\in X} \deg_\Lambda(x) \ge \sum_{x\in X_{\text{good}}} \deg_\Lambda(x) \overset{(66)}{\ge} \frac{\alpha^{\binom{k-1}{2}}}{\ell^{k-1}}|Y|\left(1 - 2\delta^{\frac{1}{36(k-1)}}\right)|X_{\text{good}}|$$
$$\overset{(66)}{\ge} \frac{\alpha^{\binom{k-1}{2}}}{\ell^{k-1}}|Y|\left(1 - 2\delta^{\frac{1}{36(k-1)}}\right)\left(1 - \delta^{1/36}\right)|X| \overset{(55)}{\ge} \left(\frac{\alpha^{\binom{k-1}{2}}}{\ell^{k-1}}|X|\right)|Y|\left(1 - \delta^{\frac{1}{37(k-1)}}\right),$$

as promised in (67).

5.2.2. Proof that (68) $\Longrightarrow$ (69). Recall that we are supposed to bound $\sum_{K^-\in Y} \deg_\Lambda(K^-)^2$. To this end, we first observe that

$$\sum_{\{x,y\}\in\binom{X}{2}} \deg_\Lambda(x,y) = \sum_{K^-\in Y} \binom{\deg_\Lambda(K^-)}{2} \overset{(55)}{=} \left(\frac{1}{2} - o(1)\right)\sum_{K^-\in Y} \deg_\Lambda(K^-)^2, \tag{70}$$

where $o(1) \to 0$ as $n \to \infty$. It therefore suffices to work with the sum $\sum_{\{x,y\}\in\binom{X}{2}} \deg_\Lambda(x,y)$.

Denote by $\binom{X}{2}_{\text{good}}$ the set of pairs $\{x,y\} \in \binom{X}{2}$ for which (68) holds. For pairs $\{x,y\} \in \binom{X}{2} \setminus \binom{X}{2}_{\text{good}}$, we observe $\deg_\Lambda(x,y) \le \deg_\Pi(x,y)$ (since $\Lambda \subseteq \Pi$), where we recall from Fact 5.2 that Π is $(\ell^{1-k}, \varepsilon^{1/3})$-regular. As such (and similarly to (61)), we have that all but $4\varepsilon^{1/3}n^2$ pairs $\{x,y\} \in \binom{X}{2}$ satisfy

$$\deg_\Pi(x,y) = \left(\frac{1}{\ell^{k-1}} \pm \varepsilon^{1/3}\right)^2 |Y| \overset{(55)}{<} \frac{2}{\ell^{2(k-1)}}|Y|. \tag{71}$$

Denote by $\binom{X}{2}_{\text{fair}}$ the set of pairs $\{x,y\} \in \binom{X}{2}$ for which (71) holds.

Returning to (70), we then see that the sum $\sum_{\{x,y\}\in\binom{X}{2}} \deg_\Lambda(x,y)$ equals

$$\begin{aligned}
&\sum_{\{x,y\}\in\binom{X}{2}_{\text{good}}} \deg_\Lambda(x,y) + \sum_{\{x,y\}\in\binom{X}{2}_{\text{fair}}\setminus\binom{X}{2}_{\text{good}}} \deg_\Lambda(x,y) \\
&\qquad + \sum_{\{x,y\}\in\binom{X}{2}\setminus\left(\binom{X}{2}_{\text{fair}}\cup\binom{X}{2}_{\text{good}}\right)} \deg_\Lambda(x,y) \\
&\le \sum_{\{x,y\}\in\binom{X}{2}_{\text{good}}} \deg_\Lambda(x,y) + \sum_{\{x,y\}\in\binom{X}{2}_{\text{fair}}\setminus\binom{X}{2}_{\text{good}}} \deg_\Pi(x,y) + |Y|\left|\binom{X}{2}\setminus\binom{X}{2}_{\text{fair}}\right| \\
&\overset{(68),\,(71)}{\le} \left(\frac{\alpha^{\binom{k-1}{2}}}{\ell^{k-1}}\right)^2 |Y|\left(1+2\delta^{\frac{1}{36(k-1)}}\right)\left|\binom{X}{2}_{\text{good}}\right| + \frac{2}{\ell^{2(k-1)}}|Y|\left|\binom{X}{2}\setminus\binom{X}{2}_{\text{good}}\right| \\
&\qquad +4\varepsilon^{1/3}n^2|Y| \\
&\overset{(68)}{\le} \left(\frac{\alpha^{\binom{k-1}{2}}}{\ell^{k-1}}\right)^2 |Y|\left(1+2\delta^{\frac{1}{36(k-1)}}\right)\binom{n}{2} + 2\delta^{1/36}\frac{n^2}{\ell^{2(k-1)}}|Y| + 4\varepsilon^{1/3}n^2|Y| \\
&\le \left(\frac{\alpha^{\binom{k-1}{2}}}{\ell^{k-1}}\right)^2 |Y|n^2\left(\frac{1}{2} + \delta^{\frac{1}{36(k-1)}} + 2\frac{\delta^{1/36}}{\alpha^{2\binom{k-1}{2}}} + 4\varepsilon^{1/3}\frac{\ell^{2(k-1)}}{\alpha^{2\binom{k-1}{2}}}\right) \\
&\overset{(55)}{\le} \left(\frac{\alpha^{\binom{k-1}{2}}}{\ell^{k-1}}|X|\right)^2 |Y|\left(\frac{1}{2} + \delta^{\frac{1}{37(k-1)}}\right),
\end{aligned}$$

so that, from (70), we have

$$\begin{aligned}
\sum_{K^-\in Y} \deg_\Lambda(K^-)^2 &\le \left(\frac{\alpha^{\binom{k-1}{2}}}{\ell^{k-1}}|X|\right)^2 |Y|\left(1+2\delta^{\frac{1}{37(k-1)}}\right)(1+o(1)) \\
&\overset{(55)}{<} \left(\frac{\alpha^{\binom{k-1}{2}}}{\ell^{k-1}}|X|\right)^2 |Y|\left(1+3\delta^{\frac{1}{37(k-1)}}\right),
\end{aligned}$$

as promised in (69).

5.3. Proofs of (66) and (68): Setting up the proofs. We claim that (66) and (68) are direct applications of the following auxiliary graph counting lemma.

Lemma 5.4 (a "two graphs" counting lemma). *For all integers t and constants $\lambda_0 \geq 0$ there exists $\delta_{0,\lambda_0} > 0$ so that for all $0 < \delta_{\lambda_0} \leq \delta_{0,\lambda_0}$ and $p > 0$ there exists $\varepsilon_p > 0$ so that the following holds.*

Let

$$\mathfrak{L} = \bigcup_{1 \leq i < j \leq t} \mathfrak{L}^{ij} \subseteq \mathfrak{P} = \bigcup_{1 \leq i < j \leq t} \mathfrak{P}^{ij}$$

be t-partite graphs with common t-partition $U_1 \cup \cdots \cup U_t$, $|U_i| = m_i > m_0(t, \lambda_0, \delta_\lambda, p, \varepsilon_p)$, satisfying that for all $1 \leq i \neq j \leq t$,

1. $|\mathfrak{L}^{ij}|/|\mathfrak{P}^{ij}| \overset{\text{def}}{=} \lambda_{ij} \geq \lambda_0$,
2. *all but $\delta_{\lambda_0} m_i^2$ pairs $u_1, u_2 \in U_i$ satisfy*

$$\deg_{\mathfrak{L}^{ij}}(u_1, u_2) = (\lambda_{ij} p)^2 m_j (1 \pm \delta_{\lambda_0}), \tag{72}$$

3. *the graph $\mathfrak{P}^{ij}$ is (p, ε_p)-regular.*

Then,

$$\left|\mathcal{K}_t^{(2)}(\mathfrak{L})\right| = \left(\prod_{1 \leq i < j \leq t} \lambda_{ij}\right) p^{\binom{t}{2}} \left(\prod_{i=1}^{t} m_i\right) \left(1 \pm \delta_{\lambda_0}^{\frac{1}{5t}}\right).$$

We prove Lemma 5.4 in section 6.

We now connect (66) and (68) with Lemma 5.4. Indeed, for arbitrary vertices $x, y \in X = V_1$, observe from Construction 5.1 that

$$\deg_\Lambda(x) = \left|\mathcal{K}_{k-1}^{(2)}(L_x)\right| \quad \text{and} \quad \deg_\Lambda(x, y) = \left|\mathcal{K}_{k-1}^{(2)}(L_{xy})\right|, \tag{73}$$

where

$$L_x = \Big\{\{u, v\} \in P : \{x, u, v\} \in \mathcal{H}\Big\} \quad \text{and} \quad L_{xy} = L_x \cap L_y \tag{74}$$

are the links L_x and colinks L_{xy} of x and $\{x, y\}$, respectively (cf. (36) and (37)). In view of (73), we may prove (66) and (68) by respectively applying Lemma 5.4 to graphs L_x (i.e., $\mathfrak{L} = L_x$) and L_{xy} (i.e., $\mathfrak{L} = L_{xy}$), $x, y \in X = V_1$, whenever it is appropriate to do so.

To further develop our plans for proving (66) and (68) by applying Lemma 5.4, we continue with some notation (some of which is similar to that used in section 4).

Notation 5.5. Let T be a fixed one- or two-element subset of X. We set

$$U_{i,T} = N_{P^{1i}}(T) = \bigcap_{x \in T} N_{P^{1i}}(x) \quad \text{and} \quad m_{i,T} = |U_{i,T}|, \ \ 1 < i \leq k. \tag{75}$$

Set

$$\mathfrak{P}_T^{ij} = P_T^{ij} = P^{ij}[N_{P^{1i}}(T), N_{P^{1j}}(T)], \ 1 < i < j \leq k,$$

and

$$\mathfrak{P}_T = \bigcup_{1 < i < j \leq k} \mathfrak{P}_T^{ij} = \bigcup_{1 < i < j \leq k} P_T^{ij} = P_T.$$

Note that $\mathfrak{P}_T$ is $(k-1)$-partite with $(k-1)$-partition $U_{2,T} \cup \cdots \cup U_{k,T}$. Set

$$\mathfrak{L}_T = L_T = \bigcap_{x \in T} L_x,$$

where L_x, $x \in T$, is given in (74). For $1 < i < j \leq k$, set $\mathfrak{L}_T^{ij} = \mathfrak{L}_T \cap P^{ij}$ so that

$$\mathfrak{L}_T = \bigcup_{1<i<j\leq k} \mathfrak{L}_T^{ij}$$

is $(k-1)$-partite with $(k-1)$-partition $U_{2,T} \cup \cdots \cup U_{k,T}$. Note that $\mathfrak{L}_T \subseteq \mathfrak{P}_T$.

We now make further preparations by considering some constants. For $T \in \binom{X}{1} \cup \binom{X}{2} = \binom{V_1}{1} \cup \binom{V_1}{2}$, set

$$(76) \qquad \lambda_T^{ij} = \frac{|\mathfrak{L}_T^{ij}|}{|\mathfrak{P}_T^{ij}|},\ 1 < i < j \leq k, \quad \lambda_0 = \frac{\alpha^3}{2}, \quad \delta_{\lambda_0} = 4\delta^{1/7}, \quad p = \frac{1}{\ell}, \quad \varepsilon_p = 4\ell^2\varepsilon.$$

Recall from the hierarchy in (55) that

$$\frac{1}{k}, \alpha \gg \delta \geq \min\{\ell^{-1}, \delta\} \gg \varepsilon.$$

As such, we are easily afforded the hierarchy

$$(77) \qquad \frac{1}{k-1}, \lambda_0 = \frac{\alpha^3}{2} \gg \delta_{\lambda_0} = 4\delta^{1/7} \geq \min\{p = \ell^{-1}, \delta_{\lambda_0} = 4\delta^{1/7}\} \gg \varepsilon_p = 4\ell^2\varepsilon.$$

As such, the quantification of constants in Lemma 5.4 implies that

(78) *constants $k-1$, λ_0, δ_{λ_0}, p, ε_p (defined in* (76)*) are sufficient for an application of Lemma* 5.4.

We conclude our preparations with the following fact.

Fact 5.6. *For $t = 1, 2$, all but $(k-1)2^t\varepsilon\binom{n}{t}$ sets $T \in \binom{X}{t} = \binom{V_1}{t}$ satisfy that $\mathfrak{P}_T^{ij}$ is (p, ε_p)-regular for all $1 < i < j \leq k$.*

Indeed, we see from (39) and (41) that all but $(k-1)2^t\varepsilon\binom{n}{t}$ sets $T \in \binom{X}{t} = \binom{V_1}{t}$ satisfy that for all $1 < i \leq k$,

$$(79) \qquad m_{i,T} = |U_{i,T}| = \left(\frac{1}{\ell} \pm \varepsilon\right)^t n = \frac{n}{\ell^t}\left(1 \pm \ell t\varepsilon\right).$$

As a consequence (cf. (39), (41)) these same sets $T \in \binom{X}{t} = \binom{V_1}{t}$, $t = 1, 2$, satisfy that for all $1 < i < j \leq k$, $\mathfrak{P}_T^{ij}$ is $(\ell^{-1}, (2\ell)^t\varepsilon)$-regular, or, in the language above, $\mathfrak{P}_T^{ij}$ is (p, ε_p)-regular, $1 < i < j \leq k$.

We now prove (66) and (68), in reverse order.

5.4. Proof of (68).

We wish to apply Lemma 5.4 to the graphs $\mathfrak{L} = L_{xy}$, $\{x, y\} \in \binom{X}{2} = \binom{V_1}{2}$. The following claims qualify the pairs for which this end is appropriate.

Claim 5.7. *All but $3\delta^{1/3}\binom{k-1}{2}\binom{n}{2}$ pairs $\{x, y\} \in \binom{X}{2} = \binom{V_1}{2}$ satisfy*

$$(80) \qquad \lambda_{xy}^{ij} \overset{(76)}{=} \frac{|\mathfrak{L}_{xy}^{ij}|}{|\mathfrak{P}_{xy}^{ij}|} = \alpha^2\left(1 \pm 4\delta^{1/3}\right) \overset{(55)}{\geq} \frac{\alpha^3}{2} \overset{(76)}{=} \lambda_0$$

for all $1 < i < j \leq k$.

Claim 5.8. *All but $2\delta^{1/6}(k-1)^2\binom{n}{2}$ pairs $\{x, y\} \in \binom{X}{2} = \binom{V_1}{2}$ satisfy the following property:*

all but $\delta_{\lambda_0}\binom{m_{i,\{x,y\}}}{2}$ *pairs* $\{a,b\} \in \binom{U_{i,\{x,y\}}}{2}$ *satisfy*

$$\deg_{\mathfrak{L}_{xy}^{ij}}(a,b) = \left(\lambda_{xy}^{ij} p\right)^2 m_{j,\{x,y\}} \left(1 \pm \delta_{\lambda_0}\right) \tag{81}$$

for all $1 < i \neq j \leq k$.

Before verifying Claims 5.7 and 5.8, let us use them to prove (68).

Claims 5.7 and 5.8 confirm that the hypothesis of Lemma 5.4 is met by the graph $\mathfrak{L}_{xy}$ for "most" choices of $\{x,y\} \in \binom{X}{2} = \binom{V_1}{2}$. Indeed, Claim 5.7 confirms that property 1 of Lemma 5.4 is met for all but $3\delta^{1/3}\binom{k-1}{2}\binom{n}{2}$ pairs $\{x,y\} \in \binom{V_1}{2}$. Claim 5.8 confirms that property 2 of Lemma 5.4 is met for all but an additional $2(k-1)^2\delta^{1/6}\binom{n}{2}$ pairs $\{x,y\} \in \binom{V_1}{2}$. By Fact 5.6, we see that all but $4(k-1)\varepsilon\binom{n}{2}$ pairs $\{x,y\} \in \binom{V_1}{2}$ satisfy property 3 of Lemma 5.4. Finally, (78) confirms that our set of constants is sufficient for an application of Lemma 5.4. As such, for all but

$$3\delta^{1/3}\binom{k-1}{2}\binom{n}{2} + 2\delta^{1/6}(k-1)^2\binom{n}{2} + 4(k-1)\varepsilon\binom{n}{2} \leq 9\delta^{1/6}k^2n^2 \overset{(55)}{\leq} \delta^{1/8}n^2$$

pairs $\{x,y\} \in \binom{X}{2} = \binom{V_1}{2}$, the graphs $\mathfrak{L}_{xy} = L_{xy}$ and $\mathfrak{P}_{xy} = P_{xy}$ satisfy the hypothesis of Lemma 5.4 with appropriate constants. Fixing one such pair $\{x,y\} \in \binom{X}{2} = \binom{V_1}{2}$, Lemma 5.4 yields

$$\begin{aligned}
&\deg_\Lambda(x,y) \overset{(73)}{=} \left|\mathcal{K}_{k-1}^{(2)}(L_{xy})\right| = \left|\mathcal{K}_{k-1}^{(2)}(\mathfrak{L}_{xy})\right| \\
&= \left(\prod_{1<i<j\leq k} \lambda_{xy}^{ij}\right) p^{\binom{k-1}{2}} \left(\prod_{1<i\leq k} m_{i,xy}\right) \left(1 \pm \delta_{\lambda_0}^{\frac{1}{5(k-1)}}\right) \\
&\overset{(75),\,(76),\,(79),\,(80)}{=} \left[\alpha^2\left(1 \pm 4\delta^{1/3}\right)\right]^{\binom{k-1}{2}} \frac{1}{\ell^{\binom{k-1}{2}}} \left[\frac{n}{\ell^2}\left(1 \pm 2\ell\varepsilon\right)\right]^{k-1} \left[1 \pm \left(4\delta^{1/7}\right)^{\frac{1}{5(k-1)}}\right] \\
&\qquad = \left(\frac{\alpha^2}{\ell}\right)^{\binom{k-1}{2}} \left(\frac{n}{\ell^2}\right)^{k-1} \left(1 \pm \delta^{\frac{1}{36(k-1)}}\right), \qquad (82)
\end{aligned}$$

as promised in (68).

It now remains to prove Claims 5.7 and 5.8. Claims 5.7 and 5.8 are ensured by Propositions 4.5 and 4.6, respectively, from section 4. We emphasize the following remark for future reference.

Remark 5.9. Claims 5.7 and 5.8 are guaranteed by applying statement 2 of Proposition 4.5 and statement 2(a) of Proposition 4.6, respectively.

In what immediately follows, we easily (and simultaneously) check that these lemmas may be applied in our current context. Afterward, we confirm that Propositions 4.5 and 4.6 indeed yield Claims 5.7 and 5.8.

5.4.1. Applying Propositions 4.5 and 4.6. Fix $1 < i < j \leq k$. We check that statement 2 of Proposition 4.5 and statement 2(a) of Proposition 4.6 (see section 4) may be applied to $\mathcal{H}^{1ij}$ and $P^{1i} \cup P^{1j} \cup P^{ij}$. Note that our hypothesis in Theorem 2.7, the counting lemma, includes that $\mathcal{H}^{1ij}$ and $P^{1i} \cup P^{1j} \cup P^{ij}$ satisfy the assumptions in Setup 2.1 with constants α, ℓ, and ε, as required by statement 2 of Proposition 4.5 and statement 2(a) of Proposition 4.6. Moreover, our hypothesis in Theorem 2.7 includes that $\mathcal{H}^{1ij}$ is (α,δ)-minimal w.r.t. $P^{1i} \cup P^{1j} \cup P^{ij}$, or in the language of section 4, $\mathcal{H}^{1ij}$ is $(\alpha,\delta)_2$-minimal w.r.t. $P^{1i} \cup P^{1j} \cup P^{ij}$, as required by statement 2 of Proposition 4.5

and statement 2(a) of Proposition 4.6. Our hypothesis in Theorem 2.7 also includes that our constants α, δ, ℓ, ε, and n satisfy the hierarchy in (55), and as such, satisfy the quantifications of Propositions 4.5 and 4.6. We conclude that statement 2 of Proposition 4.5 and statement 2(a) of Proposition 4.6 may be applied to $\mathcal{H}^{1ij}$ and $P^{1i} \cup P^{1j} \cup P^{ij}$.

5.4.2. Proof of Claim 5.7. We simply apply statement 2 of Proposition 4.5 for fixed $1 < i < j \leq k$. This statement guarantees that all but $3\delta^{1/3}\binom{n}{2}$ pairs $\{x, y\} \in \binom{X}{2} = \binom{V_1}{2}$ satisfy

$$\lambda_{xy}^{ij} \overset{(76)}{=} \frac{|\mathfrak{L}_{xy}^{ij}|}{|\mathfrak{P}_{xy}^{ij}|} = \frac{|L_{xy}^{ij}|}{|P_{xy}^{ij}|} = \alpha^2 \left(1 \pm 4\delta^{1/3}\right).$$

Thus, all but $3\delta^{1/3}\binom{k-1}{2}\binom{n}{2}$ pairs satisfy the above inequalities for all $1 < i < j \leq k$.

5.4.3. Proof of Claim 5.8. We shall use statement 2(a) of Proposition 4.6 in the context of a proof by contradiction.

Assume, on the contrary, that there exist $2\delta^{1/6}(k-1)^2\binom{n}{2}$ pairs $\{x, y\} \in \binom{X}{2} = \binom{V_1}{2}$ for which there exist $1 < i \neq j \leq k$ for which some $\delta_{\lambda_0}\binom{m_{i,\{x,y\}}}{2}$ pairs $\{a, b\} \in \binom{U_{i,\{x,y\}}}{2}$ satisfy

$$\deg_{\mathfrak{L}_{xy}^{ij}}(a, b) \neq (\lambda_{xy}^{ij} p)^2 m_{j,\{x,y\}}(1 \pm \delta_{\lambda_0}). \tag{83}$$

As such, for some fixed pair of indices $1 < i < j \leq k$, there must exist $2\delta^{1/6}\binom{n}{2}$ pairs $\{x, y\} \in \binom{X}{2}$ (the set of which we denote by $X(i, j)$) for which there exist at least $\delta_{\lambda_0}\binom{m_{i,\{x,y\}}}{2}$ pairs $\{a, b\} \in \binom{U_{i,\{x,y\}}}{2}$ (the set of which we denote by $U(i, x, y)$) satisfying

$$\deg_{\mathfrak{L}_{xy}^{ij}}(a, b) \neq (\lambda_{xy}^{ij} p)^2 m_{j,\{x,y\}}(1 \pm \delta_{\lambda_0}). \tag{84}$$

We show that the existence of the set $X(i, j)$, as described above, contradicts statement 2(a) of Proposition 4.6.

Our first step is to refine the set $X(i, j)$ down to suitable pairs. Denote by $X'(i, j) \subseteq X(i, j)$ the set of those pairs $\{x, y\}$ for which

$$\lambda_{xy}^{ij} = \alpha^2(1 \pm 4\delta^{1/3}) \quad \text{and} \quad m_{i,\{x,y\}}, m_{j,\{x,y\}} = \left(\frac{1}{\ell} \pm \varepsilon\right)^2 n. \tag{85}$$

We claim

$$|X'(i, j)| \geq \delta^{1/6}\binom{n}{2}. \tag{86}$$

Indeed, by Claim 5.7, we lose $3\delta^{1/3}\binom{k-1}{2}\binom{n}{2}$ pairs from $X(i, j)$ on account of the left condition of (85) failing. By the (ℓ^{-1}, ε)-regularity of graphs P^{1i} and P^{1j}, we lose another $8\varepsilon\binom{n}{2}$ pairs from $X(i, j)$ on account of the right condition of (85) failing. This shows

$$|X'(i, j)| \geq |X(i, j)| - 3\delta^{1/3}\binom{k-1}{2}\binom{n}{2} - 8\varepsilon\binom{n}{2} \geq |X(i, j)| - 4\delta^{1/3}\binom{k-1}{2}\binom{n}{2},$$

where the last inequality holds with $\varepsilon \ll \delta$ (cf. (55)). On account of our assumption that $|X(i, j)| \geq 2\delta^{1/6}\binom{n}{2}$, we see that (86) holds with $\delta > 0$ sufficiently small (cf. (55)).

We claim the set

$$C_4(1,i) = \{(\{x,y\},\{a,b\}) :\ \{x,y\} \in X'(i,j),\ \{a,b\} \in U(i,x,y)\} \subseteq \mathcal{K}^{(2)}_{2,2}(P^{1i})$$

is in contradiction with statement 2(a) of Proposition 4.6. (Note that every element $(\{x,y\},\{a,b\}) \in C_4(1,i)$ corresponds to a copy of C_4, with vertices x,y,a,b and edges $\{x,a\},\{a,y\},\{y,b\},\{b,x\}$, where $x,y \in V_1$ and $a,b \in V_i$.) In particular, we claim that

$$|C_4(1,i)| > 3\delta^{1/3}\left|\mathcal{K}^{(2)}_{2,2}(P^{1i})\right| \tag{87}$$

and that for each $(\{x,y\},\{a,b\}) \in C_4(1,i)$,

$$\deg_{L^{ij}_{xy}}(a,b) \neq \left(\frac{\alpha^2}{\ell}\right)^2 \frac{n}{\ell^2}\left(1 \pm 3\delta^{1/3}\right). \tag{88}$$

Then (87) and (88) are in violation with statement 2(a) of Proposition 4.6. As such, establishing (87) and (88) shows that our assumption in (83) is incorrect, and, therefore, proves Claim 5.8.

To see (87), note that, on account of (84)–(86), we have

$$|C_4(1,i)| \geq \delta^{1/6}\binom{n}{2} \times \delta_{\lambda_0}\binom{(\frac{1}{\ell}-\varepsilon)^2 n}{2} \overset{(55),(76)}{\geq} \delta^{13/42}\frac{n^4}{2\ell^4} \overset{(43),(55)}{>} 3\delta^{1/3}\left|\mathcal{K}^{(2)}_{2,2}(P^{1i})\right|. \tag{89}$$

We now verify (88). Fix $(\{x,y\},\{a,b\}) \in C_4(1,i)$. Since

$$\deg_{L^{ij}_{xy}}(a,b) = \deg_{\mathfrak{L}^{ij}_{xy}}(a,b) \neq (\lambda^{ij}_{xy}p)^2 m_{j,\{x,y\}}(1 \pm \delta_{\lambda_0}),$$

we have either

$$\deg_{L^{ij}_{xy}}(a,b) < (\lambda^{ij}_{xy}p)^2 m_{j,\{x,y\}}(1-\delta_{\lambda_0}) \quad \text{or} \quad \deg_{L^{ij}_{xy}}(a,b) > (\lambda^{ij}_{xy}p)^2 m_{j,\{x,y\}}(1+\delta_{\lambda_0}).$$

Without loss of generality, we assume the former inequality holds and will prove that the former inequality implies

$$\deg_{L^{ij}_{xy}}(a,b) < \left(\frac{\alpha^2}{\ell}\right)^2 \frac{n}{\ell^2}\left(1 - 3\delta^{1/3}\right). \tag{90}$$

This will complete our proof of (88).

Indeed, by (85), where $\lambda^{ij}_{xy} = \alpha^2(1 \pm 4\delta^{1/3})$, and since in (76), where we set $\delta_{\lambda_0} = 4\delta^{1/7}$ and $p = 1/\ell$, we have

$$\begin{aligned}\deg_{L^{ij}_{xy}}(a,b) < (\lambda^{ij}_{xy}p)^2 m_{j,\{x,y\}}(1-\delta_{\lambda_0}) &\leq \left(\frac{\alpha^2}{\ell}\left(1+4\delta^{1/3}\right)\right)^2\left(\frac{1}{\ell}+\varepsilon\right)^2 n\left(1-4\delta^{1/7}\right)\\ &< \left(\frac{\alpha^2}{\ell}\right)^2\frac{n}{\ell^2}\left(1-3\delta^{1/3}\right),\end{aligned}$$

where we used $0 < \varepsilon \ll \ell^{-1}, \delta$ and $\delta > 0$ sufficiently small, as given in (55). This proves (90).

5.5. Proof of (66). The proof of (66) is nearly identical to what we did above in section 5.4, save one detail: *we argue that all steps above can be done for a "typical" vertex* $x \in X = V_1$ *rather than for a "typical" pair* $\{x, y\} \in \binom{X}{2} = \binom{V_1}{2}$. More formally, we assert the following claims.

Claim 5.10. *All but* $3\delta^{1/3}\binom{k-1}{2}n$ *vertices* $x \in X = V_1$ *satisfy*

$$\lambda_x^{ij} \overset{(76)}{=} \frac{|\mathfrak{L}_x^{ij}|}{|\mathfrak{P}_x^{ij}|} = \alpha\left(1 \pm 4\delta^{1/3}\right) \overset{(55)}{\geq} \frac{\alpha^3}{2} \overset{(76)}{=} \lambda_0 \tag{91}$$

for all $1 < i < j \leq k$.

Claim 5.11. *All but* $2(k-1)^2\delta^{1/6}n$ *many vertices* $x \in X = V_1$ *satisfy the following property:*

all but $\delta_{\lambda_0}\binom{m_{i,x}}{2}$ *pairs* $\{a, b\} \in U_{i,x}$ *satisfy*

$$\deg_{\mathfrak{L}_x^{ij}}(a, b) = \left(\lambda_x^{ij}p\right)^2 m_{j,x}\left(1 \pm \delta_{\lambda_0}\right) \tag{92}$$

for all $1 < i \neq j \leq k$.

Precisely as we did in section 5.4, Claims 5.10 and 5.11, Fact 5.6, and the hierarchy in (77) say that for all but

$$3\delta^{1/3}\binom{k-1}{2}n + 2(k-1)^2\delta^{1/6}n + 2(k-1)\varepsilon n \leq 7\delta^{1/6}k^2n^2 \overset{(55)}{\leq} \delta^{1/8}n$$

vertices $x \in X = V_1$, the graphs $\mathfrak{L}_x = L_x$ and $\mathfrak{P}_x = P_x$ satisfy the hypothesis of Lemma 5.4 with appropriate constants. Fixing one such vertex $x \in X = V_1$, Lemma 5.4 yields (with $t = k - 1$)

$$\begin{aligned}
\deg_\Lambda(x) &\overset{(73)}{=} \left|\mathcal{K}_{k-1}^{(2)}(L_x)\right| = \left|\mathcal{K}_{k-1}^{(2)}(\mathfrak{L}_x)\right| \\
&= \left(\prod_{1<i<j\leq k} \lambda_x^{ij}\right) p^{\binom{k-1}{2}} \left(\prod_{1<i\leq k} m_{i,x}\right)\left(1 \pm \delta_{\lambda_0}^{\frac{1}{5(k-1)}}\right) \\
&\overset{(75),(76),(79),(91)}{=} \left[\alpha\left(1 \pm 4\delta^{1/3}\right)\right]^{\binom{k-1}{2}} \frac{1}{\ell^{\binom{k-1}{2}}} \left[\frac{n}{\ell}\left(1 \pm \ell\varepsilon\right)\right]^{k-1} \left[1 \pm \left(4\delta^{1/7}\right)^{\frac{1}{5(k-1)}}\right] \\
&= \left(\frac{\alpha}{\ell}\right)^{\binom{k-1}{2}} \left(\frac{n}{\ell}\right)^{k-1} \left(1 \pm \delta^{\frac{1}{36(k-1)}}\right),
\end{aligned} \tag{93}$$

as promised in (66).

As before, Claims 5.10 and 5.11 follow from Propositions 4.5 and 4.6, respectively, from section 4. We stress, importantly, that

we now seek to apply statement 1 *of Proposition* 4.5 *and statement* 1(a) *of Proposition* 4.6

(recall Remark 5.9). As such, verifying that we may apply these statements of Propositions 4.5 and 4.6 requires the additional attention of one small detail which we now consider. Fix $1 < i < j \leq k$. We wish to repeat the same verification we did in section 5.4.1. For this, we need that $\mathcal{H}^{1ij}$ is $(\alpha, \delta)_1$-minimal w.r.t. $P^{1i} \cup P^{1j} \cup P^{ij}$, a condition not initially assumed in the hypothesis of Theorem 2.7, but which follows by an application of Proposition 4.4.

6. Proof of Lemma 5.4. In this section, we prove Lemma 5.4. For simplicity, we prove Lemma 5.4 in the special case that $\lambda = \lambda_0 = \lambda_{ij}$, $1 \leq i < j \leq t$, and $m_i = m$ for all $1 \leq i \leq t$. Note that we also applied Lemma 5.4 in essentially the same special case (cf. (79) and Claims 5.7 and 5.10).

Our proof follows by induction on t where the base case $t = 2$ is trivial. We assume Lemma 5.4 holds up through $t-1 \geq 2$ and consider Lemma 5.4 for $t \geq 3$. We do not wish to begin our fairly simple argument with a tedious determination of constants. As such, with integer t given above, let constants λ, $\delta_{0,\lambda}$, δ_λ, p, ε_p, and m be given satisfying the hierarchy

$$\frac{1}{t}, \lambda \gg \delta_{0,\lambda} > \delta_\lambda \geq \min\{\delta_\lambda, p\} \gg \varepsilon_p \gg \frac{1}{m}, \tag{94}$$

which we note is consistent with the quantification of Lemma 5.4. With constants t, $\lambda = \lambda_0 = \lambda_{ij}$, $1 \leq i < j \leq t$, δ_λ, p, ε_p, and m, let graphs

$$\mathfrak{L} = \bigcup_{1 \leq i < j \leq t} \mathfrak{L}^{ij} \subseteq \mathfrak{P} = \bigcup_{1 \leq i < j \leq t} \mathfrak{P}^{ij}$$

on t-partition

$$U_1 \cup \cdots \cup U_t, \quad |U_1| = \cdots = |U_1| = m$$

be given as in the hypothesis of Lemma 5.4. We show

$$\left|\mathcal{K}_t^{(2)}(\mathfrak{L})\right| = (\lambda p)^{\binom{t}{2}} m^t \left(1 \pm \delta_\lambda^{\frac{1}{5t}}\right), \tag{95}$$

as promised by Lemma 5.4.

To prove Lemma 5.4, we first define auxiliary bipartite graphs $\mathbb{L} \subseteq \mathbb{P}$. As we see momentarily, these graphs formulate Lemma 5.4 in slightly different language.

Construction 6.1. *With t-partition $U_1 \cup \cdots \cup U_t$ of graphs $\mathfrak{L} = \bigcup_{1 \leq i < j \leq t} \mathfrak{L}^{ij}$ and $\mathfrak{P} = \bigcup_{1 \leq i < j \leq t} \mathfrak{P}^{ij}$, define auxiliary bipartite graphs $\mathbb{L} \subseteq \mathbb{P}$ with bipartition $A \cup B$ as follows:*

- $$A = U_1 \quad \textit{and} \quad B = \mathcal{K}_{t-1}^{(2)}\Big(\mathfrak{P}\Big[U_2, \dots, U_t\Big]\Big) = \mathcal{K}_{t-1}^{(2)}\Big(\bigcup_{1 < i < j \leq t} \mathfrak{P}^{ij}\Big),$$

  where elements of B, each denoted by K^-, correspond to vertex sets of $(t-1)$-cliques $K_{t-1}^{(2)}$ in the $(t-1)$-partite graph $\mathfrak{P}\Big[U_2, \dots, U_t\Big] = \bigcup_{1 < i < j \leq t} \mathfrak{P}^{ij}$. Note that

  $$|A| = m \quad \textit{and} \quad |B| \overset{\text{Fact 1.2}}{=} p^{\binom{t-1}{2}} m^{t-1} \left(1 \pm \varepsilon_p^{\frac{1}{t-1}}\right). \tag{96}$$

- *For $a \in A$ and $K^- \in B$,*

  $$\begin{aligned} \{a, K^-\} \in \mathbb{P} &\iff \{a\} \cup K^- \in \mathcal{K}_t^{(2)}(\mathfrak{P}) \iff K^- \subset N_{\mathfrak{P}}(a) \\ &\iff K^- \in \mathcal{K}_{t-1}^{(2)}\Big(\mathfrak{P}[N_{\mathfrak{P}}(a)]\Big), \end{aligned}$$

  where $\mathfrak{P}[N_{\mathfrak{P}}(a)]$ is the subgraph of $\mathfrak{P}$ induced on the $\mathfrak{P}$-neighborhood $N_{\mathfrak{P}}(a) = N_{\mathfrak{P}^{12}}(a) \cup \cdots \cup N_{\mathfrak{P}^{1t}}(a)$ of a.

- *For $a \in A$ and $K^- \in B$,*

$$\{a, K^-\} \in \mathbb{L} \iff K^- \subset N_{\mathfrak{L}}(a) \iff K^- \in \mathcal{K}_{t-1}^{(2)}\Big(\mathfrak{P}[N_{\mathfrak{L}}(a)]\Big)$$

where $\mathfrak{P}[N_{\mathfrak{L}}(a)]$ is the subgraph of $\mathfrak{P}$ induced on the $\mathfrak{L}$-neighborhood $N_{\mathfrak{L}}(a) = N_{\mathfrak{L}^{12}}(a) \cup \cdots \cup N_{\mathfrak{L}^{1t}}(a)$ of a.

Note that $\mathfrak{L} \subseteq \mathfrak{P}$ implies $\mathbb{L} \subseteq \mathbb{P}$. The following fact is identical to Fact 5.2.

Fact 6.2. *With ε_p given in* (94)*, the graph $\mathbb{P}$ is $(p^{t-1}, \varepsilon_p^{1/3})$-regular.*

We now make a few easy observations establishing connections between the graphs $\mathbb{L}$ and $\mathbb{P}$ and Lemma 5.4. For the purpose of stating these observations, set

$$B_{\mathfrak{L}} \overset{\text{def}}{=} \mathcal{K}_{t-1}^{(2)}\Big(\mathfrak{L}\Big[U_2, \dots, U_t\Big]\Big) = \mathcal{K}_{t-1}^{(2)}\Big(\bigcup_{1<i<j\leq t} \mathfrak{L}^{ij}\Big)$$

and note that elements of $B_{\mathfrak{L}}$ correspond to $(t-1)$-cliques $K_{t-1}^{(2)}$ in the $(t-1)$-partite graph $\mathfrak{L}\Big[U_2, \dots, U_t\Big] = \bigcup_{1<i<j\leq t} \mathfrak{L}^{ij}$.

Observations.

- *Since $\mathfrak{L} \subseteq \mathfrak{P}$, $B_{\mathfrak{L}} \subseteq B$.*
- *By our induction hypothesis on Lemma* 5.4 *(for $t-1$),*

$$\Big|B_{\mathfrak{L}}\Big| = \Big|\mathcal{K}_{t-1}^{(2)}\Big(\mathfrak{L}\Big[U_2, \dots, U_t\Big]\Big)\Big| = \Big|\mathcal{K}_{t-1}^{(2)}\Big(\bigcup_{1<i<j\leq t} \mathfrak{L}^{ij}\Big)\Big|$$

$$(97) \qquad = (\lambda p)^{\binom{t-1}{2}} m^{t-1}\left(1 \pm \delta_\lambda^{\frac{1}{5(t-1)}}\right) \overset{(94),\,(96)}{=} \lambda^{\binom{t-1}{2}} |B| \left(1 \pm 2\delta_\lambda^{\frac{1}{5(t-1)}}\right).$$

-

$$(98) \qquad \Big|\mathcal{K}_t^{(2)}(\mathfrak{L})\Big| = \sum_{K^- \in B_{\mathfrak{L}}} \deg_{\mathbb{L}}(K^-).$$

We see from (97) and (98) that, to prove Lemma 5.4, it suffices to analyze the terms in the sum (98). Proposition 6.3 does precisely this (and is similar to Proposition 5.3 of the preceding section).

Proposition 6.3. *All but $2\delta_\lambda^{1/12}|B| = 2\delta_\lambda^{1/12}|\mathcal{K}_{t-1}^{(2)}(\bigcup_{1<i<j\leq t} \mathfrak{P}^{ij})|$ many vertices $K^- \in B = \mathcal{K}_{t-1}^{(2)}(\bigcup_{1<i<j\leq t} \mathfrak{P}^{ij})$ satisfy*

$$(99) \qquad \deg_{\mathbb{L}}(K^-) = (\lambda p)^{t-1}|A|\left(1 \pm 2\delta_\lambda^{1/12}\right) = (\lambda p)^{t-1} m \left(1 \pm 2\delta_\lambda^{1/12}\right).$$

We defer the proof of Proposition 6.3 to section 6.2 and proceed now with the easy confirmation that Lemma 5.4 follows from Proposition 6.3.

6.1. Proposition 6.3 $\Longrightarrow$ Lemma 5.4. Our proof that follows is quite similar to what we saw when using Proposition 5.3 to prove Theorem 2.7.

We employ $\deg_{\mathbb{L}}(K^-) \leq \deg_{\mathbb{P}}(K^-)$ for all $K^- \in B_{\mathfrak{L}}$ not satisfying (99). As such, we note from Fact 6.2 that all but $2\varepsilon_p^{1/3}|B|$ vertices $K^- \in B$ satisfy

$$(100) \qquad \deg_{\mathbb{P}}(K^-) = p^{t-1}|A|\left(1 \pm \varepsilon_p^{1/3}\right) \overset{(94)}{\leq} 2p^{t-1}m.$$

Now, set

$$B_{\mathbb{L}} = \{K^- \in B : K^- \text{ satisfies } (99)\} \quad \text{and} \quad B_{\mathbb{P}} = \{K^- \in B : K^- \text{ satisfies } (100)\}.$$

Proposition 6.3 and (100) then imply

$$\begin{aligned}|B_{\mathfrak{L}} \setminus B_{\mathbb{L}}| &\le |B \setminus B_{\mathbb{L}}| \le 2\delta_\lambda^{1/12}|B| \overset{(96)}{\le} 4\delta_\lambda^{1/12} p^{\binom{t-1}{2}} m^{t-1},\\ |B_{\mathfrak{L}} \setminus B_{\mathbb{P}}| &\le 2\varepsilon_p^{1/3}|B| \le 4\varepsilon_p^{1/3} p^{\binom{t-1}{2}} m^{t-1}.\end{aligned} \tag{101}$$

Now the proof of Lemma 5.4 is immediate. Returning to (98), we see

$$\left|\mathcal{K}_t^{(2)}\left(\mathfrak{L}\right)\right| = \sum_{K^- \in B_{\mathfrak{L}}} \deg_{\mathbb{L}}\left(K^-\right)$$

$$= \sum_{K^- \in (B_{\mathfrak{L}} \cap B_{\mathbb{L}})} \deg_{\mathbb{L}}\left(K^-\right) + \sum_{K^- \in (B_{\mathbb{P}} \setminus B_{\mathbb{L}}) \cap B_{\mathfrak{L}}} \deg_{\mathbb{L}}\left(K^-\right) + \sum_{K^- \in (B_{\mathfrak{L}} \setminus (B_{\mathbb{L}} \cup B_{\mathbb{P}}))} \deg_{\mathbb{L}}\left(K^-\right). \tag{102}$$

To obtain the formula for $|\mathcal{K}_t^{(2)}(\mathfrak{L})|$ promised in (95), we need to bound (102) from above and below. For the lower bound, we employ (99) in (102) to obtain

$$\begin{aligned}\left|\mathcal{K}_t^{(2)}\left(\mathfrak{L}\right)\right| &\ge (\lambda p)^{t-1} m \left(1 - 2\delta_\lambda^{1/12}\right) |B_{\mathfrak{L}} \cap B_{\mathbb{L}}|\\ &= (\lambda p)^{t-1} m \left(1 - 2\delta_\lambda^{1/12}\right) \left(|B_{\mathfrak{L}}| - |B_{\mathfrak{L}} \setminus B_{\mathbb{L}}|\right).\end{aligned}$$

By (97) and (101), we then see

$$\left|\mathcal{K}_t^{(2)}\left(\mathfrak{L}\right)\right| \ge (\lambda p)^{\binom{t}{2}} m^t \left(1 - 2\delta_\lambda^{1/12}\right) \left(1 - \delta_\lambda^{\frac{1}{5(t-1)}} - 4\frac{\delta_\lambda^{1/12}}{\lambda^{\binom{t-1}{2}}}\right) \ge (\lambda p)^{\binom{t}{2}} m^t \left(1 - \delta_\lambda^{\frac{1}{5t}}\right),$$

which holds with $t \ge 3$ and $0 < \delta_\lambda \ll \lambda, t^{-1}$ sufficiently small in (94).

For the upper bound, we employ (99) and (100) in (102) to see that

$$\begin{aligned}\left|\mathcal{K}_t^{(2)}\left(\mathfrak{L}\right)\right| &\le (\lambda p)^{t-1} m \left(1 + 2\delta_\lambda^{1/12}\right) |B_{\mathfrak{L}} \cap B_{\mathbb{L}}| + 2p^{t-1} m |\left(B_{\mathbb{P}} \setminus B_{\mathbb{L}}\right) \cap B_{\mathfrak{L}}|\\ &\qquad + m|B_{\mathfrak{L}} \setminus \left(B_{\mathbb{L}} \cup B_{\mathbb{P}}\right)|\\ &\le (\lambda p)^{t-1} m \left(1 + 2\delta_\lambda^{1/12}\right) |B_{\mathfrak{L}}| + 2p^{t-1} m |B_{\mathfrak{L}} \setminus B_{\mathbb{L}}| + m|B_{\mathfrak{L}} \setminus B_{\mathbb{P}}|.\end{aligned}$$

Using (97) and (101), we obtain

$$\begin{aligned}\left|\mathcal{K}_t^{(2)}\left(\mathfrak{L}\right)\right| &\le (\lambda p)^{\binom{t}{2}} m^t \left(1 + 2\delta_\lambda^{1/12}\right) \left(1 + \delta_\lambda^{\frac{1}{5(t-1)}}\right) + 8\delta_\lambda^{1/12} p^{\binom{t}{2}} m^t + 4\varepsilon_p^{1/3} p^{\binom{t-1}{2}} m^t\\ &= (\lambda p)^{\binom{t}{2}} m^t \left[\left(1 + 2\delta_\lambda^{1/12}\right)\left(1 + \delta_\lambda^{\frac{1}{5(t-1)}}\right) + \frac{8\delta_\lambda^{1/12}}{\lambda^{\binom{t}{2}}} + \frac{4\varepsilon_p^{1/3}}{p^{t-1}}\right] \overset{(94)}{\le} (\lambda p)^{\binom{t}{2}} m^t \left(1 + \delta_\lambda^{\frac{1}{5t}}\right),\end{aligned}$$

which holds with $t \ge 3$ and the hierarchy in (94). This completes the proof of the induction step for Lemma 5.4.

6.2. Proof of Proposition 6.3. All that remains is to prove Proposition 6.3. We do so by proving the following slightly stronger version of Proposition 6.3 (which is similar to how we handled Proposition 5.3 through Proposition 5.3*).

Proposition 6.3*.

- *All but* $4\delta_\lambda^{1/3}m$ *vertices* $u \in A = U_1$ *satisfy*
  (103)
  $$\deg_{\mathbb{L}}(u) = p^{\binom{t-1}{2}}(\lambda pm)^{t-1}\left(1 \pm (4t)^2\delta_\lambda^{1/3}\right) \overset{(94),(96)}{=} (\lambda p)^{t-1}|B|\left(1 \pm 2(4t)^2\delta_\lambda^{1/3}\right).$$
  In particular,
  $$(104) \qquad \sum_{K^- \in B} \deg_{\mathbb{L}}(K^-) > \left((\lambda p)^{t-1}|A|\right)|B|\left(1 - \delta_\lambda^{1/4}\right).$$
- *All but* $\delta_\lambda m^2$ *pairs* $u, v \in A = U_1$ *satisfy*
  (105)
  $$\deg_{\mathbb{L}}(u,v) = p^{\binom{t-1}{2}}\left((\lambda p)^2 m\right)^{t-1}\left(1 \pm t^2\delta_\lambda\right) \overset{(94),(96)}{=} (\lambda p)^{2(t-1)}|B|\left(1 \pm 2t^2\delta_\lambda\right).$$
  In particular,
  $$(106) \qquad \sum_{K^- \in B} \deg_{\mathbb{L}}(K^-)^2 \le \left((\lambda p)^{t-1}|A|\right)^2|B|\left(1 + 3\delta_\lambda^{1/2}\right)$$
  $$\overset{(94)}{<} \left((\lambda p)^{t-1}|A|\right)^2|B|\left(1 + \delta_\lambda^{1/4}\right).$$

It is easy to prove Proposition 6.3 from Proposition 6.3*. Indeed, using the approximate Cauchy–Schwarz inequality (that is, Fact 4.7), Proposition 6.3 is an immediate corollary of (104) and (106). (To see this, in Fact 4.7 set $a = (\lambda p)^{t-1}|A|$, $r = |B|$, and $\gamma = \delta_\lambda^{1/4}$, and let terms a_i, $1 \le i \le r$, correspond to terms $\deg_{\mathbb{L}}(K^-)$, $K^- \in B$.) It therefore remains to prove Proposition 6.3*.

To prove Proposition 6.3*, we must show four things: the assertions of (103) and (105) and the implications (103) $\Longrightarrow$ (104) and (105) $\Longrightarrow$ (106). We proceed to first prove the implications, and then we shall prove the assertions.

6.2.1. Proof that (103) $\Longrightarrow$ (104). Observe that

$$\sum_{K^- \in B} \deg_{\mathbb{L}}(K^-) = \sum_{u \in A} \deg_{\mathbb{L}}(u).$$

Now, denote by A_{good} the set of vertices $u \in A = U_1$ for which (103) holds. We then have

$$\sum_{K^- \in B} \deg_{\mathbb{L}}(K^-) = \sum_{u \in A} \deg_{\mathbb{L}}(u) \ge \sum_{u \in A_{\text{good}}} \deg_{\mathbb{L}}(u) \overset{(103)}{\ge} (\lambda p)^{t-1}|B|\left(1 - 2(4t)^2\delta_\lambda^{1/3}\right)|A_{\text{good}}|$$
$$\overset{\text{Prop. 6.3}^*}{\ge} (\lambda p)^{t-1}|B|\left(1 - 2(4t)^2\delta_\lambda^{1/3}\right)\left(1 - 4\delta_\lambda^{1/3}\right)|A| \overset{(94)}{\ge} \left((\lambda p)^{t-1}|A|\right)|B|\left(1 - \delta_\lambda^{1/4}\right),$$

as promised in (104).

6.2.2. Proof that (105) $\Longrightarrow$ (106). Recall that we are supposed to bound $\sum_{K^- \in B} \deg_{\mathbb{L}}(K^-)^2$ from above. First we observe that

$$(107) \quad \sum_{\{u,v\} \in \binom{A}{2}} \deg_{\mathbb{L}}(u,v) = \sum_{K^- \in B} \binom{\deg_{\mathbb{L}}(K^-)}{2} \overset{(94)}{=} \left(\frac{1}{2} - o(1)\right) \sum_{K^- \in B} \deg_{\mathbb{L}}(K^-)^2,$$

where $o(1) \to 0$ as $m \to \infty$. It therefore suffices to consider the sum $\sum_{\{u,v\}\in\binom{A}{2}} \dot{}$ $\deg_{\mathbb{L}}(u,v)$.

Denote by $\binom{A}{2}_{\rm good}$ the set of all pairs $\{u,v\} \in \binom{A}{2}$ for which (105) holds. For pairs $\{u,v\} \notin \binom{A}{2}_{\rm good}$, we observe $\deg_{\mathbb{L}}(u,v) \le \deg_{\mathbb{P}}(u,v)$ (since $\mathbb{L} \subseteq \mathbb{P}$), where we recall from Fact 6.2 that $\mathbb{P}$ is $(p^{t-1}, \varepsilon_p^{1/3})$-regular. As such (and similarly to (100)), we have that all but $4\varepsilon_p^{1/3}m^2$ pairs $\{u,v\} \in \binom{A}{2}$ satisfy

$$\deg_{\mathbb{P}}(u,v) = \left(p^{t-1} + \varepsilon_p^{1/3}\right)^2 |B| \overset{(94)}{\le} 2p^{2(t-1)}|B|. \tag{108}$$

Denote by $\binom{A}{2}_{\rm fair}$ the set of all pairs $\{u,v\} \in \binom{A}{2}$ for which (108) holds.

Returning to (107), we see that the sum $\sum_{\{u,v\}\in\binom{A}{2}} \deg_{\mathbb{L}}(u,v)$ equals

$$\begin{aligned}
&\sum_{\{u,v\}\in\binom{A}{2}_{\rm good}} \deg_{\mathbb{L}}(u,v) + \sum_{\{u,v\}\in\binom{A}{2}_{\rm fair}\setminus\binom{A}{2}_{\rm good}} \deg_{\mathbb{L}}(u,v) \\
&\qquad + \sum_{\{u,v\}\in\binom{A}{2}\setminus\left(\binom{A}{2}_{\rm fair}\cup\binom{A}{2}_{\rm good}\right)} \deg_{\mathbb{L}}(u,v) \\
&\le \sum_{\{u,v\}\in\binom{A}{2}_{\rm good}} \deg_{\mathbb{L}}(u,v) + \sum_{\{u,v\}\in\binom{A}{2}_{\rm fair}\setminus\binom{A}{2}_{\rm good}} \deg_{\mathbb{P}}(u,v) + |B| \left|\binom{A}{2} \setminus \binom{A}{2}_{\rm fair}\right| \\
&\overset{\text{Prop. 6.3}^*,\,(108)}{\le} (\lambda p)^{2(t-1)}|B| \left(1 + 2t^2\delta_\lambda\right) \left|\binom{A}{2}_{\rm good}\right| + 2p^{2(t-1)}|B| \left|\binom{A}{2} \setminus \binom{A}{2}_{\rm good}\right| \\
&\qquad + 4\varepsilon_p^{1/3}m^2|B| \\
&\overset{\text{Prop. 6.3}^*}{\le} (\lambda p)^{2(t-1)}|B| \left(1 + 2t^2\delta_\lambda\right) \binom{m}{2} + 2\delta_\lambda p^{2(t-1)}|B|m^2 + 4\varepsilon_p^{1/3}m^2|B| \\
&\le (\lambda p)^{2(t-1)}|B|m^2 \left(\left(\frac{1}{2} + t^2\delta_\lambda\right) + 2\frac{\delta_\lambda}{\lambda^{2(t-1)}} + 4\frac{\varepsilon_p^{1/3}}{(\lambda p)^{2(t-1)}}\right) \\
&\overset{(94)}{<} \left((\lambda p)^{t-1}|A|\right)^2 |B| \left(\frac{1}{2} + \delta_\lambda^{1/2}\right),
\end{aligned}$$

so that, with (107), we have

$$\begin{aligned}
\sum_{K^-\in B} \deg_{\mathbb{L}}(K^-)^2 &\le \left((\lambda p)^{t-1}|A|\right)^2 |B| \left(1 + 2\delta_\lambda^{1/2}\right)(1 + o(1)) \\
&\overset{(94)}{<} \left((\lambda p)^{t-1}|A|\right)^2 |B| \left(1 + 3\delta_\lambda^{1/2}\right),
\end{aligned}$$

as promised in (106).

It remains to prove (103) and (105). We begin with the latter.

6.2.3. Proof of (105). For an arbitrary pair of vertices $u, v \in A = U_1$, observe that

$$\deg_{\mathbb{L}}(u,v) = \left|\mathcal{K}_{t-1}^{(2)}\Big(\mathfrak{P}\left[N_{\mathfrak{L}}(u,v)\right]\Big)\right| = \left|\mathcal{K}_{t-1}^{(2)}\Big(\bigcup_{1<i<j\le t} \mathfrak{P}^{ij}\left[N_{\mathfrak{L}^{1i}}(u,v), N_{\mathfrak{L}^{1j}}(u,v)\right]\Big)\right| \tag{109}$$

follows by our construction of graph $\mathbb{L}$. Estimating $|\mathcal{K}_{t-1}^{(2)}(\mathfrak{P}[N_{\mathfrak{L}}(u,v)])|$ is, however, a mere application of Fact 1.2.

Indeed, the hypothesis of Lemma 5.4 gives that all but $\delta_\lambda m^2$ pairs $u, v \in U_1 = A$ satisfy $\deg_{\mathfrak{L}^{1i}}(u,v) = (\lambda p)^2 m(1 \pm \delta_\lambda)$, $1 < i \le t$. Fix one such pair $u, v \in U_1 = A$. We claim u, v satisfy the conclusion of Proposition 6.3*. Indeed, fix $1 < i < j \le t$ and observe that

$$\min\{\deg_{\mathfrak{L}^{1i}}(u,v),\ \deg_{\mathfrak{L}^{1j}}(u,v)\} \ge (\lambda p)^2 m(1-\delta_\lambda) \overset{(94)}{\ge} \frac{1}{2}(\lambda p)^2 m \overset{(94)}{\gg} \varepsilon_p^{1/2} m. \tag{110}$$

Using (110), the (p, ε_p)-regularity of $\mathfrak{P}^{ij}$ implies that the graph $\mathfrak{P}^{ij}\,[N_{\mathfrak{L}^{1i}}(u,v), N_{\mathfrak{L}^{1j}}(u,v)]$ is $(p, \varepsilon_p^{1/2})$-regular (see Fact 1.5, the slicing lemma, from [26]). As such, we apply Fact 1.2 to conclude

$$\begin{aligned}\left|\mathcal{K}_{t-1}^{(2)}\Big(\mathfrak{P}\,[N_{\mathfrak{L}}(u,v)]\Big)\right| \;&=\; p^{\binom{t-1}{2}}\,\big[(\lambda p)^2 m\,(1 \pm \delta_\lambda)\big]^{t-1}\left(1 \pm \varepsilon_p^{\frac{1}{2(t-1)}}\right)\\ &\overset{(94)}{=} p^{\binom{t-1}{2}}\,\big[(\lambda p)^2 m\big]^{t-1}\,\big(1 \pm t^2\delta_\lambda\big)\end{aligned}$$

which renders the result.

6.2.4. Proof of (103). We will follow essentially the same procedure as described above, but for single vertices $u \in A = U_1$ rather than pairs $u, v \in A = U_1$. Note that, however, in this case, Lemma 5.4 admits no hypothesis on $\deg_{\mathfrak{L}^{1i}}(u)$ for single vertices $u \in A = U_1$. For this reason, we require the following fact.

FACT 6.4. *With $\mathfrak{L}$ and $\mathfrak{P}$ given in Lemma* 5.4, *all but $4\delta_\lambda^{1/3} m$ vertices $u \in U_1$ satisfy that for each $1 < i \le t$,*

$$\deg_{\mathfrak{L}^{1i}}(u) = \lambda p m\left(1 \pm 4\delta_\lambda^{1/3}\right).$$

Note that Fact 6.4 gives the analogue of condition 2 of Lemma 5.4 for single vertices $u \in U_1$, $1 \le i < t$. The proof of Fact 6.4 follows from conditions 1 and 2 of Lemma 5.4 by a standard Cauchy–Schwarz argument. We omit the standard details. We now use Fact 6.4 to finish the proof of Proposition 6.3*.

Similarly to (109), for a fixed vertex $u \in A = U_1$, we have

$$\deg_{\mathbb{L}}(u) = \left|\mathcal{K}_{t-1}^{(2)}\Big(\mathfrak{P}\,[N_{\mathfrak{L}}(u)]\Big)\right| = \left|\mathcal{K}_{t-1}^{(2)}\Big(\bigcup_{1<i<j\le t} \mathfrak{P}^{ij}\,[N_{\mathfrak{L}^{1i}}(u), N_{\mathfrak{L}^{1j}}(u)]\Big)\right|.$$

Fact 6.4 ensures that all but $4\delta_\lambda^{1/3} m$ vertices $u \in U_1$ satisfy that for all $1 < i < j \le t$,

$$\min\{\deg_{\mathfrak{L}^{1i}}(u),\ \deg_{\mathfrak{L}^{1j}}(u)\} \ge \lambda p m\left(1 - 4\delta_\lambda^{1/3}\right) \overset{(94)}{\ge} \frac{1}{2}\lambda p m \overset{(94)}{\gg} \varepsilon_p^{1/2} m.$$

Fix one such vertex $u \in U_1$. Similar to (110), the (p, ε_p)-regularity of $\mathfrak{P}^{ij}$, $1 < i < j \le t$, implies that the graph $\mathfrak{P}^{ij}\,[N_{\mathfrak{L}^{1i}}(u) N_{\mathfrak{L}^{1j}}(u)]$ is $(p, \varepsilon_p^{1/2})$-regular. As such, we apply Fact 1.2 to conclude that

$$\begin{aligned}\left|\mathcal{K}_{t-1}^{(2)}\Big(\mathfrak{P}\,[N_{\mathfrak{L}}(u)]\Big)\right| \;&=\; p^{\binom{t-1}{2}}\left[\lambda p m\left(1 \pm 4\delta_\lambda^{1/3}\right)\right]^{t-1}\left(1 \pm \varepsilon_p^{\frac{1}{2(t-1)}}\right)\\ &\overset{(94)}{=} p^{\binom{t-1}{2}}\,(\lambda p m)^{t-1}\left(1 \pm (4t)^2\delta_\lambda^{1/3}\right),\end{aligned}$$

which renders the result.

7. Proof of statement 2 of Proposition 3.2. In section 2.4, we asserted that statement 2 of Proposition 3.2 follows from Lemma 5.8 (cf. Algorithm A) of [8]. (Proposition 3.2 and Lemma 5.8 of [8] are very similar statements, with almost the same quantification.) In this section, we show that the earlier Lemma 5.8 implies Proposition 3.2. We begin by presenting Lemma 5.8 from [8], as well as the upcoming Fact 7.3 that we shall also employ in our proof.

7.1. Background material. To state Lemma 5.8 of [8], we need the following concept.

Definition 7.1 ((γ, δ, R)-regular). *Let bipartite graph F have bipartition $U \cup W$. For given constants $\gamma, \delta > 0$ and for given integer R, we say that F is (γ, δ, R)*-regular *if, for all $U_1, \ldots, U_R \subseteq U$ and all $W_1, \ldots, W_R \subseteq W$ for which*

$$\left| \bigcup_{i=1}^{R} (U_i \times W_i) \right| > \delta |U||W|,$$

we have

$$\left| F \cap \bigcup_{i=1}^{R} (U_i \times W_i) \right| = \gamma(1 \pm \delta) \left| \bigcup_{i=1}^{R} U_i \times W_i \right|.$$

(Here, for a fixed $1 \leq i \leq R$, we define $U_i \times W_i$ to be $\{\{u_i, w_i\} : u_i \in U_i,\ w_i \in W_i\}$.)

Lemma 5.8 of [8] is then stated[2] as follows. (In what follows, we use primes for some of our constants to distinguish them from their corresponding constants in Proposition 3.2.)

Lemma 7.2. *For all $\alpha, \delta'_B > 0$ there exists $\delta'_A > 0$ such that for all integers ℓ and r'_B, there exist $\varepsilon' > 0$ and integer r'_A so that whenever $\mathcal{H}$ and P satisfy the hypothesis of Setup* 2.1 *with constants α, ℓ, and ε' and with n sufficiently large, the following holds.*

Suppose there exist $\delta'_B n^2$ pairs $\{x, y\} \in \binom{V_1}{2}$ for which L_{xy} is not $(\alpha^2/\ell, \delta'_B, r'_B)$-regular and for which witnesses $U_1^{xy}, \ldots, U_{r'_B}^{xy} \subseteq N_{P^{12}}(x, y)$ and $W_1^{xy}, \ldots, W_{r'_B}^{xy} \subseteq N_{P^{13}}(x, y)$ against the $(\alpha^2/\ell, \delta'_B, r'_B)$-regularity of L_{xy} are given. Then $\mathcal{H}$ is not (δ'_A, r'_A)-regular w.r.t. P, and there exists an algorithm $\mathbf{A}_{7.2}$ which, in time $O(n^5)$, converts the witnesses $U_1^{xy}, \ldots, U_{r'_B}^{xy}$ and $W_1^{xy}, \ldots, W_{r'_B}^{xy}$, over the $\delta'_B n^2$ pairs $\{x, y\}$ above, into a witness $\mathbf{Q}_{r'_A} = (Q_1, \ldots, Q_{r'_A})$ of the (δ'_A, r'_A)-irregularity of $\mathcal{H}$ w.r.t. P.

To prove that statement 2 of Proposition 3.2 follows from Lemma 7.2, we shall need the following auxiliary fact. (This fact will allow us to build, in the context of Lemma 7.2, the witnesses $U_1^{xy}, \ldots, U_{r'_B}^{xy}$ and $W_1^{xy}, \ldots, W_{r'_B}^{xy}$.)

Fact 7.3. *For all $\beta, \delta_2 > 0$, there exists $\delta_1 > 0$ so that for all $d > 0$, there exist integer R and $\zeta > 0$ so that the following holds: let $F \subseteq G$ be bipartite graphs with bipartition $U \cup W$, where $|U|, |W|$ are sufficiently large, and G is (d, ζ)-regular. If there exist $\delta_2 |U|^2$ pairs $\{u, u'\} \in \binom{U}{2}$ satisfying*

$$\deg_F(u, u') \neq (\beta d)^2 |W| (1 \pm \delta_2),$$

then F is not $(\beta d, \delta_1, R)$-regular, and there exists an algorithm $\mathbf{A}_{7.3}$ which, in time $O(|U|^2 |W|)$, constructs witnesses $U_1, \ldots, U_R \subseteq U$ and $W_1, \ldots, W_R \subseteq W$ of the $(\beta d, \delta_1, R)$-irregularity of F.

[2] In [8], a slightly stronger statement is proved which includes an additional parameter β. To derive Lemma 7.2 from the original Lemma 5.8 of [8], one sets $\beta = \delta'_B/2$ and then uses that we may take $\varepsilon < \delta'_B/2$. (For the definition of V_1^{good} in Lemma 5.8 of [8], see Definition 4.4 [8, p. 303].)

Fact 7.3 was proved, without any focus on the algorithmic assertion, as Fact 8.12 [30, p. 395]. For completeness, we repeat the proof at the end of the paper and emphasize the algorithmic aspects.

7.2. Proof of Proposition 3.2. We begin our proof by formally describing the constants involved. The reader not interested in the determination of these constants may skip ahead.

7.2.1. Constants. As in the quantification of Proposition 3.2, let $\alpha, \delta_B > 0$ be given. In Fact 7.3, put $\beta = \alpha^2$ and

$$\delta_2 = \frac{\delta_B^3}{1000} \leq \frac{\delta_B^2}{4}. \tag{111}$$

Let

$$\delta_1 = \delta_1^{(7.3)}(\alpha^2, \delta_2) \leq \frac{\delta_B^2}{2} \tag{112}$$

be the constant guaranteed by Fact 7.3 (where we may assume, without loss of generality, that $\delta_1 \leq \delta_B^2/2$). Putting $\delta_B' = \delta_1$, let

$$\delta_A' = \delta_{A,(7.2)}'(\alpha, \delta_B') \tag{113}$$

be the constant guaranteed by Lemma 7.2. For Proposition 3.2, we take the promised constant δ_A as

$$\delta_A = \delta_A' = \delta_{A,(7.2)}'(\alpha, \delta_B'). \tag{114}$$

Now, as in Proposition 3.2, let integer ℓ be given. In Fact 7.3, set $d = 1/\ell$ and let

$$\zeta = \zeta^{(7.3)}(\alpha^2, \delta_2, \delta_1, 1/\ell) \quad \text{and} \quad R = R^{(7.3)}(\alpha^2, \delta_2, \delta_1, 1/\ell) \tag{115}$$

be the constants guaranteed by Fact 7.3. In Lemma 7.2, set $r_B' = R$ and let

$$\varepsilon_{A,(7.2)}' = \varepsilon_{A,(7.2)}'(\alpha, \delta_B', \delta_A', \ell, r_B'), \qquad r_{A,(7.2)}' = r_{A,(7.2)}'(\alpha, \delta_B', \delta_A', \ell, r_B') \tag{116}$$

be the constants guaranteed by Lemma 7.2. Let

$$\varepsilon^{(4.6)} = \varepsilon^{(4.6)}(\alpha, \delta_B, \ell)$$

be the constant guaranteed by Proposition 4.6. For Proposition 3.2, we take

$$r = r_{A,(7.2)}' \quad \text{and} \quad \varepsilon = \min\{\varepsilon_{A,(7.2)}', \zeta^2, \varepsilon^{(4.6)}\}. \tag{117}$$

This concludes our definitions of the constants.

7.2.2. The algorithm. We now prove statement 2 of Proposition 3.2. Let $\mathcal{H}$ and $P = P^{12} \cup P^{23} \cup P^{13}$ be given as in Setup 2.1 with constants $\alpha, \delta_B, \delta_A, \ell, r, \varepsilon$ described in (111)–(117). Suppose $\mathcal{H}$ is (α, δ_B)-excessive w.r.t. P. We establish an algorithm $\mathbf{A}_{3.2}$ which, in time $O(n^5)$, constructs a witness $\mathbf{Q}_r = (Q_1, \ldots, Q_r)$ of the (δ_A, r)-irregularity of $\mathcal{H}$ w.r.t. P. In what immediately follows, we list the steps of the algorithm $\mathbf{A}_{3.2}$. Afterward, we fill in all the details.

Algorithm $\mathbf{A}_{3.2}$.

Given. *$\mathcal{H}$ and P, as in the hypothesis of Proposition* 3.2, *where $\mathcal{H}$ is (α, δ_B)-excessive w.r.t. P.*

Output. *In time $O(n^5)$, a witness $\mathbf{Q}_r = (Q_1, \ldots, Q_r)$ of the (δ_A, r)-irregularity of $\mathcal{H}$ w.r.t. P.*

Steps.

1. Statement 2(b) of Proposition 4.6 guarantees $\delta_2 n^2$ pairs $\{x,y\} \in \binom{V_1}{2}$ so that for each such $\{x,y\}$, there are $\delta_2[\deg_{P^{12}}(x,y)]^2$ pairs $\{a,b\} \in \binom{N_{P^{12}}(x,y)}{2}$, each of which satisfies
$$\deg_{L_{xy}}(a,b) \neq \left(\frac{\alpha^2}{\ell}\right)^2 \deg_{P^{13}}(x,y)(1 \pm \delta_2).$$
The pairs $\{x,y\}$ and corresponding pairs $\{a,b\}$ can be found in time $O(n^5)$.
2. For a fixed $\{x,y\}$ from Step 1, Algorithm $\mathbf{A}_{7.3}$ (cf. Fact 7.3) constructs, in time $O(n^3)$, witnesses $U_1^{xy}, \dots, U_R^{xy}$ and $W_1^{xy}, \dots, W_R^{xy}$ of the $(\alpha^2/\ell, \delta_1, R)$-irregularity of L_{xy}. Over all $\{x,y\}$ from Step 1, we build a family of witnesses in time $O(n^5)$.
3. Algorithm $\mathbf{A}_{7.2}$ (cf. Lemma 7.2) converts, in time $O(n^5)$, the witnesses $U_1^{xy}, \dots, U_R^{xy}$ and $W_1^{xy}, \dots, W_R^{xy}$, over all $\{x,y\}$ from Step 1, into a witness $\mathbf{Q}_r = (Q_1, \dots, Q_r)$ of the (δ_A, r)-irregularity of $\mathcal{H}$ w.r.t. P.

We now establish the steps above, beginning with the first.

Step 1. We appeal to statement 2(b) of Proposition 4.6, setting, in that context, $\delta = \delta_B$. Since $\mathcal{H}$ is (α, δ_B)-excessive w.r.t. P, statement 2(b) of Proposition 4.6 guarantees at least (with $\varepsilon \ll 1/\ell$)

$$\frac{\delta_B^3}{27}|\mathcal{K}_{2,2}^{(2)}(P^{12})| \overset{(43)}{\geq} \frac{\delta_B^3}{28\ell^4}\binom{n}{2}^2 \geq \frac{\delta_B^3}{115\ell^4}n^4$$

elements $(\{x,y\},\{a,b\}) \in \mathcal{K}_{2,2}^{(2)}(P^{12})$ for which

$$\deg_{L_{xy}}(a,b) \neq \left(\frac{\alpha^2}{\ell}\right)^2 |N_{P^{13}}(x,y)| \left(1 \pm 2\frac{\delta_B^3}{27}\right).$$

Our choice of $\delta_2 < 2\delta_B^3/(27)$ in (111) then ensures that these elements also satisfy

$$\deg_{L_{xy}}(a,b) \neq \left(\frac{\alpha^2}{\ell}\right)^2 |N_{P^{13}}(x,y)| \left(1 \pm \delta_2\right). \tag{118}$$

These elements can be found in time $O(n^5)$. In the next two steps, we shall consider only those elements $(\{x,y\},\{a,b\})$ for which

$$\frac{n}{2\ell^2} \leq \deg_{P^{12}}(x,y),\ \deg_{P^{13}}(x,y) \leq 2\frac{n}{\ell^2}, \tag{119}$$

of which the (ℓ^{-1}, ε)-regularity of P^{12} and P^{13} implies there must be at least (using $\varepsilon \ll 1/\ell$)

$$\frac{\delta_B^3}{115\ell^4}n^4 - 4\varepsilon n^4 > \frac{\delta_B^3}{120\ell^4}n^4 \tag{120}$$

many. Using (119) and (120), a simple pigeon-hole calculation then shows that there must be at least

$$\frac{\delta_B^3}{500}n^2 \overset{(111)}{\geq} \delta_2 n^2$$

pairs $\{x,y\} \in \binom{V_1}{2}$, each of which has at least

$$\frac{\delta_B^3}{250}\frac{n^2}{\ell^4} \geq \frac{\delta_B^3}{1000}\big[\deg_{P^{12}}(x,y)\big]^2 \overset{(111)}{=} \delta_2\big[\deg_{P^{12}}(x,y)\big]^2$$

pairs $\{a,b\} \in \binom{N_{P^{12}}(x,y)}{2}$ for which (118) holds. Let $\binom{V_1}{2}_{\mathrm{bad}}$ denote the set of these pairs $\{x,y\}$, and for each such $\{x,y\}$, let $\binom{N_{P^{12}}(x,y)}{2}_{\mathrm{bad}}$ denote its set of corresponding pairs $\{a,b\}$.

Step 2. Fix a pair $\{x,y\} \in \binom{V_1}{2}_{\mathrm{bad}}$. As in Fact 7.3, set $\beta = \alpha^2$, $d = \ell^{-1}$, and $\zeta = \varepsilon^{1/2}$, and set

$$F = L_{xy}, \quad G = P_{xy}, \quad U = N_{P^{12}}(x,y), \quad W = N_{P^{13}}(x,y).$$

Since the graph P_{xy} is $(1/\ell, \varepsilon^{1/2})$-regular (cf. (41)) where $\varepsilon \leq \zeta^2$ (see (117)), the graph G is (d,ζ)-regular. The set $\binom{N_{P^{12}}(x,y)}{2}_{\mathrm{bad}} = \binom{U}{2}_{\mathrm{bad}}$ corresponds to a collection of $\delta_2|U|^2$ pairs $\{u,u'\} \in \binom{U}{2}$ for which

$$\deg_F(u,u') = \deg_{L_{xy}}(u,u') \neq \left(\frac{\alpha^2}{\ell}\right)|N_{P^{13}}(x,y)|(1\pm\delta_2) = (\beta d)^2|W|(1\pm\delta_2).$$

By our choice of δ_1 in (112), R in (115), and ζ in (115), Fact 7.3 applies to say that the graph $F = L_{xy}$ is $(\alpha^2/\ell, \delta_1, R)$-irregular. Moreover, Algorithm $\mathbf{A}_{7.3}$ constructs, in time $O(|U|^2|W|) = O(n^3)$, witnesses $U_1^{xy},\dots,U_R^{xy} \subseteq U$ and $W_1^{xy},\dots,W_R^{xy} \subseteq W$ of the $(\alpha^2/\ell, \delta_1, R)$-irregularity of L_{xy}. In time $O(n^5)$, we repeat this process over all $\{x,y\} \in \binom{V_1}{2}$.

Step 3. We apply Lemma 7.2 to $\mathcal{H}$, P, and the collection of witnesses $U_1^{xy},\dots,U_R^{xy}$ and $W_1^{xy},\dots,W_R^{xy}$ over $\{x,y\} \in \binom{V_1}{2}_{\mathrm{bad}}$. To begin, we recall that $\mathcal{H}$ and P are as in Setup 2.1 with constants α, ℓ, and ε. Step 2 constructed witnesses $U_1^{xy},\dots,U_R^{xy}$ and $W_1^{xy},\dots,W_R^{xy}$ of the $(\alpha^2/\ell, \delta_1, R)$-irregularity of L_{xy} for $\delta_1 n^2$ pairs $\{x,y\} \in \binom{V_1}{2}_{\mathrm{bad}}$. By our choice of $\delta_A = \delta_A'$ in (113) and (114) and $r_A = r'_{A,(7.2)}$ and $\varepsilon \leq \varepsilon'_{A,(7.2)}$ in (116) and (117), Lemma 7.2 applies to say that $\mathcal{H}$ is (δ_A, r)-irregular w.r.t. P. Moreover, Algorithm $\mathbf{A}_{7.2}$ converts the witnesses $U_1^{xy},\dots,U_R^{xy}$ and $W_1^{xy},\dots,W_R^{xy}$, over all $\{x,y\} \in \binom{V_1}{2}_{\mathrm{bad}}$, into a witness $\mathbf{Q}_r = (Q_1,\dots,Q_r)$ of the (δ_A, r)-irregularity of $\mathcal{H}$ w.r.t. P.

7.3. Proof of Fact 7.3. For simplicity, we shall give an informal description of the constants. Let $\beta, \delta_2 > 0$ be given. Without loss of generality, we shall assume that $\delta_2 \ll \beta$. Choose $0 < \delta_1 \ll \delta_2$. Let $d > 0$ be given. Set $R = \delta_2^3/d$ and choose $0 < \zeta \ll \min\{d, \delta_1\}$. With these constants, let F and G be given as in the hypothesis of Fact 7.3. We show that in time $O(|U|^2|W|)$, we may construct witnesses $U_1,\dots,U_R \subseteq U$ and $W_1,\dots,W_R \subseteq W$ against the $(\beta d, \delta_1, R)$-regularity of F.

Denote by U^- the set of vertices $u \in U$ for which $\deg_F(u) < \beta d|W|(1-\delta_1)$. If $|U^-| > \delta_1|U|$, then we are done. Indeed, we may take $U_1 = U^-$ and $W_1 = W$ so that the 1-tuple U_1, W_1 satisfies $|U_1||W_1| > \delta_1|U||W|$ but, by construction, $|F \cap (U_1 \times W_1)| < \beta d|U_1||W_1|(1-\delta_1)$. As such, F is not $(\beta d, \delta_1, 1)$-regular, and therefore, not $(\beta d, \delta_1, R)$-regular, and the witness U_1, W_1 is found in time $O(|U||W|)$. In the remainder of the proof, we shall therefore assume that $|U^-| \leq \delta_1|U|$.

Now, suppose that there exist $\delta_2|U|^2$ pairs $\{u,u'\} \in \binom{U}{2}$ for which

$$\deg_F(u,u') \neq (\beta d)^2|W|(1\pm\delta_2).$$

Then, at least $(\delta_2/2)|U|^2$ pairs must have codegree in F which is, say, too small, and these pairs can be found in time $O(|U|^2|V|)$. We can then find a set $U_0 \subseteq U$, $|U_0| \geq (\delta_2/4)|U|$, so that for each $u \in U_0$, there exists a set $U_u \subseteq U$, $|U_u| \geq (\delta_2/4)|U|$, so that for each $u' \in U_u$,

$$\deg_F(u, u') < (\beta d)^2|W|(1-\delta_2).$$

Moreover, the set U_0 and corresponding sets U_u, $u \in U_0$, can be found in time $O(|U|^2)$. We show that the graph F cannot be $(\beta d, \delta_1, R)$-regular, and we shall construct a witness $U_1, \ldots, U_R$, $W_1, \ldots, W_R$ to this effect.

To this end, we find in time $O(|U|^2|W|)$ a subset $U_0^* = \{u_1, \ldots, u_R\} \subset U_0 \setminus U^-$ with the property that for each $1 \leq i < j \leq R$,

$$\deg_G(u_i, u_j) \leq 2d^2|W|. \tag{121}$$

Indeed, it is well known from the (d, ζ)-regularity of G that all but $4\zeta|U|^2$ pairs u_i, u_j satisfy (121). Let Γ be the graph of the pairs u_i, u_j not satisfying (121). Pick $u_1 \in U_0 \setminus U^-$ to be any vertex with $\deg_\Gamma(u_1) \leq 3\zeta^{1/2}|U|$. There are at least

$$|U_0| - \delta_1|U| - 3\zeta^{1/2}|U| \geq \left(\frac{\delta_2}{4} - \delta_1 - 3\zeta^{1/2}\right)|U| > 0$$

choices for u_1. If $\{u_1, \ldots, u_{t-1}\} \subset U_0 \setminus U^-$ have already been chosen, $t-1 < R$, pick any $u_t \in U_0 \setminus (U^- \cup \bigcup_{1 \leq i \leq t-1} N_\Gamma(u_i))$ with $\deg_\Gamma(u_t) \leq 3\zeta^{1/2}|U|$. There are at least

$$\begin{aligned}|U_0| - \delta_1|U| - 3(t-1)\zeta^{1/2}|U| - 3\zeta^{1/2}|U| &\geq \left(\frac{\delta_2}{4} - \delta_1 - 3R\zeta^{1/2}\right)|U| \\ &= \left(\frac{\delta_2}{4} - \delta_1 - \frac{3\zeta^{1/2}}{d}\right)|U| > 0\end{aligned}$$

choices for u_t.

We now give the promised witness. For each $1 \leq i \leq R$, let $U_i = U_{u_i}$, where $u_i \in U_0^*$, and let $W_i = N_F(u_i)$. This R-tuple was constructed in time $O(|U|^2|W|)$. We now verify that the R-tuple is a witness for the $(\beta d, \delta_1, R)$-irregularity of F. To that end, we see by inclusion-exclusion that

$$\left|\bigcup_{i=1}^R (U_i \times W_i)\right| \geq \sum_{i=1}^R |U_i \times W_i| - \sum_{1 \leq i < j \leq R} \left|(U_i \times W_i) \cap (U_j \times W_j)\right|.$$

For a fixed $1 \leq i \leq R$, note that

$$|U_i \times W_i| = |U_i| \deg_F(u_i) \geq |U_i|(\beta d)|W|(1-\delta_1) \geq \frac{\delta_2 \beta d}{4}|U||W|(1-\delta_1) \geq \frac{\delta_2 \beta d}{8}|U||W|$$

and for a fixed $1 \leq i < j \leq R$,

$$\left|(U_i \times W_i) \cap (U_j \times W_j)\right| \leq |U| \deg_G(u_i, u_j) \leq 2d^2|U||W|.$$

As such, with $R = \delta_2^3/d$ and $\delta_1 \ll \delta_2 \ll \beta$, we have

$$\left|\bigcup_{i=1}^R (U_i \times W_i)\right| \geq \frac{\delta_2^4 \beta}{8}|U||W| - \delta_2^6|U||W| > \frac{\delta_2^4 \beta}{16}|U||W| > \delta_1|U||W|. \tag{122}$$

To finish verifying the witness, note that implicit in the work above is the inequality

$$\left|\bigcup_{i=1}^{R}\big(U_i \times W_i\big)\right| \geq \beta d|W|(1-\delta_1)\sum_{i=1}^{R}|U_i| - \delta_2^6|U||W|. \tag{123}$$

We have, with $\delta_1 \ll \delta_2 \ll \beta$,

$$\begin{aligned}\left|F \cap \bigcup_{i=1}^{R}\big(U_i \times W_i\big)\right| &\leq \sum_{i=1}^{R}\big|F \cap \big(U_i \times W_i\big)\big| = \sum_{i=1}^{R}\sum_{u' \in U_i}\deg_F(u_i, u') \\ &< \sum_{i=1}^{R}\sum_{u' \in U_i}(\beta d)^2|W|(1-\delta_2) = (\beta d)^2|W|(1-\delta_2)\sum_{i=1}^{R}|U_i| \\ &\overset{(123)}{\leq} (\beta d)^2|W|(1-\delta_2)\frac{\left[\left|\bigcup_{i=1}^{R}\big(U_i \times W_i\big)\right| + \delta_2^6|U||W|\right]}{\beta d|W|(1-\delta_1)} \\ &= \beta d\left(\frac{1-\delta_2}{1-\delta_1}\right)\left[1 + \frac{\delta_2^6|U||W|}{\left|\bigcup_{i=1}^{R}\big(U_i \times W_i\big)\right|}\right]\left|\bigcup_{i=1}^{R}\big(U_i \times W_i\big)\right| \\ &\overset{(122)}{\leq} \beta d\left(\frac{1-\delta_2}{1-\delta_1}\right)\left(1 + \frac{16\delta_2^2}{\beta}\right)\left|\bigcup_{i=1}^{R}\big(U_i \times W_i\big)\right| \\ &< \beta d\left(1 - \frac{\delta_2}{2}\right)\left|\bigcup_{i=1}^{R}\big(U_i \times W_i\big)\right| < \beta d(1-\delta_1)\left|\bigcup_{i=1}^{R}\big(U_i \times W_i\big)\right|.\end{aligned}$$

This completes the proof.

Acknowledgment. We would like to thank an anonymous referee for a careful reading of this paper, and for suggestions which led to a much improved exposition.

REFERENCES

[1] N. Alon, R. Duke, H. Lefmann, V. Rödl, and R. Yuster, *The algorithmic aspects of the Regularity Lemma*, J. Algorithms, 16 (1994), pp. 80–109.

[2] N. Alon, R. Duke, H. Lefmann, V. Rödl, and R. Yuster, *The algorithmic aspects of the Regularity Lemma (extended abstract)*, in Proceedings of the IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 1992, pp. 473–481.

[3] N. Alon, A. Shapira, and B. Sudakov, *Additive approximation for edge-deletion problems*, in Proceedings of the IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 2005, pp. 419–428.

[4] F. R. K. Chung, *Regularity lemmas for hypergraphs and quasi-randomness*, Random Structures Algorithms, 2 (1991), pp. 241–252.

[5] F. R. K. Chung and R. L. Graham, *Quasi-random set systems*, J. Amer. Math. Soc., 4 (1991), pp. 151–196.

[6] A. Czygrinow and B. Nagle, *On small subsystems of uniformly distributed* 3*-graphs*, submitted.

[7] A. Czygrinow and V. Rödl, *An algorithmic regularity lemma for hypergraphs*, SIAM J. Comput., 30 (2000), pp. 1041–1066.

[8] Y. Dementieva, P. Haxell, B. Nagle, and V. Rödl, *On characterizing hypergraph regularity*, Random Structures Algorithms, 21 (2002), pp. 293–395.

[9] D. Dor and M. Tarsi, *Graph decomposition is NPC—a complete proof of Holyer's conjecture*, in Proceedings of the ACM Symposium on Theory of Computing, ACM, New York, 1992, pp. 252–263.

[10] R. A. Duke, H. Lefmann, and V. Rödl, *A fast approximation algorithm for computing the frequencies of subgraphs of a given graph*, SIAM J. Comput., 24 (1995), pp. 598–620.

[11] P. Frankl and V. Rödl, *Extremal problems on set systems*, Random Structures Algorithms, 20 (2002), pp. 131–164.

[12] P. Frankl and V. Rödl, *The uniformity lemma for hypergraphs*, Graphs Combin., 8 (1992), pp. 309–312.

[13] A. Frieze and R. Kannan, *The Regularity Lemma and approximation schemes for dense problems*, in Proceedings of the IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 1996, pp. 12–20.

[14] A. Frieze and R. Kannan, *Quick approximation to matrices and applications*, Combinatorica, 19 (1999), pp. 175–220.

[15] W. T. Gowers, *Quasirandomness, counting and regularity for* 3*-uniform hypergraphs*, Combin. Probab. Comput., 15 (2006), pp. 143–184.

[16] W. T. Gowers, *Hypergraph Regularity and the Multidimensional Szemerédi Theorem*, preprint, Department of Pure Mathematics and Mathematical Statistics, University of Cambridge, Cambridge, UK, 2004.

[17] P. Haxell, B. Nagle, and V. Rödl, *An algorithmic version of the hypergraph regularity method (extended abstract)*, in Proceedings of the IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 2005, pp. 439–446.

[18] P. Haxell, B. Nagle, and V. Rödl, *Integer and fractional packings in dense* 3*-uniform hypergraphs*, Random Structures Algorithms, 22 (2003), pp. 248–310.

[19] P. Haxell and V. Rödl, *Integer and fractional packings in dense graphs*, Combinatorica, 21 (2001), pp. 13–38.

[20] Y. Kohayakawa, B. Nagle, and V. Rödl, *Hereditary properties of triple systems*, Combin. Probab. Comput., 12 (2003), pp. 155–189.

[21] Y. Kohayakawa, B. Nagle, and V. Rödl, *Efficient testing of hypergraphs*, in ICALP 2002, 29th International Colloquium on Automata, Languages and Programming (Malaga, Spain), Lecture Notes in Comput. Sci. 2286, Springer-Verlag, Berlin, 2002, pp. 278–292.

[22] Y. Kohayakawa, V. Rödl, and J. Skokan, *Equivalent conditions of hypergraph regularity*, J. Combin. Theory Ser. A, 97 (2002), pp. 307–352.

[23] Y. Kohayakawa, V. Rödl, and L. Thoma, *An optimal algorithm for checking regularity (extended abstract)*, in Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2002), San Francisco, CA, 2002, pp. 277–286.

[24] Y. Kohayakawa, V. Rödl, and L. Thoma, *An optimal algorithm for checking regularity*, SIAM J. Comput., 32 (2003), pp. 1210–1235.

[25] J. Komlós, A. Shokoufandeh, M. Simonovits, and E. Szemerédi, *The regularity lemma and its applications in graph theory*, in Theoretical Aspects of Computer Science (Tehran, 2000), Lecture Notes in Comput. Sci. 2292, Springer-Verlag, Berlin, 2002, pp. 84–112.

[26] J. Komlós and M. Simonovits, *Szemerédi's Regularity Lemma and its applications in graph theory*, in Combinatorics, Paul Erdös is Eighty, 2 (1996), pp. 295–352.

[27] B. Nagle and V. Rödl, *Regularity properties for triple systems*, Random Structures Algorithms, 23 (2003), pp. 264–332.

[28] B. Nagle and V. Rödl, *The asymptotic number of triple systems not containing a fixed one*, Discrete Math., 235 (2001), pp. 271–290.

[29] B. Nagle, V. Rödl, and M. Schacht, *The counting lemma for regular k-uniform hypergraphs*, Random Structures Algorithms, 26 (2006), pp. 1–67.

[30] Y. Peng, V. Rödl, and J. Skokan, *Counting small subgraphs in a* 3*-hypergraph*, Combin. Probab. Comput., 14 (2005), pp. 371–413.

[31] V. Rödl and A. Ruciński, *Ramsey properties of random hypergraphs*, J. Combin. Theory Ser. A, 81 (1998), pp. 1–33.

[32] V. Rödl, A. Ruciński, and E. Szemerédi, *A Dirac-type theorem for* 3*-uniform hypergraphs*, Combin. Probab. Comput., 15 (2006), pp. 229–251.

[33] V. Rödl, M. Schacht, M. Siggers, and N. Tokushige, *Fractional packings of hypergraphs*, J. Combin. Theory Ser. B, to appear.

[34] V. Rödl, M. Schacht, E. Tengan, and N. Tokushige, *Density theorems and the Hypergraph Regularity Method*, Israel J. Math., 152 (2006), pp. 371–380.

[35] V. Rödl and J. Skokan, *Regularity lemma for k-uniform hypergraphs*, Random Structures Algorithms, 25 (2004), pp. 1–42.

[36] V. Rödl and J. Skokan, *Applications of the regularity lemma for k-uniform hypergraphs*, Random Structures Algorithms, 26 (2006), pp. 180–194.

[37] J. Skokan and L. Thoma, *Bipartite subgraphs and quasi-randomness*, Graphs Combin., 20 (2004), pp. 255–262.

[38] J. Solymosi, *A note on a question of Erdos and Graham*, Combin. Probab. Comput., 13 (2004), pp. 263–267.

[39] E. Szemerédi, *On sets of integers containing no k elements in arithmetic progression*, Acta Arith., 27 (1975), pp. 199–245.

[40] E. Szemerédi, *Regular partitions of graphs*, in Problèmes Combinatoires et Théorie des Graphes (Colloq. Internat. CNRS, Univ. Orsay, Orsay, 1976), J.-C. Bermond, J.-C. Fournier, M. Las Vergnas, and D. Sotteau, eds., Colloq. Internat. CNRS 260, CNRS, Paris, 1978, pp. 399–401.

[41] T. Tao, *A variant of the hypergraph removal lemma*, J. Combin. Theory Ser. A, 113 (2006), pp. 1257–1280.

[42] T. Tao, *The Gaussian primes contain arbitrarily shaped constellations*, J. Anal. Math., to appear.

[43] A. Thomason, *Dense expanders and pseudo-random bipartite graphs*, in Graph Theory and Combinatorics (Cambridge, 1988), Discrete Math., 75 (1989), pp. 381–386.

SIAM J. COMPUT.
Vol. 37, No. 6, pp. 1777–1805

AGNOSTICALLY LEARNING HALFSPACES*

ADAM TAUMAN KALAI[†], ADAM R. KLIVANS[‡], YISHAY MANSOUR[§], AND ROCCO A. SERVEDIO[¶]

Abstract. We give a computationally efficient algorithm that learns (under distributional assumptions) a halfspace in the difficult *agnostic* framework of Kearns, Schapire, and Sellie [*Mach. Learn.*, 17 (1994), pp. 115–141], where a learner is given access to a distribution on labelled examples but where the labelling may be arbitrary (similar to malicious noise). It constructs a hypothesis whose error rate on future examples is within an additive ϵ of the optimal halfspace, in time poly(n) for any constant $\epsilon > 0$, for the uniform distribution over $\{-1, 1\}^n$ or unit sphere in $\mathbb{R}^n$, as well as any log-concave distribution in $\mathbb{R}^n$. It also agnostically learns Boolean disjunctions in time $2^{\tilde{O}(\sqrt{n})}$ with respect to *any* distribution. Our algorithm, which performs L_1 polynomial regression, is a natural noise-tolerant arbitrary-distribution generalization of the well-known "low-degree" Fourier algorithm of Linial, Mansour, and Nisan. We observe that significant improvements on the running time of our algorithm would yield the fastest known algorithm for learning parity with noise, a challenging open problem in computational learning theory.

Key words. agnostic learning, halfspaces, Fourier

AMS subject classification. 68Q32

DOI. 10.1137/060649057

1. Introduction.

Halfspaces have been used extensively in machine learning for decades. From the early work on the Perceptron algorithm in the 1950's, through the learning of artificial neural networks in the 1980's, and up to and including today's AdaBoost [14] and support vector machines [45], halfspaces have played a central role in the development of the field's most important tools.

Formally, a *halfspace* is a Boolean function $f(x) = \text{sgn}(\sum_{i=1}^{n} w_i x_i - \theta)$. While efficient algorithms are known for learning halfspaces if the data is guaranteed to be noise-free, learning a halfspace from noisy examples remains a challenging and important problem. Halfspace-based learning methods appear repeatedly in both theory and practice, and they are frequently applied to labeled data sets which are not linearly separable. This motivates the following natural and well-studied questions: What can one *provably* say about the performance of halfspace-based learning methods in the presence of noisy data or distributions that do not obey constraints induced by an unknown halfspace? Can we develop learning algorithms which tolerate data generated from a "noisy" halfspace and output a meaningful hypothesis?

1.1. Agnostic learning.

We consider the standard model in statistical learning theory, which is a natural model for learning from possibly noisy data. Kearns,

[*] Received by the editors January 4, 2006; accepted for publication (in revised form) February 13, 2007; published electronically March 26, 2008.

http://www.siam.org/journals/sicomp/37-6/64905.html

[†] College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 (atk@cc.gatech.edu). This author's research was supported in part by NSF award SES-0527656.

[‡] Department of Computer Science, University of Texas at Austin, Austin, TX 78712 (klivans@cs.utexas.edu). Part of this research was done while this author was visiting TTI-Chicago.

[§] Department of Computer Science, Tel Aviv University, 69978 Tel Aviv, Israel (mansour@cs.tau.ac.il).

[¶] Department of Computer Science, Columbia University, New York, NY 10027 (rocco@cs.columbia.edu). This author's research was supported in part by NSF CAREER award CCF-0347282 and a Sloan Foundation Fellowship.

Schapire, and Sellie [25] termed this *agnostic learning* and gave an elegant formalization of this model and defined what it means to be a computationally efficient learner. In this model the learner receives labeled examples (x, y) drawn from a fixed distribution over example-label pairs, but (in contrast with Valiant's standard probably approximately correct (PAC) learning model [43]) it is not necessarily the case that the labels y are generated by applying some target function f to the examples x. Of course, without any assumptions on the distribution it is impossible for the learner to always output a meaningful hypothesis. Kearns et al. instead require the learner to output a hypothesis whose accuracy with respect to future examples drawn from the distribution approximates that of the optimal concept from some fixed concept class of functions $\mathcal{C}$, such as the class of all halfspaces $f(x) = \operatorname{sgn}(v \cdot x - \theta)$. Given a concept class $\mathcal{C}$ and a distribution $\mathcal{D}$ over labeled examples (x, y), we write $\mathsf{opt} = \min_{f \in \mathcal{C}} \Pr_{\mathcal{D}}[f(x) \neq y]$ to denote the error rate of the optimal (smallest error) concept from $\mathcal{C}$ with respect to $\mathcal{D}$.

For intuition, one can view agnostic learning as a noisy learning problem in the following way: There is a distribution $\mathcal{D}$ over examples x and the data *is* assumed to be labeled according to a function $f \in \mathcal{C}$, but an adversary is allowed to corrupt an $\eta = \mathsf{opt}$ fraction of the labels given to the learning algorithm. The goal is to find a hypothesis h with error $\Pr_{\mathcal{D}}[h(x) \neq y]$ as close as possible to η. (We note that such a noise scenario is far more challenging than the *random classification noise* model, in which an η fraction of labels are flipped independently at random and for which a range of effective noise-tolerant learning algorithms are known [23, 4].)

Unfortunately, only few positive results are known for agnostically learning expressive concept classes. Kearns, Schapire, and Sellie [25] gave an algorithm for agnostically learning piecewise linear functions, and Goldman, Kearns, and Schapire [17] showed how to agnostically learn certain classes of geometric patterns. Lee, Bartlett, and Williamson [29] showed how to agnostically learn some very restricted classes of neural networks in time exponential in the fan-in. We note that standard results on the Perceptron algorithm and support vector machines [15, 41] give error rates for those algorithms in terms of the hinge loss of the optimal linear threshold function. Our goal is different, since we want to give a bound on the error rate that depends on—in fact, is almost identical to—the *error rate* (rather than the hinge loss) of the optimal linear threshold function.

1.2. Known negative results.

Some strong negative results are known for the case of *proper agnostic learning*, where the output hypothesis must belong to the concept class to be learned. Properly agnostically learning halfspaces, for example, is known to be NP-hard [18, 13]; it is even NP-hard to properly agnostically learn the concept class of disjunctions [25, 12]. More specifically, these results [18, 13] show that there exist distributions that are consistent with a halfspace on a $1 - \epsilon$ (for any $\epsilon > 0$) fraction of inputs, and it is NP-hard to output a halfspace with accuracy $1/2 + \epsilon$ with respect to this distribution (the same holds for even disjunctions [12]).

For nonproper learning, fewer negative results are known. Given the recent representation-independent hardness results for learning majorities of halfspaces [13] and intersections of halfspaces [28], it is easy to see (and pointed out in [13]) that a polynomial-time, distribution-free agnostic learning algorithm for halfspaces for $\epsilon = o(1/n^{\gamma})$ for some $\gamma > 0$ (regardless of the output representation of the hypothesis) would imply polynomial-time solutions to well-studied lattice problems. These lattice problems are thought to be intractable and form the basis of several recent public key cryptosystems (see, e.g., [39]). It is also known [30] that agnostically learning disjunctions, again with no restrictions on the hypotheses used, is at least as hard as PAC learning DNF formulas, a longstanding open question in learning theory.

Thus, it is natural to consider, as we do in this paper, agnostic learning with respect to various restricted distributions $\mathcal{D}$ for which the marginal distribution $\mathcal{D}_X$ over the example space X satisfies some prescribed property. This corresponds to a learning scenario in which the *labels* are arbitrary, but the distribution over *examples* is restricted.

1.3. Our main technique. The following two observations are the starting point of our work:

- The "low-degree" Fourier learning algorithm of Linial et al. can be viewed as an algorithm for performing L_2-norm *polynomial regression* under the uniform distribution on $\{-1,1\}^n$. (See section 2.2.)
- A simple analysis (Observation 3) shows that the low-degree algorithm has some attractive agnostic learning properties under the uniform distribution on $\{-1,1\}^n$. (See section 2.3.)

The "low-degree" algorithm, however, will achieve only partial results for agnostic learning (the output hypothesis will be within a factor of 8 of optimal). As described in section 3, the above two observations naturally motivate a new algorithm which can be viewed as an L_1-norm version of the low-degree algorithm; we call this simply the *polynomial regression algorithm.* (At this point, it may be slightly mysterious why the L_1-norm would be significantly better than the L_2-norm; we discuss this point in section 3.)

Roughly speaking, our main result about the polynomial regression algorithm, Theorem 5, shows the following (see section 3 for the detailed statement):

> Given a concept class $\mathcal{C}$ and a distribution $\mathcal{D}$, if concepts in $\mathcal{C}$ can be approximated by low-degree polynomials in the L_2-norm relative to the marginal distribution $\mathcal{D}_X$, then the L_1 polynomial regression algorithm is an efficient agnostic learning algorithm for $\mathcal{C}$ with respect to $\mathcal{D}$.

A long line of research has focused on how well the truncated Fourier polynomial over the parity basis approximates concept classes with respect to the L_2-norm; this has led to numerous algorithms for learning concepts with respect to the uniform distribution over the Boolean hypercube $\{-1,1\}^n$ [31, 8, 20, 22, 26]. For learning with respect to the uniform distribution on the unit sphere, our analysis uses the Hermite polynomials [42], a family of orthogonal polynomials with a weighting scheme related to the density function of the Gaussian distribution. As such, these polynomials are well suited for approximating concepts with respect to the L_2-norm over S^{n-1}. We believe this approach will find further applications in the future.

Additionally, we show that a slightly modified version of the wildly popular *support vector machine* (SVM) algorithm [45], with a polynomial kernel, can achieve the same result.[1] Unfortunately, with the number of examples we require for our analysis, the SVM algorithm is no more efficient than our simple polynomial regression algorithm (the "Kernel trick" does not help). But it is interesting to give strong provable guarantees about the agnostic learning ability of an algorithm that is so popular in practice.

1.4. Our main results. As described below, our main result about the polynomial regression algorithm can be applied to obtain many results for agnostic learning

[1] This was pointed out to us by Avrim Blum.

of halfspaces with respect to a number of different distributions, both discrete and continuous, some uniform and some nonuniform.

THEOREM 1. *Let $\mathcal{D}$ be a distribution over $\mathbb{R}^n \times \{-1,1\}$. The L_1 polynomial regression algorithm has the following properties: its runtime is polynomial in the number of examples it is given, and*

1. *if the marginal $\mathcal{D}_X$ is* (a) *uniform on $\{-1,1\}^n$* or (b) uniform on the unit sphere in $\mathbb{R}^n$*, then with probability $1-\delta$ the polynomial regression algorithm outputs a hypothesis with error $\mathsf{opt}+\epsilon$ given* $\mathrm{poly}(n^{1/\epsilon^4}, \log\frac{1}{\delta})$ *examples;*
2. *if the marginal $\mathcal{D}_X$ is* log-concave*, then with probability $1-\delta$ the polynomial regression algorithm outputs a hypothesis with error $\mathsf{opt}+\epsilon$ given* $\mathrm{poly}(n^{d(\epsilon)}, \log\frac{1}{\delta})$ *examples, where $d : \mathbb{R}_+ \to \mathbb{Z}_+$ is a universal function independent of $\mathcal{D}_X$ or n.*

Part 1(a) follows from our analysis of the L_1 polynomial regression algorithm combined with the Fourier bounds on halfspaces given by Klivans, O'Donnel, and Servedio [26]. Part 1(b) follows from the same analysis of the algorithm combined with concentration bounds over the n-dimensional sphere. In proving such bounds, we use the Hermite polynomial basis in analogy with the Fourier basis used previously. (We note that learning halfspaces under the uniform distribution on the sphere is a well-studied problem; see, e.g., [1, 2, 23, 32, 33].) As before, we show that a related algorithm gives a hypothesis with error $O(\mathsf{opt}+\epsilon)$ in time $n^{O(1/\epsilon^2)}$.

In section 4.2 we show that algorithms for agnostically learning halfspaces with respect to the uniform distribution on $\{0,1\}^n$ can be used to solve the well-known problem of learning parity functions with respect to random classification noise [6]. This indicates that substantially improving the results of part 1 of Theorem 1 may be very difficult. For example, improving our $n^{O(1/\epsilon^4)}$ time algorithm for agnostically learning halfspaces to accuracy $\mathsf{opt}+\epsilon$ (with respect to the uniform distribution over the hypercube) to an $n^{O(1/\epsilon^{2-\beta})}$ time algorithm ($\beta > 0$) would yield the fastest known algorithm for learning parity with noise.

As indicated by part 2 of Theorem 1, for any constant ϵ, we can also achieve a polynomial-time algorithm for learning with respect to any log-concave distribution. Recall that any Gaussian distribution, exponential distribution, and uniform distribution over a convex set is log-concave.

We next consider a simpler class of halfspaces: disjunctions on n variables. The problem of agnostically learning an unknown disjunction (or learning noisy disjunctions) has long been a difficult problem in computational learning theory and was recently reposed as a challenge by Avrim Blum in his FOCS 2003 tutorial [3]. By combining Theorem 5 with known constructions of low-degree polynomials that are good L_∞-approximators of the OR function, we obtain a subexponential time algorithm for agnostically learning disjunctions with respect to *any* distribution (recall that since this problem is at least as hard as PAC-learning DNF, given the current state of the art we do not expect to achieve a polynomial-time algorithm).

THEOREM 2. *Let $\mathcal{D}$ be a distribution on $X \times Y$, where $\mathcal{D}$ is an arbitrary distribution over $\{-1,1\}^n$ and $Y = \{-1,1\}$. For the class of disjunctions, with probability $1-\delta$ the polynomial regression algorithm outputs a hypothesis with error $\leq \mathsf{opt}+\epsilon$ in time $2^{\tilde{O}(\sqrt{n}\cdot\log(1/\epsilon))}\cdot \mathrm{poly}(\log\frac{1}{\delta})$.*

1.5. Extensions and other applications. We believe that the polynomial regression algorithm will have many extensions and applications; so far we have explored only a few of these, which we now describe.

In section 4.3 we show how our approach can be used to improve the algorithm due to Klivans, O'Donnel, and Servedio [26] for learning intersections of halfspaces with respect to the uniform distribution over the hypercube.

In section 5 we give a detailed analysis of an algorithm, which is essentially the same as the degree-1 version of the polynomial regression algorithm, for agnostic learning the concept class of origin-centered halfspaces $\operatorname{sgn}(v \cdot x)$ over the uniform distribution on the unit sphere, $S^{n-1} = \{x \in \mathbb{R}^n \mid \|x\| = 1\}$. (Similar results also hold for the ball $\{x \in \mathbb{R}^n \mid \|x\| \leq 1\}$.) While our analysis from section 3 implies only that this algorithm should achieve some fixed constant error $\Theta(1)$ independent of opt, we are able to show that in fact we do much better if opt is small.

THEOREM 3. *Let $\mathcal{D}$ be a distribution on $X \times Y$, where $Y = \{-1, 1\}$ and the marginal $\mathcal{D}_X$ is uniform on the sphere S^{n-1} in $\mathbb{R}^n$. There is a simple algorithm for agnostically learning origin-centered halfspaces with respect to $\mathcal{D}$ which uses $m = O(\frac{n^2}{\epsilon^2} \log \frac{n}{\delta})$ examples, runs in* $\operatorname{poly}(n, 1/\epsilon, \log \frac{1}{\delta})$ *time, and outputs a hypothesis with error $O(\mathsf{opt}\sqrt{\log \frac{1}{\mathsf{opt}}} + \epsilon)$.*

This result thus trades off accuracy versus runtime compared with Theorem 1. We feel that Theorem 3 is intriguing, since it suggests that a deeper analysis might yield improved runtime bounds for Theorem 1 as well.

In section 6 we consider the problem of learning an unknown origin-centered halfspace under the uniform distribution on S^{n-1} in the presence of *malicious noise* (we give a precise definition of the malicious noise model in section 6). Recall from section 1.1 that we can view agnostic learning with respect to a particular marginal distribution $\mathcal{D}_X$ as the problem of learning under $\mathcal{D}_X$ in the presence of an adversary who may change the *labels* of an η fraction of the examples, without changing the actual distribution $\mathcal{D}_X$ over examples. In contrast, in the model of learning under *malicious noise* with respect to $\mathcal{D}_X$, roughly speaking the adversary is allowed to change an η fraction of the labels *and examples* given to the learner. As described in section 6 this is a very challenging noise model in which only limited positive results are known. We show that by combining the algorithm of Theorem 3 with a simple preprocessing step, we can achieve relatively high tolerance to malicious noise.

THEOREM 4. *There is a simple algorithm for learning origin-centered halfspaces under the uniform distribution on S^{n-1} to error ϵ in the presence of malicious noise when the noise rate η is at most $O(\frac{\epsilon}{n^{1/4} \log^{1/4}(n/\epsilon)})$. The algorithm runs in* $\operatorname{poly}(n, 1/\epsilon, \log \frac{1}{\delta})$ *time and uses $m = O(\frac{n^2}{\epsilon^2} \log \frac{n}{\delta})$ many examples.*

This is the highest known rate of malicious noise that can be tolerated in polynomial time for any nontrivial halfspace learning problem. The preprocessing step can be viewed as a somewhat counterintuitive form of outlier removal—instead of identifying and discarding examples that lie "too far" from the rest of the data set, we discard examples that lie too *close* to any other data point. The analysis of this approach relies on classical results from sphere packing.

2. Preliminaries. Let $\mathcal{D}$ be an arbitrary distribution on $X \times \{-1, 1\}$ for some set X. Let $\mathcal{C}$ be a class of Boolean functions on X. Define the *error of* $f : X \to \{-1, 1\}$ and the *optimal error of* $\mathcal{C}$ to be

$$\mathsf{err}(f) = \Pr_{(x,y) \leftarrow \mathcal{D}}[f(x) \neq y], \quad \mathsf{opt} = \min_{c \in \mathcal{C}} \mathsf{err}(c),$$

respectively. Roughly speaking, the goal in agnostic learning of a concept class $\mathcal{C}$ is as follows: given access to examples drawn from distribution $\mathcal{D}$, we wish to efficiently find a hypothesis with error not much larger than opt. More precisely, we say $\mathcal{C}$ is *agnostically learnable* if there exists an algorithm which takes as input δ and ϵ, has access to an example oracle $\mathsf{EX}(\mathcal{D})$, and outputs with probability greater than $1-\delta$ a *hypothesis* $h : X \rightarrow \{-1,1\}$ such that $\mathsf{err}(h) \leq \mathsf{opt} + \epsilon$. We say $\mathcal{C}$ is agnostically learnable in time t if its running time (including calls to the example oracle) is bounded by $t(\epsilon, \delta, n)$. If the above holds only for a distribution $\mathcal{D}$ whose margin is uniform over X, we say the algorithm *agnostically learns* $\mathcal{C}$ *over the uniform distribution.* See [25] for a detailed description of the agnostic learning framework.

A distribution is log-concave if its support is convex and it has a probability density function whose logarithm is a concave function from $\mathbb{R}^n$ to $\mathbb{R}$.

In all our algorithms we assume that we are given m labeled examples $\mathcal{Z} = (x^1, y^1), \ldots, (x^m, y^m)$ drawn independently from the distribution $\mathcal{D}$ over $X \times \{-1,1\}$. The sgn : $\mathbb{R} \rightarrow \{-1,1\}$ function is defined by $\mathrm{sgn}(z) = 1$ if $z \geq 0$ and $\mathrm{sgn}(z) = -1$ if $z < 0$.

2.1. Fourier preliminaries and the low-degree algorithm. For $S \subseteq [n]$, the parity function $\chi_S : \{-1,1\}^n \rightarrow \{-1,1\}$ over the variables in S is simply the multilinear monomial $\chi_S(x) = \prod_{i \in S} x_i$. The set of all 2^n parity functions $\{\chi_S\}_{S \subseteq [n]}$ forms an orthonormal basis for the vector space of real-valued functions on $\{-1,1\}^n$ with respect to the inner product $(f,g) = \mathbf{E}[f(x)g(x)]$ (which we write as $\mathbf{E}[fg]$; here and throughout section 2.1 unless otherwise indicated all probabilities and expectations are with respect to the uniform distribution over $\{-1,1\}^n$). Hence every real-valued function $f : \{-1,1\}^n \rightarrow \mathbb{R}$ can be uniquely expressed as a linear combination:

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S) \chi_S(x). \tag{1}$$

The coefficients $\hat{f}(S) = \mathbf{E}[f\chi_S]$ of the Fourier polynomial (1) are called the *Fourier coefficients* of f; collectively they constitute the *Fourier spectrum* of f. We recall *Parseval's identity*, which states that for every real-valued function f we have $\mathbf{E}[f(x)^2] = \sum_S \hat{f}(S)^2$. For Boolean functions, we thus have $\sum_S \hat{f}(S)^2 = 1$.

The "low-degree algorithm" for learning Boolean functions under the uniform distribution via their Fourier spectra was introduced by Linial, Mansur, and Nisan [31] and has proved to be a powerful tool in uniform distribution learning. The algorithm works by empirically estimating each coefficient $\hat{f}(S) \approx \tilde{f}(S) := \frac{1}{m} \sum_{j=1}^m f(x^j) \chi_S(x^j)$ with $|S| \leq d$ from the data and constructing the degree-d polynomial $p(x) = \sum_{|S| \leq d} \tilde{f}$ $(S)\chi_S(x)$ as an approximation to f. (Note that the polynomial $p(x)$ is real-valued rather than Boolean-valued. If a Boolean-valued classifier h is desired, it can be obtained by taking $h(x) = \mathrm{sgn}(p(x))$ and using the simple fact $\Pr_{\mathcal{D}}[\mathrm{sgn}(p(x)) \neq g(x)] \leq \mathbf{E}_{\mathcal{D}}[(p(x) - f(x))^2]$ which holds for any polynomial p, any Boolean function $f : \{-1,1\}^n \rightarrow \{-1,1\}$, and any distribution $\mathcal{D}$.)

Let $\alpha(\epsilon, n)$ be a function $\alpha : (0, 1/2) \times \mathsf{N} \rightarrow \mathsf{N}$. We say that concept class $\mathcal{C}$ has a *Fourier concentration bound* of $\alpha(\epsilon, n)$ if, for all $n \geq 1$, all $0 < \epsilon < \frac{1}{2}$, and all $f \in \mathcal{C}_n$, we have $\sum_{|S| \geq \alpha(\epsilon,n)} \hat{f}(S)^2 \leq \epsilon$. The low-degree algorithm is useful because it efficiently constructs a high-accuracy approximator for functions that have good Fourier concentration bounds (we suppress the logarithmic dependence on the failure probability δ to improve readability).

FACT 1 (see [31]). *Let $\mathcal{C}$ be a concept class with concentration bound $\alpha(\epsilon, n)$. Then for any $f \in \mathcal{C}$, given data labeled according to f and drawn from the uniform distribution on $X = \{-1,1\}^n$, the low-degree algorithm outputs, with probability* $1-\delta$, *a polynomial* p *such that* $\mathbf{E}[(p(x) - f(x))^2] \leq \epsilon$ *and runs in time* $\mathrm{poly}(n^{\alpha(\epsilon/2,n)}, \log \frac{1}{\delta})$.

The idea behind Fact 1 is simple: if the coefficients of p were precisely $\hat{f}(S)$ instead of $\tilde{f}(S)$, then the Fourier concentration bound and Parseval's identity would give $\sum_{|S|\geq\alpha(\epsilon/2,n)} = \mathbf{E}[(p(x)-f(x))^2] \leq \epsilon/2$. The extra $\epsilon/2$ is incurred because of approximation error in the estimates $\tilde{f}(S)$.

2.2. The low-degree algorithm and L_2 polynomial regression. The main observation of this section is that the low-degree Fourier algorithm of [31] can be viewed as a special case of least-squares polynomial regression over uniform distributions on the n-dimensional cube.

Let $\mathcal{D}$ be a distribution over $X \times \{-1,1\}$. In least-squares (L_2-norm) polynomial regression, one attempts to minimize the following:

$$\min_{p:\deg(p)\leq d} \mathbf{E}_{\mathcal{D}}\left[(p(x)-y)^2\right] \approx \min_{p:\deg(p)\leq d} \frac{1}{m}\sum_{j=1}^{m}\left(p(x^j)-y^j\right)^2. \tag{2}$$

Ideally, one would like to minimize the left-hand side (LHS), i.e., find the best degree-d polynomial L_2 approximation to y over $\mathcal{D}$. To do this (approximately) given a data set, we minimize the right-hand side (RHS). In particular, we write a polynomial as a sum over all degree $\leq d$ monomials, $p(x) = \sum_b p_b \prod_{i=1}^n (x_i)^{b_i}$, where the sum is over $\{b \in \mathbb{Z}^n | \sum_{i=1}^n b_i \leq d \text{ for all } i\ b_i \geq 0\}$. In turn, this can be viewed as a standard *linear* regression problem if we expand example x^j into a vector with a coordinate $\prod_{i=1}^n (x_i^j)^{b_i}$ for each of the $\leq n^{d+1}$ different b's. Least-squares linear regression, in turn, can be solved by a single matrix inversion; and thus in general we can approximate the RHS of (2) in $n^{O(d)}$ time.

Now let us consider L_2 polynomial regression in the uniform distribution scenario where $X = \{-1,1\}^n$, $y = f(x)$ for some function $f : X \to \{-1,1\}$, and we have a uniform distribution $\mathcal{U}_X$ over $x \in \{-1,1\}^n$. Since $x^2 = 1$ for $x \in \{-1,1\}$, we may consider only degree-d multilinear polynomials, i.e., sums of monomials $\chi_S(x) = \prod_{i\in S} x_i$ with $S \subseteq [n], |S| \leq d$. Using Parseval's identity, it is not difficult to show that best degree-d polynomial is exactly

$$\arg\min_{p:\deg(p)\leq d} \mathbf{E}_{\mathcal{U}_X}\left[(p(x)-f(x))^2\right] = \sum_{S\subseteq[n]:|S|\leq d} \hat{f}(S)\chi_S(x),$$

where $\hat{f}(S) = \mathbf{E}_{\mathcal{U}_X}[f(x)\chi_S(x)]$. Thus in this uniform case, one can simply estimate each coefficient $\hat{f}(S) \approx \frac{1}{m}\sum_{j=1}^m f(x^j)\chi_S(x^j)$ rather than solving the general least-squares regression problem; and this is precisely what the low-degree algorithm does.

In the *nonuniform* case, it is natural to consider running general L_2 polynomial regression rather than the low-degree algorithm. We do something similar to this in section 3, but first we consider the *agnostic* learning properties of the low-degree algorithm in the next subsection.

2.3. Using the low-degree algorithm as an agnostic learner. Kearns et al. prove the following statement about agnostic learning with the low-degree algorithm.

FACT 2 (see [25, Corollary 1]). *Let $\mathcal{C}$ be a concept class with concentration bound $\alpha(\epsilon, n)$. Then the low-degree algorithm agnostically learns $\mathcal{C}$ under the uniform distribution to error $\frac{1}{2} - (\frac{1}{2} - \mathsf{opt})^2 + \epsilon = \frac{1}{4} + \mathsf{opt}(1-\mathsf{opt}) + \epsilon$ with probability $1-\delta$ and in time* $\mathrm{poly}(n^{\alpha(\epsilon/2,n)}, \log\frac{1}{\delta})$.

This was termed a "weak agnostic learner" in [25] because as long as opt is bounded away from $1/2$, say $\mathsf{opt} = 1/2 - \gamma$, this resulting hypothesis has error at most $1/2-\gamma^2+\epsilon<1/2$. However, if opt is near 0, their bound is still $>1/4$. We now show that if opt is small, the low-degree algorithm can in fact achieve very low error.

OBSERVATION 3. *Let $\mathcal{C}$ be a concept class with concentration bound $\alpha(\epsilon,n)$. Then the low-degree algorithm agnostically learns $\mathcal{C}$ under the uniform distribution to error $8\mathsf{opt}+\epsilon$ in time $n^{O(\alpha(\epsilon/2,n))}$.*

Proof. Let $f\in C$ be an optimal function, i.e., $\Pr[y\neq f(x)]=\mathsf{opt}$. As described above, the low-degree algorithm (approximately) finds the best degree-d approximation $p(x)$ to the data y, i.e., $\min_{\deg(p)\le d}\mathbf{E}[(p(x)-y)^2]$, and the same term represents the mean squared error of p. This can be bounded using the "almost-triangle" inequality $(a-c)^2\le 2\left((a-b)^2+(b-c)^2\right)$ for $a,b,c\in\mathbb{R}$:

$$\begin{aligned}\min_{p:\deg(p)\le d}\mathbf{E}[(y-p(x))^2] &\le \mathbf{E}\left[\left(y-\sum\nolimits_{|S|<d}\hat{f}(S)\chi_S(x)\right)^2\right]\\ &\le 2\mathbf{E}\left[(y-f(x))^2+\left(f(x)-\sum\nolimits_{|S|<d}\hat{f}(S)\chi_S(x)\right)^2\right]\\ &= 2\left(4\Pr[y\neq f(x)]+\sum\nolimits_{|S|\ge d}\hat{f}(S)^2\right).\end{aligned}$$

The first term is $8\mathsf{opt}$ and the second is at most $\epsilon/2$ for $d=\alpha(n,\epsilon/2)$, where an additional $\epsilon/2$ is due to the sampling. Outputting $h(x)=\operatorname{sgn}(p(x))$ gives error at most $8\mathsf{opt}+\epsilon$ because $\Pr[\operatorname{sgn}(p(x))\neq y]\le\mathbf{E}[(p(x)-y)^2]$. □

Another way to state this is that if f and $\tilde{f}$ are two functions and f has a Fourier concentration bound of $\alpha(\epsilon,n)$, then $\tilde{f}$ satisfies the concentration bound $\sum_{|S|\ge\alpha(n,\epsilon)}\hat{\tilde{f}}(S)^2\le 8\Pr[f(x)\neq\tilde{f}(x)]+2\epsilon$.

3. $\mathbf{L_1}$ polynomial regression. Given the setup in the previous sections, it is natural to expect that we will now show that the general L_2 polynomial regression algorithm has agnostic learning properties similar to those established for the low-degree algorithm in Observation 3. However, such an approach yields only error bounds of the form $O(\mathsf{opt}+\epsilon)$, and for agnostic learning our real goal is a bound of the form $\mathsf{opt}+\epsilon$. To achieve this, we will instead use the L_1-*norm* rather than the L_2-norm.

Analogous to (2), in L_1-norm polynomial regression we attempt to minimize the following:

$$\min_{p:\deg(p)\le d}\mathbf{E}_{\mathcal{D}}\left[|p(x)-y|\right]\approx\min_{p:\deg(p)\le d}\frac{1}{m}\sum_{j=1}^{m}\left|p(x^j)-y^j\right|. \tag{3}$$

To solve the RHS minimization problem, again each example is expanded into a vector of length $\le n^{d+1}$, and an algorithm for L_1 linear regression is applied. L_1 linear regression is a well-studied problem, and the minimizing polynomial p for the RHS of (3) can be obtained in $\operatorname{poly}(n^d)$ time using linear programming (see Appendix A for an elaboration on this point). For our purposes, we will be satisfied with an approximate minimum, and hence one can use a variety of techniques for approximately solving linear programs efficiently.

How do L_1 and L_2 polynomial regression compare? In the noiseless case, both (2) and (3) approach 0 at related rates as d increases. However, in the noisy/agnostic case, flipping the sign of $y=\pm 1$ changes $(p(x)-y)^2$ by $4p(x)$, which can potentially be very large; in contrast, flipping y's sign can change $|p(x)-y|$ only by 2. On the

Input: $\boldsymbol{\mathcal{Z} = (x^1, y^1), \ldots, (x^m, y^m), d}$.

1. Find polynomial p of degree $\leq d$ to minimize the following: $\frac{1}{m}\sum_{j=1}^{m} |p(x^j) - y^j|$. (This can be done by expanding examples to include all monomials of degree $\leq d$ and then performing L_1 *linear* regression, as described earlier.)
2. Output $h(x) = \mathrm{sgn}\,(p(x) - t)$, where $t \in [-1, 1]$ is chosen so as to minimize the error of the hypothesis on $\mathcal{Z}$.

FIG. 1. *The L_1 polynomial regression algorithm.*

other hand, it is often easier to bound the L_1-error in terms of the mathematically convenient L_2-error. Thus while our polynomial regression algorithm works only with the L_1-norm, the performance bound and analysis depend on the L_2-norm.

3.1. The algorithm and proof of correctness. We now give the polynomial regression algorithm (see Figure 1) and establish conditions under which it is an agnostic learner achieving error $\mathsf{opt} + \epsilon$. The algorithm takes as input m examples, $\mathcal{Z} = (x^1, y^1), \ldots, (x^m, y^m)$, and a degree d.

THEOREM 5. *Suppose* $\min_{\deg(p) \leq d} E_{x \sim \mathcal{D}_X}[(p(x) - c(x))^2] \leq \epsilon^2$ *for some degree d, some distribution $\mathcal{D}$ over $X \times \{-1, 1\}$ with marginal $\mathcal{D}_X$, and any c in the concept class $\mathcal{C}$. Then for h output by the degree-d L_1 polynomial regression algorithm with* $m = \mathrm{poly}(n^d/\epsilon)$ *examples,* $\mathbf{E}_{\mathcal{Z} \sim \mathcal{D}^m}[\mathsf{err}(h)] \leq \mathsf{opt} + \epsilon$.

If we repeat the same algorithm $r = O(\log(1/\delta)/\epsilon)$ *times with fresh examples each and let h be the hypothesis with lowest error on an independent test set of size* $O(\log(1/\delta)/\epsilon^2)$*, then with probability* $1 - \delta$, $\mathsf{err}(h) \leq \mathsf{opt} + \epsilon$.

Remark 4. Note that using Theorem 5, a Fourier concentration bound of $\alpha(n, \epsilon)$ immediately implies that the L_1 regression algorithm achieves error $\mathsf{opt} + \epsilon$ in time $n^{O(\alpha(n, \epsilon^2))}$ for distributions $\mathcal{D}$ with marginal $\mathcal{D}_X$ that is uniform on $\{-1, 1\}^n$. As we will see in the next section, Theorem 5 can be applied to other distributions as well.

Proof of Theorem 5. Suppose the algorithm chooses polynomial p and threshold t. First, we claim that the empirical error of h on $\mathcal{Z}$ is at most one half the L_1-error of p:

$$\frac{1}{m}\sum_{j=1}^{m} \mathrm{I}(h(x^j) \neq y^j) \leq \frac{1}{2m}\sum_{j=1}^{m} \left|y^j - p(x^j)\right|. \tag{4}$$

To see this, note that $h(x^j) \neq y^j$ if and only if the threshold $t \in [-1, 1]$ lies in between the numbers $p(x^j)$ and y^j; i.e., if they are on the same side of t, then $\mathrm{sgn}(p(x^j) - t) = \mathrm{sgn}(y^j - t) = y^j$. Hence, even if we chose a *uniformly random* $t \in [-1, 1]$, for any j, the chance of t splitting these numbers is at most $\left|y^j - p(x^j)\right|/2$ because the width of $[-1, 1]$ is 2 and the separation between the numbers is $\left|y^j - p(x^j)\right|$. Thus, (4) holds in expectation for random $t \in [-1, 1]$. Since the algorithm chooses t to minimize the LHS of (4), it holds with certainty. (This reduction is a general procedure for converting an L_1 bound on error to a classification error, and a similar randomized threshold idea was used by Blum et al. [5] for the low-degree algorithm.)

Let c be an optimal classifier in $\mathcal{C}$, and let p^* be a polynomial of degree $\leq d$ with $\mathbf{E}_{\mathcal{D}}[(c(x) - p^*(x))^2] \leq \epsilon^2$. By the fact that $E[|Z|] \leq \sqrt{E[Z^2]}$ for any random variable Z, we have $\mathbf{E}_{\mathcal{D}}[|c(x) - p^*(x)|] \leq \epsilon$. By the algorithm's choice of p, we have

$$\frac{1}{m}\sum_{j=1}^{m} \left|y^j - p(x^j)\right| \leq \frac{1}{m}\sum_{j=1}^{m} \left|y^j - p^*(x^j)\right| \leq \frac{1}{m}\sum_{j=1}^{m} \left|y^j - c(x^j)\right| + \left|c(x^j) - p^*(x^j)\right|.$$

The expectation of the RHS is $\leq 2\mathsf{opt} + \epsilon$. Taking expectations and combining with (4) gives

$$\mathbf{E}_{\mathcal{Z}}\left[\frac{1}{m}\sum_{j=1}^{m} \mathrm{I}(h(x^j) \neq y^j)\right] \leq \mathsf{opt} + \frac{\epsilon}{2}.$$

By VC theory, for $m = \mathrm{poly}(n^d/\epsilon)$ examples, the empirical error $\frac{1}{m}\sum_{j=1}^{m} \mathrm{I}(h(x^j) \neq y^j)$ above and generalization error $\mathsf{err}(h)$ will differ by at most an expected $\epsilon/4$. Hence, the first part of the theorem is implied by

$$\mathbf{E}_{\mathcal{Z}}\left[\mathsf{err}(h)\right] \leq \mathsf{opt} + (3/4)\epsilon.$$

The second part of the theorem is a relatively standard reduction from expected error to high-probability guarantees. In particular, by Markov's inequality, on any single repetition,

$$\Pr_{\mathcal{Z}}\left[\mathsf{err}(h) \geq \mathsf{opt} + \left(\frac{7}{8}\right)\epsilon\right] \leq \frac{\mathsf{opt} + (3/4)\epsilon}{\mathsf{opt} + (7/8)\epsilon} \leq 1 - \frac{\epsilon}{16}.$$

Hence, after $r = O(\log(1/\delta)/\epsilon)$ repetitions of the algorithm, with probability $1 - \delta/2$, one of them will have $\mathsf{err}(h) \leq \mathsf{opt} + (7/8)\epsilon$. In this case, using an independent set of size $O(\log(1/\delta)/\epsilon^2)$, with probability at most $\delta/2$, we will choose one with error $> \mathsf{opt} + \epsilon$. $\square$

As noted at the very beginning of this section, an analogous L_2-algorithm could be defined to minimize $\frac{1}{m}\sum_{j=1}^{m}(p(x^j) - y^j)^2$ rather than $\frac{1}{m}\sum_{j=1}^{m}|p(x^j) - y^j|$. Error guarantees of the form $O(\mathsf{opt} + \epsilon)$ can be shown for this L_2-algorithm, following the same argument but again using the "almost-triangle" inequality.

3.2. Relationship to SVMs. As pointed out by Avrim Blum, our algorithm is very similar to an SVM with a polynomial kernel and can be made even more similar. The standard SVM with a degree-d *polynomial kernel* solves the following minimization problem:

$$\min_{\deg(p) \leq d} (1 - \lambda)\frac{1}{m}\sum_{i=1}^{m} L(y^i, z) + \lambda(\text{regularization term}),$$

where $L(y^i, z) = \max\{0, 1 - y^i z\}$. It does this using an algorithmic trick that requires time only $\mathrm{poly}(m, n, d)$. In theory, this could be substantially faster than our $n^{O(d)}$ algorithm. However, for our analysis, we require $m = n^{O(d)}$ samples, in which case the SVM algorithm is no faster.

Step 1 of our algorithm could be replaced by the above minimization problem, with $\lambda = 0$, and the analysis would hold almost exactly as is. Intuitively, this is because, for $|y| = 1$, $L(y, z) = |y - z|$ unless $yz > 1$. However, if $yz > 1$, thresholding z with $t \in [-1, 1]$ will certainly give us the correct prediction of this y. More technically, we have that, for $|y| = 1$, $L(y, z) \leq |y - z|$, yet we still have that $\Pr_{t \in [-1,1]}[y \neq \mathrm{sgn}(z - t)] \leq \frac{1}{2}L(y, z)$ (we now have $L(y, z)$ where we had $|y - z|$).

Hence one can use a standard SVM package to implement our algorithm, setting the regularization parameter to 0. The only nonstandard part would be choosing an optimal threshold t rather than using standard SVM choice of $t = 0$.

4. Agnostic learning halfspaces and disjunctions via polynomial regression. In this section we show how to apply Theorem 5 to prove Theorems 1 and 2.

As noted in Remark 4, Theorem 5 implies that any concept class with a Fourier concentration bound is in fact agnostically learnable to error $\mathsf{opt}+\epsilon$ under the uniform distribution on $\{-1,1\}^n$. In particular, Theorem 1, part 1(a), follows immediately from the Fourier concentration bound for halfspaces of [26].

Fact 5 (see [26]). *The concept class $\mathcal{C}$ of all halfspaces over $\{-1,1\}^n$ has a Fourier concentration bound of $\alpha(\epsilon, n) = 441/\epsilon^2$.*

For the uniform distribution on S^{n-1} and any log-concave distribution, we can prove the existence of a good low-degree polynomial as follows. Suppose we had a good degree-d univariate approximation to the sign function $p_d(x) \approx \operatorname{sgn}(x)$, and say we have an n-dimensional halfspace $\operatorname{sgn}(v \cdot x - \theta)$. Then $\operatorname{sgn}(v \cdot x - \theta) \approx p_d(v \cdot x - \theta)$. Moreover, this latter quantity is now a degree-d multivariate polynomial. The sense in which we measure approximations will be distributional, the L_2 error of our multivariate polynomial over the distribution $\mathcal{D}$. Hence, we need a polynomial p_d that well-approximates the sign function on the marginal distribution in the direction v, i.e., the distribution over projections onto the vector v.

For the uniform distribution on a sphere, the projection onto a single coordinate is distributed very close to Gaussian distribution. For a log-concave distribution, its projection is distributed log-concavely. In both of these cases, it so happens that the necessary degree to get approximation error ϵ boils down to a one-dimensional problem! For the sphere, we can upper bound the degree necessary as a function of ϵ using the following for the normal distribution $N(0, \frac{1}{\sqrt{2}})$ with density $e^{-x^2}/\sqrt{\pi}$.

Theorem 6. *For any $d > 0$ and any $\theta \in \mathbb{R}$, there is a degree-d univariate polynomial $p_{d,\theta}$ such that*

$$\int_{-\infty}^{\infty} (p_{d,\theta}(x) - \operatorname{sgn}(x-\theta))^2 \frac{e^{-x^2}}{\sqrt{\pi}} dx = O\left(\frac{1}{\sqrt{d}}\right). \tag{5}$$

We note that the $n^{O(1/\epsilon^2)}$-time, $O(\mathsf{opt}+\epsilon)$-error analogues of Theorem 1, part 1, mentioned in section 1.4 follow from Fact 5 and Theorem 6 using the L_2-norm analogue of the polynomial regression algorithm mentioned at the end of section 3. The improved time bound comes from the fact that we no longer need to invoke $\mathbf{E}[|Z|] \le \sqrt{\mathbf{E}[Z^2]}$ to bound the square loss, since we are minimizing the square loss directly rather than the absolute loss. We defer the proof of Theorem 6 to Appendix B.

Using Theorem 6, it is not difficult to establish Theorem 1, part 1(b), which we restate below:

> Let $\mathcal{D}$ be a distribution over $\mathbb{R}^n \times \{-1,1\}$ with $\mathcal{D}_X$ uniform over S^{n-1}. With probability $1-\delta$, the L_1 polynomial regression outputs a hypothesis with error $\mathsf{opt}+\epsilon$ given $\operatorname{poly}(n^{1/\epsilon^4}, \log\frac{1}{\delta})$ examples.

Proof. Let $f(x) = \operatorname{sgn}(v \cdot x - \tau)$ be any halfspace over the unit ball S^{n-1}, where without loss of generality we may assume $\|v\| = 1$ (and thus $|\tau| \le 1$). Let $\mathcal{U}$ denote the uniform distribution over S^{n-1}. It suffices to establish the existence of a degree-d polynomial $P(x)$, with $d = O(1/\epsilon^4)$, which satisfies the condition $\mathbf{E}_{x\in\mathcal{U}}[(P(x) - f(x))^2] \le \epsilon^2$; given such a polynomial, we apply Theorem 5, and Theorem 1, part 1(b), immediately follows.

Let $\theta = \sqrt{\frac{n-3}{2}}\tau$, and let $P(x) = p_{d,\theta}(\sqrt{\frac{n-3}{2}} v \cdot x)$. For $d = O(1/\epsilon^4)$, we show that the polynomial $P(x) = p_{d,\theta}\big(\sqrt{\frac{n-3}{2}}(v \cdot x)\big)$ satisfies $\mathbf{E}_{\mathcal{U}}[(P(x) - f(x))^2] \le \epsilon^2$.

We have (justifications are given below)

$$
\begin{aligned}
&\mathbf{E}_{x\in\mathcal{U}}\ [(P(x)-f(x))^2]\\
&\quad = \mathbf{E}_{x\in\mathcal{U}}\left[\left(p_{d,\theta}\left(\sqrt{\tfrac{n-3}{2}}(v\cdot x)\right)-\operatorname{sgn}\left(\sqrt{\tfrac{n-3}{2}}(v\cdot x)-\theta\right)\right)^2\right]\\
&\quad = \frac{A_{n-2}}{A_{n-1}}\int_{-1}^{1}(1-z^2)^{(n-3)/2}\left(p_{d,\theta}\left(\sqrt{\tfrac{n-3}{2}}z\right)-\operatorname{sgn}\left(\sqrt{\tfrac{n-3}{2}}z-\theta\right)\right)^2 dz \qquad (6)\\
&\quad \le \frac{A_{n-2}}{A_{n-1}}\int_{-\infty}^{\infty}e^{-z^2(n-3)/2}\left(p_{d,\theta}\left(\sqrt{\tfrac{n-3}{2}}z\right)-\operatorname{sgn}\left(\sqrt{\tfrac{n-3}{2}}z-\theta\right)\right)^2 dz \qquad (7)\\
&\quad = \frac{A_{n-2}}{A_{n-1}}\int_{-\infty}^{\infty}e^{-y^2}\left(p_{d,\theta}(y)-\operatorname{sgn}(y-\theta)\right)^2\frac{dy}{\sqrt{(n-3)/2}} \qquad (8)\\
&\quad \le \epsilon^2, \qquad (9)
\end{aligned}
$$

where (6) follows from Fact 10 on the PDF of the uniform distribution over S^{n-1}; (7) follows from $1-z\le\exp(-z)$ and the fact that the integrand is nonnegative; (8) follows from a change of variable $y=\sqrt{\frac{n-3}{2}}\cdot z$; and (9) follows from $\frac{A_{n-2}}{A_{n-1}}=\Theta(\sqrt{n})$, Theorem 6, and our choice of $d=O(1/\epsilon^4)$. This concludes the proof of Theorem 1, part 1(b). □

Since we have proven Theorem 1, part 1(a), in section 4, we are now ready to move on to the log-concave part. The first thing to notice is that, just as the normal distribution served as a prototypical distribution for all spheres, there is a log-concave distribution that is not much smaller than any other.

Lemma 6. *Let ν be the distribution on $\mathbb{R}$ with density $d\nu(x)=e^{-|x|/16}/32$. Let μ be any log-concave distribution on $\mathbb{R}$ with mean 0 and variance 1. Then, for all $x\in\mathbb{R}$, $d\mu(x)\le(32e)d\nu(x)$.*

In the above, we necessarily chose a distribution ν that did not have variance 1.

Proof. To prove this lemma, we will use the properties of log-concave functions given by Lovász and Vempala [34]. Specifically, for any log-concave density $d\mu$ with mean 0 and variance 1, for all x $d\mu(x)\le 1$, and $d\mu(0)\ge 1/8$. From the latter fact, we next argue that $d\mu(x)\le e^{-|x|/16}$ for $|x|>16$. It suffices to show this for $x>16$ by symmetry. Suppose not; i.e., suppose there exists $r>16$ such that $d\mu(r)\ge e^{-r/16}$. Then log-concavity implies that $d\mu(x)\ge(1/8)^{1-x/r}(e^{-r/16})^{x/r}$ for $x\in[0,r]$. In turn, this means

$$\int_0^{16}d\mu(x)\ge\int_0^{16}\frac{1}{8}e^{-x/16}dx>1,$$

which is a contradiction. Hence, $d\mu(x)\le e^{-|x|/16}=32d\nu(x)$ for $|x|>16$. (These bounds are far from tight.) Also, for $|x|<16$, $d\mu(x)\le 1\le(32e)d\nu(x)$. □

This lemma will enable us to transfer a bound on the error of a *fixed* log-concave function such as $e^{-2|x|}$ to *all* log-concave functions.

Lemma 7. *There exists a fixed function $d:\mathbb{R}\to\mathbb{R}$, such that, for any log-concave distribution μ, and any $\theta\in\mathbb{R}$, there exists a degree-$d(\epsilon)$ polynomial p, such that*

$$\int_{-\infty}^{\infty}(p(x)-\operatorname{sgn}(x-\theta))^2d\mu(x)\le\epsilon.$$

Proof. It suffices to show it for any log-concave distribution μ with mean 0 and variance 1. This is because we can always apply an affine transformation to x,

$x \to ax + b$, which puts it in such a standard position and maintains the properties of the lemma (for a suitably transformed polynomial p and θ). Thus, we assume that μ has mean 0 and variance 1.

Next, we claim it suffices to show the lemma for the log-concave density $d\nu(x) = e^{-|x|/16}/32$, which has mean 0 but variance > 1. To see this, suppose it holds for $d\nu$ and p, and we have some mean 0 variance 1 log-concave density $d\mu$. Then by Lemma 6,

$$\int_{-\infty}^{\infty} (p(x) - \operatorname{sgn}(x-\theta))^2 d\mu(x) \leq 32e \int_{-\infty}^{\infty} (p(x) - \operatorname{sgn}(x-\theta))^2 d\nu(x) \leq 32e\epsilon.$$

Hence it would hold for mean-0 variance-1 $d\mu$ with function $d' : \mathbb{R} \to \mathbb{R}$, where $d'(\epsilon) = d(\epsilon/(32e))$. By a similar stretching argument, it suffices to show it for $d\nu(x) = e^{-2|x|}$.

Next, again without loss of generality, it suffices to show it for $|\theta| < \log 1/\epsilon$. For if $|\theta| > 2\log 1/\epsilon$, then the constant polynomial $p(x) = -\operatorname{sgn}(\theta)$ has error less than ϵ under $d\nu(x) = e^{-2|x|}$. Continuing on the seemingly endless chain of without losses of generalities, next it suffices to show it for $\theta = 0$. Suppose it holds for $d\nu(x) = e^{-2|x|}$, a particular p and ϵ, and $\operatorname{sgn}(x)$. That is,

$$\int_{-\infty}^{\infty} (p(x) - \operatorname{sgn}(x))^2 d\nu(x) \leq \epsilon. \tag{10}$$

Then consider the function $\operatorname{sgn}(x-\theta)$ and the density $d\rho(x) = e^{-2|x-\theta|/\log(1/\epsilon)}/\log(1/\epsilon)$. For this density, by (10) and change of variable $z = \log(1/\epsilon)(x-\theta)$,

$$\int_{-\infty}^{\infty} (p(z) - \operatorname{sgn}(z))^2 d\nu(z) = \int_{-\infty}^{\infty} (p(\log(1/\epsilon)(x-\theta)) - \operatorname{sgn}(x-\theta))^2 d\rho(x) \leq \epsilon. \tag{11}$$

Now observe that as long as $\log(1/\epsilon) > 1$ (where $\epsilon \leq 1/e$),

$$\frac{d\nu(x)}{d\rho(x)} = \log\left(\frac{1}{\epsilon}\right) e^{2(\frac{|x-\theta|}{\log(1/\epsilon)} - |x|)} \leq \log\left(\frac{1}{\epsilon}\right) e^{2(\frac{|x-\theta|-|x|}{\log(1/\epsilon)})} \leq \log\left(\frac{1}{\epsilon}\right) e^{2\frac{|\theta|}{\log(1/\epsilon)}} \leq \log\left(\frac{1}{\epsilon}\right) e^2.$$

By this and (11),

$$\int_{-\infty}^{\infty} (p(\log(1/\epsilon)(x-\theta)) - \operatorname{sgn}(x-\theta))^2 d\mu(x) \leq e^2 \epsilon \log(1/\epsilon).$$

Hence a bound of ϵ on the error of p for $\operatorname{sgn}(x)$ implies a bound of $e^2\epsilon\log(1/\epsilon)$ on the error of $p(\log(1/\epsilon)(x-\theta))$. So, it suffices to show we can achieve such a bound for $\operatorname{sgn}(x)$, $d\nu(x) = e^{-2|x|}$, and arbitrarily small ϵ.

At this point, we have a single function $\operatorname{sgn}(x)$ and a single density $e^{-2|x|}$, and we must establish that for any ϵ there is some $d = d(\epsilon)$ for which there is a degree-d polynomial p for which (10) holds. But $\operatorname{sgn}(x) \in L^2(\mathbb{R}, e^{-2|x|})$ because $\int_{-\infty}^{\infty} \operatorname{sgn}(x)^2 e^{-2|x|} dx = 1 < \infty$ and it is known that polynomials are dense in $L^2(\mathbb{R}, e^{-2|x|})$ [42]. $\square$

4.1. Agnostically learning disjunctions under any distribution. We can use the polynomial regression algorithm to learn disjunctions agnostically with respect to any distribution in subexponential time. We make use of the existence of low-degree polynomials that strongly approximate the OR function in the L_∞-norm.

THEOREM 7 (see [38, 36, 26]). *Let $f(x_1, \ldots, x_n)$ compute the OR function on some subset of (possibly negated) input variables. Then there exists a polynomial p of degree $O(\sqrt{n}\log(1/\epsilon))$ such that, for all $x \in \{-1,1\}^n$, we have $|f(x) - p(x)| \le \epsilon$.*

For $\epsilon = \Theta(1)$, this fact appears in [38, 36]; an easy extension to arbitrary ϵ is given in [26]. Theorem 2 follows immediately from Theorems 7 and 5, since for any distribution $\mathcal{D}$ the L_∞ bound given by Theorem 7 clearly implies the bound on expectation required by Theorem 5.

We note that low-degree L_∞-approximators are know for richer concept classes than just disjunctions. For example, results of O'Donnell and Servedio [37] show that any Boolean function $f : \{-1,1\}^n \to \{-1,1\}$ computed by a Boolean formula of linear size and constant depth is ϵ-approximated in the L_∞-norm by a polynomial of degree $\tilde{O}(\sqrt{n}) \cdot \mathrm{poly}\log\frac{1}{\epsilon}$. By combining Theorem 5 with such existence results, one can immediately obtain arbitrary-distribution agnostic learning results analogous to Theorem 2 for concept classes of such formulas as well; one well-studied example of such a concept class is the class of *read-k DNF formulas* for constant k.

4.2. Hardness results for agnostically learning halfspaces over the hypercube. In this section we show that the challenging "learning noisy parity" problem reduces to the problem of agnostically learning halfspaces with respect to the uniform distribution over the hypercube. Recall that a vector $c \in \{0,1\}^n$ induces a parity function $c : \{0,1\}^n \to \{0,1\}$ as follows: $c(x) = c{\cdot}x \bmod 2$ (the indices of c equal to 1 are the *relevant* variables). The *noisy parity learning problem* is the problem of PAC learning an unknown parity function with respect to the uniform distribution on $\{0,1\}^n$ where the *label* of each example is flipped (independently) with probability η. The fastest known learning algorithm for this well-known problem is due to Blum, Kalai, and Wasserman [6] and runs in time $2^{O(n/\log n)}$.

An algorithm for agnostically learning halfspaces can be easily transformed into an algorithm for learning parity with noise.

THEOREM 8. *Let A be an algorithm for agnostically learning halfspaces to accuracy $\mathsf{opt} + \epsilon$ with respect to the uniform distribution over $\{0,1\}^n$ running in time $t = t(1/\epsilon, n)$. Then there exists an algorithm B for learning parity with noise which runs in time* $\mathrm{poly}(n,t)$.

Proof. Assume that the unknown parity function c has k relevant variables (and for simplicity assume k is even). Note that for a set S of k variables, the majority function on S (equal to 1 if $k/2+1$ or more of the variables in S are set to 1) agrees with the parity function on all variables in S for a $1/2+\Theta(1/\sqrt{k})$ fraction of inputs of $\{0,1\}^n$. This is because the majority function equals parity for all inputs of hamming weight equal to $k/2$ (which have mass $\Theta(1/\sqrt{k})$) and agrees with parity on half of all other inputs.

Now choose a random example (labeled by c) and flip its label with probability η. The probability that the majority function on S correctly labels the example equals $\eta + (1-2\eta)(1/2 - \Theta(1/\sqrt{k})) = 1/2 - (1-2\eta)\Theta(1/\sqrt{k})$. That is, the *error rate* of the majority function on S with respect to noisy examples is bounded away from 1/2 by $(1-2\eta)\Theta(1/\sqrt{k})$.

We can now use an algorithm for agnostically learning halfspaces to identify the relevant variables of the unknown parity function c. To determine if the variable x_i is relevant, set $\epsilon = (1/2)(1-2\eta)/\sqrt{k}$ and take a number of random examples as specified by the agnostic learner. Feed the examples to the agnostic learner *with the ith bit removed from every example.* If x_i is a relevant variable, then the labels will be totally uncorrelated with the examples (now of length $n-1$), and the agnostic learner will not produce a hypothesis with error rate bounded away from 1/2. If x_i is irrelevant, then the majority function on the relevant variables has error rate bounded away from 1/2, and the agnostic learner will output a hypothesis with error less than $1/2 - (1/2)(1 - 2\eta)/\sqrt{k}$. □

If the error rate η is $\Theta(1)$ and the agnostic learning algorithm runs in time $n^{O(1/\epsilon^{2-\beta})}$, then the above algorithm will learn a noisy parity in time $2^{O(n^{\gamma})}$ for some $0 < \gamma < 1$.

4.3. An application to learning intersections of halfspaces. Learning an intersection of halfspaces is a challenging and well-studied problem even in the noise-free setting. Klivans, O'Donnell, and Servedio [26] showed that the standard low-degree algorithm can learn the intersection of k halfspaces with respect to the uniform distribution on $\{-1, 1\}^n$ to error ϵ in time $n^{O(k^2/\epsilon^2)}$, provided that $\epsilon < 1/k^2$. Note that because of the requirement on ϵ, the algorithm always takes time at least $n^{\Omega(k^6)}$, even if the desired final error is $\epsilon = \Theta(1)$ independent of k.

We can use the idea of learning halfspaces agnostically to obtain the following runtime bound, which is better than [26] for $\epsilon > \frac{1}{k}$.

THEOREM 9. *Let $f = h_1 \wedge \cdots \wedge h_k$ be an intersection of k halfspaces over $\{-1, 1\}^n$. Then f is learnable with respect to the uniform distribution over $\{-1, 1\}^n$ in time $n^{O(k^4/\epsilon^2)}$ for any $\epsilon > 0$.*

We note that a comparable bound can be proved via techniques from a recent work due to Jackson, Klivans, and Servedio [22] which does not involve agnostic learning. The presentation here, however, is more straightforward and shows how agnostic learning can have applications even in the nonnoisy framework.

The approach that establishes Theorem 9 is similar to Jackson's Harmonic Sieve [21]: we apply a boosting algorithm, using the polynomial regression algorithm at each stage to identify a low-degree polynomial which, after thresholding, has advantage at least $\Omega(1/k)$ on the target function.

We begin with the following easy fact which follows directly from the "discriminator lemma" [19].

FACT 8. *Let $f = h_1 \wedge \cdots \wedge h_k$ be an intersection of k halfspaces. Then for any distribution $\mathcal{D}$ on $\{0, 1\}^n$ either there exists an h_i such that $|E_{\mathcal{D}}[fh_i]| \geq 1/k$ or we have $|E_{\mathcal{D}}[f]| \geq 1/k$.*

Hence for any distribution $\mathcal{D}$ there exists a single halfspace which has accuracy at least $1/2+1/2k$ with respect to f and $\mathcal{D}$. We will be concerned only with distributions that are c-bounded (c will be chosen later), i.e., distributions D such that $D(x) \leq c/2^n$ for all x. Fix such a c-bounded distribution $\mathcal{D}$, and let $h_{\mathcal{D}}$ denote the halfspace obtained from Fact 8. Applying Fact 5, it is not difficult to see that for any halfspace (and in particular $h_{\mathcal{D}}$) and sufficiently large constant a,

$$\sum_{S, |S| \geq a \cdot k^4 c^2} \hat{h_{\mathcal{D}}}(S)^2 \leq 1/16ck^2.$$

By setting $g = \sum_{S, |S| \leq a \cdot k^4 c^2} \hat{h_{\mathcal{D}}}(S) \chi_S(x)$, we have $E_{\mathcal{D}}[|g - h_D|] \leq 1/4k$ for any c-bounded distribution $\mathcal{D}$.

We now show that the polynomial regression algorithm can be used as a weak learning algorithm for f.

LEMMA 9. *There exists an algorithm A such that for any c-bounded distribution $\mathcal{D}$ and $0 < \delta < 1$, if A is given access to examples drawn from $\mathcal{D}$ labeled according to*

f, then A runs in time poly$(n^{k^4c^2}, 1/\delta)$ and, with probability at least $1-\delta$, A outputs a hypothesis h such that $Pr_{\mathcal{D}}[f(x) = h(x)] \geq 1/2 + 1/8k$.

Proof. Let $\ell = ak^4c^2$ for a sufficiently large constant a. Apply the polynomial regression algorithm from section 3 to obtain a hypothesis $g^* = \mathrm{sgn}\big(\sum_{|S|\leq\ell} w_S\chi_S(x) - t\big)$. For $\xi > 0$, we claim that g^* has error less than $1/2 - 1/4k + \xi$ as long as $m \geq \mathrm{poly}(n^\ell, 1/\xi^2, \log(1/\delta))$ as in Theorem 5. To see this, note that

$$E_{\mathcal{D}}[|f(x) - g^*|] \leq E_{\mathcal{D}}[|f(x) - h_{\mathcal{D}}(x)|] + E_{\mathcal{D}}[|h_{\mathcal{D}}(x) - g^*|]$$

and recall that the first term on the RHS is at most $1/2 - 1/2k$. For the second term, recall that $\min_w E_{\mathcal{D}}[|h_{\mathcal{D}}(x) - \sum_{|S|\leq\ell} w_S\chi_S|] \leq 1/4k$. But g^* is an approximation to the truncated Fourier polynomial for $h_{\mathcal{D}}(x)$, and as in the proof of Theorem 5, for our choice of m, $E_{\mathcal{D}}[|h_{\mathcal{D}}(x) - g^*(x)|] \leq \min_w E_{\mathcal{D}}[|h_{\mathcal{D}}(x) - \sum_{|S|\leq\ell} w_S\chi_S|] + \xi$ with probability greater than $1-\delta$. Hence with probability $1-\delta$ we have $E_{\mathcal{D}}[|f(x) - g^*(x)|] \leq 1/2 - 1/4k + \xi$. Taking $\xi = 1/(8k)$ gives the lemma. □

At this point, we will need to recall the definition of a *boosting* algorithm; see, e.g., [16]. Roughly speaking, a boosting algorithm iteratively applies a weak learning algorithm as a subroutine in order to construct a highly accurate final hypothesis. At each iteration, the boosting algorithm generates a distribution $\mathcal{D}$ and runs the weak learner to obtain a hypothesis which has accuracy $1/2 + \gamma$ with respect to $\mathcal{D}$. After $t = \mathrm{poly}(1/\gamma, 1/\epsilon)$ iterations, the boosting algorithm outputs a hypothesis with accuracy greater than $1-\epsilon$. The following fact from [27] is sufficient for our purposes.

THEOREM 10. *There is a boosting algorithm which runs in $t = O(1/\epsilon^2\gamma^2)$ iterations and at each stage generates an $O(1/\epsilon)$-bounded distribution $\mathcal{D}$.*

By combining this boosting algorithm with the weak learning algorithm from Lemma 9, we obtain Theorem 9.

Proof of Theorem 9. Run the boosting algorithm to learn f using the weak learner from Lemma 9 as a subroutine. The boosting algorithm requires at most $O(1/\epsilon^2k^2)$ iterations, since the distributions are all $O(1/\epsilon)$ bounded and the weak learner outputs a hypothesis with accuracy $1/2 + \Omega(1/k)$. The running time of the weak learning algorithm is at most $n^{O(k^4/\epsilon^2)}$, since each distribution is $c = O(1/\epsilon)$ bounded. □

5. Learning halfspaces over the sphere with the degree-1 version of the polynomial regression algorithm.

Let us return to the case where the marginal distribution $\mathcal{D}_X$ is uniform over S^{n-1}, and now consider the homogeneous $d = 1$ version of the polynomial regression algorithm. In this case, we would like to find the vector $w \in \mathbb{R}^n$ that minimizes $\mathbf{E}_{\mathcal{D}_X}[(w \cdot x - y)^2]$. By differentiating with respect to w_i and using the fact that $\mathbf{E}[x_i] = \mathbf{E}[x_ix_j] = 0$ for $i \neq j$ and $\mathbf{E}[x_i^2] = \frac{1}{n}$, we see that the minimum is achieved at $w_i = \frac{1}{n}E[x_iy_i]$.

This is essentially the same as the simple `Average` algorithm which was proposed by Servedio in [40] for learning origin-centered halfspaces under uniform in the presence of random misclassification noise. The `Average` algorithm draws examples until it has a sample of m positively labeled examples $x^1, \ldots, x^m$, and then it returns the hypothesis $h(x) = \mathrm{sgn}(\overline{v} \cdot x)$, where $\overline{v} = \frac{1}{m}\sum_{i=1}^m x^i$ is the vector average of the positive examples. The intuition for this algorithm is simple: if there were no noise, then the average of the positive examples should (in the limit) point exactly in the direction of the target normal vector.

A straightforward application of the bounds from sections 3 and 4 implies only that the degree-1 polynomial regression algorithm should achieve some fixed constant accuracy $\Theta(1)$ independent of `opt` for agnostic learning halfspaces under the uniform distribution on S^{n-1}. However, a more detailed analysis shows that the simple `Average` algorithm does surprisingly well, in fact obtaining a hypothesis with error rate $O(\mathsf{opt}\sqrt{\log(1/\mathsf{opt})}) + \epsilon$; this is Theorem 3. We give useful preliminaries in section 5.1 and prove Theorem 3 in section 5.2.

5.1. Learning halfspaces on the unit sphere: Preliminaries.

We write $\mathcal{U}$ to denote the uniform distribution over $S^{n-1} = \{x \in \mathbb{R}^n \mid \|x\| = 1\}$. Given two nonzero vectors $u, v \in \mathbb{R}^n$, we write $\alpha(u, v)$ to denote $\arccos(\frac{u \cdot v}{\|u\| \cdot \|v\|})$, the angle between u and v. If the target halfspace is $\operatorname{sgn}(u \cdot x)$ and $\operatorname{sgn}(v \cdot x)$ is a hypothesis halfspace, then it is easy to see that we have $\Pr_{x \in \mathcal{U}}[\operatorname{sgn}(u \cdot x) \neq \operatorname{sgn}(v \cdot x)] = \alpha(u, v)/\pi$.

We write A_{n-1} to denote the surface area of S^{n-1}. It is well known (see, e.g., [1]) that $A_{n-2}/A_{n-1} = \Theta(n^{1/2})$. The following fact (see, e.g., [1]) is useful.

Fact 10. *For any unit vector $v \in \mathbb{R}^n$ and any $-1 \leq \alpha < \beta \leq 1$, we have*

$$\Pr_{x \in \mathcal{U}}[\alpha \leq v \cdot x \leq \beta] = \frac{A_{n-2}}{A_{n-1}} \cdot \int_{\alpha}^{\beta} (1 - z^2)^{(n-3)/2} dz.$$

The following straightforward result lets us deal easily with sample error.

Fact 11. *Let $\mathcal{D}$ be any distribution over S^{n-1}. Let v denote the expected location $\mathbf{E}_{x \in \mathcal{D}}[x]$ of a random draw from $\mathcal{D}$, and suppose that $\|v\| \geq \xi$. Then if $\overline{v} = \frac{1}{m}\sum_{i=1}^{m} x^i$ is a sample estimate of $\mathbf{E}_{x \in \mathcal{D}}[x]$, where each x^i is drawn independently from $\mathcal{D}$ and $m = O(\frac{n}{\epsilon^2 \xi^2} \log \frac{n}{\delta})$, we have that $\Pr_{x \in \mathcal{U}}[sgn(\overline{v} \cdot x) \neq \operatorname{sgn}(v \cdot x)] \leq \epsilon$ with probability at least $1 - \delta$.*

Proof. We define an orthonormal basis for $\mathbb{R}^n$ by letting vector u^1 denote $\frac{v}{\|v\|}$ and letting $u^2, \ldots, u^n$ be an arbitrary orthonormal completion. Given a vector $z \in \mathbb{R}^n$, we may write z_1 for $z \cdot u^1$ and $z_2, \ldots, z_n$ for $z \cdot u^2, \ldots, z \cdot u^n$, respectively. We have $\mathbf{E}_{x \in \mathcal{D}}[x_1] = \xi$ so that standard additive Chernoff bounds imply that taking $m = O(\frac{1}{\xi^2} \log \frac{1}{\delta})$ many draws will result in $|\overline{v}_1 - \xi| \leq \frac{\xi}{2}$ with probability at least $1 - \frac{\delta}{2}$. For $i = 2, \ldots, n$, we have $\mathbf{E}_{x \in \mathcal{D}}[x_i] = 0$; again standard additive Chernoff bounds imply that taking $m = O(\frac{n}{\epsilon^2 \xi^2} \log \frac{n}{\delta})$ many draws will result in $|\overline{v}_i| \leq \frac{\epsilon \xi}{2\sqrt{n}}$ for each i with probability at least $1 - \frac{\delta}{2}$. Thus, with overall probability at least $1 - \delta$ we have

$$\alpha(\overline{v}, v) = \arctan\left(\frac{\sqrt{\overline{v}_2^2 + \cdots + \overline{v}_n^2}}{\overline{v}_1}\right) \leq \arctan(\epsilon) \leq \epsilon,$$

and thus $\Pr_{x \in \mathcal{U}}[sgn(\overline{v} \cdot x) \neq \operatorname{sgn}(v \cdot x)] \leq \alpha(\overline{v}, v)/\pi < \epsilon/\pi < \epsilon$. □

5.2. Proof of Theorem 3.

We have that $\mathcal{D}$ is a distribution over $X \times \{-1, 1\}$ whose marginal is the uniform distribution $\mathcal{U}$ on S^{n-1}. Without loss of generality we may suppose that the optimal origin-centered halfspace is $f(x) = \operatorname{sgn}(x_1)$; i.e., the normal vector to the separating hyperplane is $e_1 = (1, 0, \ldots, 0)$. We write S^+ to denote the "positive hemisphere" $\{x \in S^{n-1} : x_1 \geq 0\}$ and write S^- to denote $S^{n-1} \setminus S^+$. We may also suppose without loss of generality that the optimal halfspace's error rate opt is such that $O(\mathsf{opt}\sqrt{\log \frac{1}{\mathsf{opt}}})$ is less than $\frac{1}{4}$; i.e., opt is less than some fixed absolute constant that we do not specify here.

Let $p : S^{n-1} \to [0, 1]$ be the function

$$p(z) = \Pr_{(x,y) \in \mathcal{D}}[y \neq f(z) \mid x = z] \tag{12}$$

so intuitively $p(z)$ is the probability of getting a "noisy label" y on instance z. (We assume the joint distribution $\mathcal{D}$ on $X \times Y$ is sufficiently "nice" in terms of measurability, etc., so that p is well defined as specified above.) Let v denote the true vector average of all positively labeled examples generated by $\mathcal{D}$, i.e.,

$$v = \int_{x \in S^+} x(1 - p(x))\mathcal{U}(x) \ + \int_{x \in S^-} xp(x)\mathcal{U}(x).$$

If the number m of examples used by Average went to infinity, the vector average $\overline{v}$ that Average computes would converge to v. We prove Theorem 3 by first establishing bounds on v and then using Fact 11 (in Appendix 5.1) to deal with sample error.

Let u denote the vector average of all points in S^+. It is clear from symmetry that $u = (u_1, 0, \ldots, 0)$ for some $u_1 > 0$; in fact we have the following.

Claim 12. $u_1 = 2 \cdot \frac{A_{n-2}}{A_{n-1}} \cdot \int_0^1 z(1-z^2)^{(n-3)/2} dz = \Theta(\frac{1}{\sqrt{n}})$.

Proof. The first equality follows immediately from Fact 10 (the factor of 2 is present because u is the vector average of half the points of S^{n-1}). For the second equality, since $\frac{A_{n-2}}{A_{n-1}} = \Theta(\sqrt{n})$ we need to show that $\int_0^1 z(1-z^2)^{(n-3)/2} dz$ is $\Theta(1/n)$. For each $z \in [1/\sqrt{n}, 2/\sqrt{n}]$, the value of the integrand $z(1-z^2)^{(n-3)/2}$ is at least $(1/\sqrt{n})(1 - \frac{4}{n})^{(n-3)/2} = \Theta(1/\sqrt{n})$, and so this implies that the whole integral is $\Omega(1/n)$. The integrand is clearly at most $1/\sqrt{n}$ for all $z \in [0, 1/\sqrt{n}]$, and so we have $\int_0^{2/\sqrt{n}} z(1-z^2)^{(n-3)/2} dz = \Theta(1/n)$; to finish the proof we need only show that $\int_{2/\sqrt{n}}^1 z(1-z^2)^{(n-3)/2} dz = O(1/n)$. We can piecewise approximate this integral (in increments of $1/\sqrt{n}$) as

$$\int_{2/\sqrt{n}}^1 z(1-z^2)^{(n-3)/2} dz$$
$$\approx \sum_{j=2}^{\sqrt{n}} \frac{j}{\sqrt{n}} e^{-j^2/2} \cdot \frac{1}{\sqrt{n}} = \frac{1}{n} \sum_{j=2}^{\sqrt{n}} j e^{-j^2/2} < \frac{1}{n} \sum_{j=2}^{\infty} j e^{-j^2/2} = O\left(\frac{1}{n}\right),$$

and this gives the claim. □

If there were no noise, then the vector average v would equal u; since there is noise we must add in a contribution from true negative examples that are falsely labeled as positive and subtract off a contribution from true positive examples that are falsely labeled as negative.

Let opt_- and opt_+ be defined as

$$\mathsf{opt}_- = \int_{x \in S^-} p(x)\mathcal{U}(x) \qquad \text{and} \qquad \mathsf{opt}_+ = \int_{x \in S^+} p(x)\mathcal{U}(x),$$

and so opt_- is the overall probability of receiving an example that is truly negative but falsely labeled as positive, and vice versa for opt_+. Clearly, $\mathsf{opt} = \mathsf{opt}_- + \mathsf{opt}_+$. Let u^- and u^+ be the vectors

$$u^- = \frac{\int_{x \in S^-} xp(x)\mathcal{U}(x)}{\mathsf{opt}_-} \qquad \text{and} \qquad u^+ = \frac{\int_{x \in S^+} xp(x)\mathcal{U}(x)}{\mathsf{opt}_+},$$

and so u^- (u^+, respectively) is the vector average of all the false positive (false negative, respectively) examples generated by p. Then the vector average v of all positively labeled examples is

$$v = \frac{u/2 + \mathsf{opt}_- u^- - \mathsf{opt}_+ u^+}{1/2 + \mathsf{opt}_- - \mathsf{opt}_+} = C_1 \cdot v',$$

where $v' = u/2 + \mathsf{opt}_{-}u^{-} - \mathsf{opt}_{+}u^{+}$ and $\frac{4}{3} \leq C_1 = \frac{1}{1/2 + \mathsf{opt}_{-} - \mathsf{opt}_{+}} \leq 4$; the bounds on C_1 hold, since by assumption we have $\mathsf{opt} \leq \frac{1}{4}$. So v' is a constant multiple of v, and it suffices to analyze v'.

We have $v' = (v'_1, \ldots, v'_n)$, where v'_1 is the component parallel to e_1. In the rest of this subsection we will establish the following bounds on v'.

THEOREM 11. (i) *The component of v' that is parallel to the target vector e_1 is* $v'_1 \geq u_1(\frac{1}{2} - O(\mathsf{opt}\sqrt{\log\frac{1}{\mathsf{opt}}})) > \frac{u_1}{4}$.

(ii) *The component of v' that is orthogonal to e_1, namely $v'_\perp = v' - v'_1 e_1 = (0, v'_2, \ldots, v'_n)$, satisfies $\|v'_\perp\| = O(\mathsf{opt}\sqrt{\log\frac{1}{\mathsf{opt}}})u_1$.*

Given Theorem 11, the error rate of the hypothesis $\operatorname{sgn}(v \cdot x)$ under $\mathcal{U}$ is

$$\Pr[\operatorname{sgn}(v' \cdot x) \neq \operatorname{sgn}(x_1)] = \frac{\arctan\left(\frac{\|v'_\perp\|}{v'_1}\right)}{\pi} \leq \frac{\arctan(O(\mathsf{opt}\sqrt{\log\frac{1}{\mathsf{opt}}}))}{\pi} = O\left(\mathsf{opt}\sqrt{\log\frac{1}{\mathsf{opt}}}\right).$$

By Fact 11, the sample average vector $\overline{v}$ has $\Pr_{x\in\mathcal{U}}[\operatorname{sgn}(\overline{v} \cdot x) \neq \operatorname{sgn}(v \cdot x)] \leq \epsilon$ with probability at least $1 - \delta$, and we obtain Theorem 3.

Now we prove Theorem 11. Note that if $\mathsf{opt}_{-}u^{-} - \mathsf{opt}_{+}u^{+}$ is the zero vector, then the theorem clearly holds, and so we henceforth assume that $\mathsf{opt}_{-}u^{-} - \mathsf{opt}_{+}u^{+}$ is not the zero vector.

Fix any unit vector $w \in S^{n-1}$. Suppose that p is such that the vector $\mathsf{opt}_{-}u^{-} - \mathsf{opt}_{+}u^{+}$ points in the direction of w, i.e., $w = \frac{\mathsf{opt}_{-}u^{-} - \mathsf{opt}_{+}u^{+}}{\|\mathsf{opt}_{-}u^{-} - \mathsf{opt}_{+}u^{+}\|}$; let $\tau > 0$ denote $\|\mathsf{opt}_{-}u^{-} - \mathsf{opt}_{+}u^{+}\|$, and so $v' = u/2 + \tau w$. To establish Theorem 11, it suffices to show that the desired bounds hold for any function p which satisfies (12) and is such that (a) the vector $\mathsf{opt}_{-}u^{-} - \mathsf{opt}_{+}u^{+}$ points in the direction of w, and (b) the magnitude of $\tau = \|\mathsf{opt}_{-}u^{-} - \mathsf{opt}_{+}u^{+}\|$ is as large as possible. (Since $u/2$ contributes zero to $v'_\perp$, we have that $\|v_\perp\|$ scales with τ, and thus condition (ii) becomes only harder to satisfy as τ increases. If $w_1 > 0$, then condition (i) holds for any $\tau > 0$, and if $w_1 < 0$, then the larger τ is, the more difficult it is to satisfy condition (i).) We let $\tau_{\max}$ denote this maximum possible value of τ; if we can show that $|\tau_{\max}| = O(\mathsf{opt}\sqrt{\log\frac{1}{\mathsf{opt}}})u_1$, then since $v'_1 = \frac{u_1}{2} + \tau w_1$ and $v'_\perp = \tau(0, w_2, w_3, \ldots, w_n)$, this gives Theorem 11.

We upper bound $\tau_{\max}$ by considering an even more relaxed scenario. Let w be any unit vector in S^{n-1}. Let A be any subset of S^{n-1}, and let B be any subset of $S^{n-1} \setminus A$ such that $\mathsf{opt}_A + \mathsf{opt}_B = \mathsf{opt}$, where $\mathsf{opt}_A = \int_{x\in A} p(x)\mathcal{U}(x)$ and $\mathsf{opt}_B = \int_{x\in B} p(x)\mathcal{U}(x)$. Let $u^A = \frac{\int_{x\in A} xp(x)\mathcal{U}(x)}{\mathsf{opt}_A}$ and $u^B = \frac{\int_{x\in B} xp(x)\mathcal{U}(x)}{\mathsf{opt}_B}$. Let $p : S^{n-1} \to [0,1]$ be any function such that (i) (12) holds, and (ii) the vector $\mathsf{opt}_A u^A - \mathsf{opt}_B u^B$ points in the direction of w. If we can upper bound the magnitude of $\mathsf{opt}_A u^A - \mathsf{opt}_B u^B$, then this gives an upper bound on $\tau_{\max}$. (This is a more relaxed scenario because we are not requiring that $A \subseteq S^-$ and $B \subseteq S^+$.) But now a simple convexity argument shows that $\|\mathsf{opt}_A u^A - \mathsf{opt}_B u^B\|$ is maximized by taking A to be $\{x \in S^{n-1} : x \cdot w \geq y\}$, where y is chosen so that $\int_{x\in A} \mathcal{U}(x) = \frac{\mathsf{opt}}{2}$; taking B to be $-A$; and taking $p(x)$ to be 1 on $x \in (A \cup B)$ and 0 on $x \notin (A \cup B)$ (note that this gives $\mathsf{opt}_A = \mathsf{opt}_B = \frac{\mathsf{opt}}{2}$). Let τ_{MAX} be the value of $\|\mathsf{opt}_A u^A - \mathsf{opt}_B u^B\|$ that results from taking $A, B, \mathsf{opt}_A, \mathsf{opt}_B$, and p as described in the previous sentence; we will show that $\tau_{MAX} = O(\mathsf{opt}\sqrt{\log\frac{1}{\mathsf{opt}}})u_1$ and thus prove Theorem 11.

It is clear that $\mathsf{opt}_A u^A = -\mathsf{opt}_B u^B$, and so it suffices to bound $\|\mathsf{opt}_A u^A\| = \frac{\tau_{MAX}}{2}$.

Let $y \in [0,1]$ be the value specified above, and so

$$\frac{\mathsf{opt}}{2} = \Pr_{x \in \mathcal{U}}[x \cdot w \geq y] = \frac{A_{n-2}}{A_{n-1}} \cdot \int_y^1 (1-z^2)^{(n-3)/2} dz. \tag{13}$$

We have

$$\mathsf{opt}_A u^A = \int_{x \in A} x p(x) \mathcal{U}(x) = \left(\frac{A_{n-2}}{A_{n-1}} \cdot \int_y^1 z(1-z^2)^{(n-3)/2} dz \right) w,$$

and so it remains to show that $\gamma = O(\mathsf{opt}\sqrt{\log \frac{1}{\mathsf{opt}}})$, where $\gamma > 0$ is such that

$$\frac{A_{n-2}}{A_{n-1}} \cdot \int_y^1 z(1-z^2)^{(n-3)/2} dz = \gamma u_1, \tag{14}$$

where y satisfies (13). We do this in the following two claims.

CLAIM 13. *Let ℓ be such that $y = \frac{\ell}{\sqrt{n}}$. Then $e^{-\ell^2/2} = \Theta(\mathsf{opt})$.*

Proof. We have $\int_y^1 (1-z^2)^{(n-3)/2} dz = \frac{\mathsf{opt}}{2A_{n-2}/A_{n-1}} = \Theta(\frac{\mathsf{opt}}{\sqrt{n}})$. Write $y = \frac{\ell}{\sqrt{n}}$. Piecewise approximating the integral in increments of $1/\sqrt{n}$, we have

$$\int_y^1 (1-z^2)^{(n-3)/2} dz \approx \sum_{j=\ell}^{\sqrt{n}} e^{-j^2/2} \cdot \frac{1}{\sqrt{n}} = \Theta(e^{-\ell^2/2}) \cdot \frac{1}{\sqrt{n}}.$$

Since this equals $\Theta(\frac{\mathsf{opt}}{\sqrt{n}})$, we have that $e^{-\ell^2/2} = \Theta(\mathsf{opt})$, which gives the claim. (Note that we have $\ell = \Theta(\sqrt{\log \frac{1}{\mathsf{opt}}}) \gg 1$, which is compatible with approximating the integral with a sum as done above.) □

CLAIM 14. *We have $\gamma = \Theta(\mathsf{opt}\sqrt{\log(1/\mathsf{opt})})$.*

Proof. From Claim 12 we have $u_1 = \Theta(\frac{1}{\sqrt{n}})$. Since $\frac{A_{n-2}}{A_{n-1}} = \Theta(\sqrt{n})$, by (14) we have that $\gamma = \Theta(n \cdot \int_y^1 z(1-z^2)^{(n-3)/2} dz)$. Since $y = \ell/\sqrt{n}$, where $\ell = \Theta(\sqrt{\log(1/\mathsf{opt})})$ (and, more precisely, $e^{-\ell^2/2} = \Theta(\mathsf{opt})$) by Claim 13, again a piecewise approximation with pieces of length $1/\sqrt{n}$ gives us

$$\int_y^1 z(1-z^2)^{(n-3)/2} dz \approx \sum_{j=\ell}^{\sqrt{n}} \frac{j}{\sqrt{n}} \cdot e^{-j^2/2} \cdot \frac{1}{\sqrt{n}} < \frac{1}{n} \sum_{j=\ell}^{\infty} j e^{-j^2/2} = \Theta\left(\frac{\ell e^{-\ell^2/2}}{n} \right),$$

and thus $\gamma = \Theta(\mathsf{opt}\sqrt{\log 1/\mathsf{opt}})$ as desired. □

6. Learning halfspaces in the presence of malicious noise.

We now consider the problem of PAC learning an unknown origin-centered halfspace, under the uniform distribution on S^{n-1}, in the demanding *malicious noise model* introduced by Valiant [44] and subsequently studied by Kearns and Li [24] and many others.

We first define the malicious noise model. Given a target function f and a distribution $\mathcal{D}$ over X, a *malicious example oracle with noise rate* η is an oracle $\mathsf{EX}_\eta(f, \mathcal{D})$ that behaves as follows. Each time it is called, with probability $1-\eta$ the oracle returns a noiseless example $(x, f(x))$, where x is drawn from $\mathcal{D}$, and with probability η it returns a pair (x, y) about which nothing can be assumed; in particular such a "malicious" example may be chosen by a computationally unbounded adversary which has complete knowledge of f, $\mathcal{D}$, and the state of the learning algorithm when the oracle is invoked. We say that an algorithm *learns to error ϵ in the presence of malicious noise at rate η under the uniform distribution* if it satisfies the following condition: given access to $\mathsf{EX}_\eta(f,\mathcal{U})$ with probability $1-\delta$, the algorithm outputs a hypothesis h such that $\Pr_{x\in\mathcal{U}}[h(x) \neq f(x)] \leq \epsilon$.

Few positive results are known for learning in the presence of malicious noise. Improving on [44, 24], Decatur [10] gave an algorithm to learn disjunctions under any distribution that tolerates a noise rate of $O(\frac{\epsilon}{n}\ln\frac{1}{\epsilon})$. More recently, Mansour and Parnas [35] studied the problem of learning disjunctions under product distributions in an "oblivious" variant of the malicious noise model, giving an algorithm that can tolerate a noise rate of $O(\epsilon^{5/3}/n^{2/3})$. We note that the Perceptron algorithm can be shown to tolerate malicious noise at rate $O(\epsilon/\sqrt{n})$ when learning an origin-centered halfspace under the uniform distribution $\mathcal{U}$ on S^{n-1}.

It is not difficult to show that the simple `Average` algorithm can also tolerate malicious noise at rate $O(\epsilon/\sqrt{n})$.

THEOREM 12. *For any $\epsilon > 0$, algorithm* `Average` *(with $m = O(\frac{n^2}{\epsilon^2}\cdot\log\frac{n}{\delta})$) learns the class of origin-centered halfspaces to error ϵ in the presence of malicious noise at rate $\eta = O(\frac{\epsilon}{\sqrt{n}})$ under the uniform distribution.*

Proof. If there were no noise, the true average vector (average of all positive examples) would be $(u_1, 0, \ldots, 0)$, where by Claim 12 we have $u_1 = \Theta(1/\sqrt{n})$. By Chernoff bounds, we may assume that the true frequency η' of noisy examples in the sample is at most $2\eta = O(\epsilon/\sqrt{n})$. Let v denote the average of the noiseless vectors in the sample; Chernoff bounds are easily seen to imply that we have $v_1 = \Theta(1/\sqrt{n})$ and $|v_i| \leq \frac{\epsilon}{n}$ for each $i = 2, \ldots, n$. Let z denote the average location of the malicious examples in the sample; since even malicious examples must lie on S^{n-1} (for otherwise we could trivially identify and discard them), it must be the case that $\|z\| \leq 1$. From this it is easy to see that the average $\overline{v}$ of the entire sample must satisfy $\overline{v}_1 = \Theta(1/\sqrt{n}) - \epsilon/\sqrt{n} = \Theta(1/\sqrt{n})$ and $\sqrt{\overline{v}_2^2 + \cdots + \overline{v}_n^2} = O(\epsilon/\sqrt{n})$. We thus have $\Pr_{x\in\mathcal{U}}[\operatorname{sgn}(\overline{v}\cdot x) \neq \operatorname{sgn}(x_1)] = \alpha(\overline{v}, e_1)/\pi = \arctan(\frac{O(\epsilon/\sqrt{n})}{\Theta(1/\sqrt{n})})/\pi \leq \epsilon$. □

As the main result of this section, in section 6.1 we show that by combining the `Average` algorithm with a simple preprocessing step to eliminate some noisy examples, we can handle a higher malicious noise rate of $O(\frac{\epsilon}{(n\log n)^{1/4}})$; this is Theorem 4. This algorithm, which we call `TestClose`, is the following:

1. Draw examples from $\mathsf{EX}_\eta(f,\mathcal{U})$ until $m = O(\frac{n^2}{\epsilon^2}\log\frac{n}{\delta})$ positively labeled examples have been received; let $S = \{x^1, \ldots, x^m\}$ denote this set of examples.
2. Let $\rho = \sqrt{\frac{C}{n}\log\frac{m}{\delta}}$, where C is a fixed constant (specified later in section 6.1). If any pair of examples x^i, x^j with $i \neq j$ has $\|x^i - x^j\| < \sqrt{2-\rho}$, remove both x^i and x^j from S. (We say that such a pair of examples is *too close*.) Repeat this until no two examples in S are too close to each other. Let S' denote this "reduced" set of examples.
3. Now run `Average` on S' to obtain a vector $\overline{v}$, and return the hypothesis $h(x) = \operatorname{sgn}(\overline{v}\cdot x)$.

The idea behind this algorithm is simple. If there were no noise, then all examples received by the algorithm would be independent uniform random draws from the positive half of S^{n-1}, and it is not difficult to show that with high probability no two examples would be too close to each other. Roughly speaking, the adversary controlling the noise would like to cause $\overline{v}$ to point as far away from the true target vector as possible; in order to do this, his best strategy (if we were simply running the

Average algorithm on the original data set S without discarding any points) would be to have all noisy examples be located at some single particular point $x^\star \in S^{n-1}$. However, our "closeness" test rules out this adversary strategy, since it would certainly identify all these collocated points as being noisy and discard them. Thus, intuitively, in order to fool our closeness test, the adversary is constrained to place his noisy examples relatively far apart on S^{n-1} so that they will not be identified and discarded. But this means that the noisy examples cannot have a very large effect on the average vector $\overline{v}$, since intuitively placing the noisy examples far apart on S^{n-1} causes their vector average to have small magnitude and thus to affect the overall average $\overline{v}$ by only a small amount. The actual analysis in the proof of Theorem 4 uses bounds from the theory of sphere packing in $\mathbb{R}^n$ to make these intuitive arguments precise.

6.1. Proof of Theorem 4. Let $S_{bad} \subseteq S$ denote the set of "bad" examples in S that were chosen by the adversary, and let S_{good} be $S \backslash S_{bad}$, the set of "good" noiseless examples. Let S'_{bad} (S'_{good}, respectively) denote $S_{bad} \cap S'$ ($S_{good} \cap S'$, respectively), i.e., the set of bad (good, respectively) examples that survive the closeness test in step 2 of our algorithm.

Let us write v'_{good} to denote the vector average of all points in S'_{good} and v'_{bad} to denote the vector average of all points in S'_{bad}. If we let η' denote $\frac{|S'_{bad}|}{|S'|}$, then we have that the overall vector average $\overline{v}$ of all examples in S' is $(1-\eta')v'_{good} + \eta' v'_{bad}$.

We first show that our closeness test does not cause us to discard any good examples.

LEMMA 15. *With probability at least* $1 - \frac{\delta}{4}$ *we have* $S'_{good} = S_{good}$.

Proof. Let x' be any fixed point on S^{n-1}. We will show that a uniform example drawn from $\mathcal{U}$ lies within distance $\sqrt{2-\rho}$ of x' with probability at most $\frac{\delta}{4m^2}$. Since there are at most m examples in S_{good}, this implies that for any individual example $x^i \in S$, the probability that x^i lies too close to any example in S_{good} is at most $\frac{\delta}{2m}$; taking a union bound gives the lemma.

Without loss of generality we may take $x' = (1, 0, \dots, 0)$. It is easy to see that for any $y = (y_1, \dots, y_n) \in S^{n-1}$, we have $\|y - x'\| = \sqrt{2-2y_1}$ and thus $\|y-x'\| < \sqrt{2-\rho}$ if and only if $y > \rho/2$. But, by Fact 10, we have that if y is drawn from $\mathcal{U}$, then

$$\Pr_{y \in \mathcal{U}}\left[y > \frac{\rho}{2}\right] = \frac{A_{n-2}}{A_{n-1}} \cdot \int_{\rho/2}^{1} (1-z^2)^{(n-3)/2} dz. \tag{15}$$

It is easy to verify from the definition of ρ that for a suitable absolute constant C, the integrand $(1-z^2)^{(n-3)/2}$ is at most $(1-(\rho/2)^2)^{(n-3)/2} \le \frac{\delta}{4m^3}$ over the interval $[\rho/2, 1]$, and thus (since $A_{n-2}/A_{n-1} = \Theta(\sqrt{n}) < m$) we have that (15) is at most $\frac{\delta}{4m^2}$ as required. ☐

The true noise rate is η, and the previous lemma implies that with probability $1 - \frac{\delta}{4}$ we do not throw away any good examples from S. Using Chernoff bounds, it is easy to show that with overall probability at least $1 - \frac{\delta}{2}$ we have $\eta' < 2\eta$.

Let v_{good} denote $\frac{1}{|S_{good}|}\sum_{x \in S_{good}} x$, the average location of the vectors in S_{good}. We have that the expected value of v_{good} is $(u_1, 0, \dots, 0)$, where $u_1 = \Theta(\frac{1}{\sqrt{n}})$ is as defined in Claim 12. For $m = O(\frac{n^2}{\epsilon^2}\log\frac{n}{\delta})$, as in the proof of Fact 11, Chernoff bounds imply that with probability at least $1 - \frac{\delta}{4}$ we have that $(v_{good})_1 = \Theta(\frac{1}{\sqrt{n}})$ while $(v_{good})_i = O(\frac{\epsilon}{n})$ for each $i = 2, \dots, n$. By Lemma 15, with probability at least $1 - \frac{\delta}{4}$ we have $v'_{good} = v_{good}$, and so with overall probability at least $1 - \frac{\delta}{2}$ we have $(v'_{good})_1 = \Theta(\frac{1}{\sqrt{n}})$ while $(v'_{good})_i = O(\frac{\epsilon}{n})$ for each $i = 2, \dots, n$.

We now show that $\|v'_{bad}\|$ must be small; once we establish this, as we will see it is straightforward to combine this with the bounds of the previous two paragraphs to prove Theorem 4. The desired bound on $\|v'_{bad}\|$ is a consequence of the following lemma.

LEMMA 16. *Let T be any set of $M = \omega(n^{3/2}/\sqrt{\rho})$ many examples on S^{n-1} such that no two examples in T lie within distance $\sqrt{2-\rho}$ of each other (recall that $\rho = \sqrt{\frac{C}{n}\log\frac{m}{\delta}}$). Then the vector average $t = \frac{1}{|T|}\sum_{x\in T} x$ of T satisfies $\|t\| = O\big(\frac{(\log\frac{m}{\delta})^{1/2}}{n^{1/4}}\big)$.*

Proof. Without loss of generality we may suppose that $t = (c, 0, \dots, 0)$ for some $c > 0$ (by rotating the set T); our goal is to upper bound c. We consider a partition of T based on the value of the first coordinate as follows. For $\tau = 1, 1 - \frac{1}{\sqrt{n}}, 1 - \frac{2}{\sqrt{n}}, \dots,$ we define the set T_τ to be $\{x \in T : \tau - \frac{1}{2\sqrt{n}} \le x_i < \tau + \frac{1}{2\sqrt{n}}\}$. The idea of the proof is that for any value of τ which is not very small, the set T_τ must be small because of sphere-packing bounds. This implies that the overwhelming majority of the M examples in T must have a small first coordinate, which gives the desired result.

More precisely, we have the following claim.

CLAIM 17. *There is a fixed constant $K > 0$ such that if $\tau > K\sqrt{\rho}$, then $|T_\tau| \le n$.*

Proof. We first give a crude argument to show that if $\tau > 0.1$, then $|T_\tau| \le n$. (It will be clear from the argument that any positive constant could be used in this argument instead of 0.1.) This argument uses the same basic ideas as the general case of $\tau > K\sqrt{\rho}$ but is simpler because we do not need our bounds to be as precise; later, for the general case it will be useful to be able to assume that $\tau < 0.1$.

Fix some $\tau > 0.1$. We first note that if τ is greater than (say) $4/5$, then T_τ can contain at most one point (since any two points of S^{n-1} which both have first coordinate $4/5 \pm o(1)$ can have Euclidean distance at most $6/5 + o(1) < \sqrt{2-\rho}$ from each other). Thus we may assume that $0.1 < \tau < 4/5$ (the key aspect of the upper bound is that τ is bounded away from 1).

For $x \in \mathbb{R}^n$, let x' denote $(x_2, \dots, x_n)$. Since each $x \in T_\tau$ has $x_1 \in [\tau - \frac{1}{2\sqrt{n}}, \tau + \frac{1}{2\sqrt{n}})$, we have that each $x \in T_\tau$ satisfies $\|x'\| = \sqrt{1-\tau^2}\cdot(1 \pm o(1))$. Let $\tilde{x}' \in \mathbb{R}^{n-1}$ denote the rescaled version of x' so that $\|\tilde{x}'\|$ equals $\sqrt{1-\tau^2}$ exactly, and let $\tilde{T}'_\tau$ denote $\{\tilde{x}' : x \in T_\tau\}$. Since the first coordinates of any two points in T_τ differ by at most $\frac{1}{\sqrt{n}}$, it is not difficult to see that that the minimum pairwise distance condition on T_τ implies that any pair of points in $\tilde{T}'_\tau$ must have distance at least $(\sqrt{2-\rho} - \frac{1}{\sqrt{n}})\cdot(1 - o(1)) = \sqrt{2}\cdot(1 - o(1))$ from each other.

We now recall *Rankin's second bound* on the minimum pairwise distance for point sets on Euclidean spheres (see, e.g., Theorem 1.4.2 of [11]). This bound states that for any value $\kappa > \sqrt{2}$, at most $n+1$ points can be placed on S^{n-1} if each point is to have distance at least κ from all other points. By rescaling, this immediately implies that at most n points can be placed on the Euclidean sphere of radius $\sqrt{1-\tau^2}$ in $\mathbb{R}^{n-1}$ if all pairwise distances are at least $\kappa\sqrt{1-\tau^2}$. Now recall from the previous paragraph that all points in $\tilde{T}'_\tau$ lie on the sphere of radius $\sqrt{1-\tau^2}$, and all pairwise distances in $\tilde{T}'_\tau$ are at least $\sqrt{2}\cdot(1-o(1))$. It follows by a suitable choice of $\kappa > \sqrt{2}$ that $|\tilde{T}'_\tau|$, and thus $|T_\tau|$, is at most n.

We henceforth assume that $K\sqrt{\rho} < \tau < 0.1$ and give a more quantitatively precise version of the above argument to handle this case. We consider the following transformation f that maps points in T_τ onto the ball of radius $\sqrt{1-\tau^2}$ in $\mathbb{R}^{n-1}$:

given $x = (x_1, \ldots, x_n) \in T_\tau$, let

$$f(x) = \sqrt{1-\tau^2} \cdot \frac{x'}{\|x'\|};$$

i.e., $f(x)$ is obtained by removing the first coordinate and normalizing the resulting $(n-1)$-dimensional vector to have magnitude $\sqrt{1-\tau^2}$.

We now claim that if $x \neq y$, $x, y \in T_\tau$, then we have $\|f(x) - f(y)\| > \sqrt{2-\rho} - \frac{1}{\sqrt{n}} - \frac{3\tau^2}{5}$. To see this, fix any $x, y \in T_\tau$. By the triangle inequality, we have

$$\|f(x) - f(y)\| \geq \|x' - y'\| - \|f(x) - x'\| - \|f(y) - y'\|, \tag{16}$$

and so it suffices to bound the terms on the RHS.

For the first term, we have

$$\sqrt{2-\rho} \leq \|x - y\| \leq \frac{1}{\sqrt{n}} + \sqrt{(x_2-y_2)^2 + \cdots + (x_n - y_n)^2},$$

where the first inequality holds since $x, y \in T$ and the second inequality holds since the first coordinates of x and y differ by at most $\frac{1}{\sqrt{n}}$. This immediately gives $\|x' - y'\| \geq \sqrt{2-\rho} - \frac{1}{\sqrt{n}}$.

For the second term, since $x_1 \in [\tau - \frac{1}{2\sqrt{n}}, \tau + \frac{1}{2\sqrt{n}})$, it must be the case that

$$\|x'\|^2 = x_2^2 + \cdots + x_n^2 \in \left(1 - \left(\tau + \frac{1}{2\sqrt{n}}\right)^2, 1 - \left(\tau - \frac{1}{2\sqrt{n}}\right)^2\right]. \tag{17}$$

We have

$$\|f(x) - x'\| = \left\| \frac{(1-\tau^2)^{1/2}}{\|x'\|} x' - x' \right\| = \left| \frac{(1-\tau^2)^{1/2}}{\|x'\|} - 1 \right| \cdot \|x'\| \leq \left| \frac{(1-\tau^2)^{1/2}}{\|x'\|} - 1 \right|, \tag{18}$$

where the last inequality uses $\|x'\| \leq 1$. A tedious but straightforward verification (using the fact that $\tau < 0.1$) shows that condition (17) implies that the RHS of (18) is at most $\frac{\tau^2}{10}$ (see section 6.1.1 for the proof). The third term $\|f(y) - y'\|$ clearly satisfies the same bound.

Combining the bounds we have obtained, it follows from (16) that $\|f(x) - f(y)\| \geq \sqrt{2-\rho} - \frac{1}{\sqrt{n}} - \frac{\tau^2}{5}$. For some fixed absolute constant $K > 0$, we have that if $\tau^2 > K^2\rho$ (i.e., $\tau > K\sqrt{\rho}$), then the RHS of this last inequality is at least $\sqrt{2} - \frac{\tau^2}{2}$. So we have established that the transformed set of points $f(T_\tau)$ have all pairwise distances at least $\sqrt{2} - \frac{\tau^2}{2}$. But just as in the crude argument at the beginning of the proof, Rankin's bound implies that any point set on the radius-$\sqrt{1-\tau^2}$ ball in $\mathbb{R}^{n-1}$ with all pairwise distances strictly greater than $\sqrt{2} \cdot \sqrt{1-\tau^2}$ must contain at most n points. Since (as is easily verified) $\sqrt{2} - \frac{\tau^2}{2} > \sqrt{2} \cdot \sqrt{1-\tau^2}$, it must be the case that $|T_\tau| \leq n$. □

With Claim 17 in hand, it is clear that at most $n^{3/2}$ examples $x \in T$ can have $x_1 \geq K\sqrt{\rho}$. Since certainly each point in T has first coordinate at most 1, the average value of the first coordinate of all M points in T must be at most

$$\frac{n^{3/2} + MK\sqrt{\rho}}{M} \leq 2K\sqrt{\rho} = \Theta\left(\frac{(\log \frac{m}{\delta})^{1/4}}{n^{1/4}}\right)$$

(where we used $M = \omega(n^{3/2}/\sqrt{\rho})$ for the inequality above), and Lemma 16 is proved. $\square$

Lemma 16 implies that $\|v'_{bad}\| = O(\frac{(\log \frac{m}{\delta})^{1/4}}{n^{1/4}})$. (Note that if S'_{bad} is not of size M, we can augment it with examples from S'_{good} in order to make it large enough so that we can apply the lemma. This can easily be done, since we need only $M = \tilde{\omega}(n^{7/4})$ for the lemma and we have $|S_{good}| = \tilde{\Theta}(\frac{n^2}{\epsilon^2})$.) Putting all the pieces together, we have that with probability $1 - \delta$ all the following are true:

- $(v'_{good})_1 = \Theta(\frac{1}{\sqrt{n}})$;
- $(v'_{good})_i = O(\frac{\epsilon}{n})$ for $i = 2, \dots, n$;
- $\|v'_{bad}\| = O(\frac{(\log \frac{m}{\delta})^{1/4}}{n^{1/4}})$;
- $\eta' \leq 2\eta$, where $\overline{v} = (1 - \eta')v'_{good} + \eta' v'_{bad}$.

Combining all these bounds, a routine analysis shows that the angle between $\overline{v}$ and the target $(1, 0, \dots, 0)$ is at most ϵ, provided that

$$\frac{(2\eta)\frac{\log^{1/4}(m/\delta)}{n^{1/4}}}{\frac{1}{\sqrt{n}}} \leq c \cdot \epsilon$$

for some sufficiently small constant c. Rearranging this inequality, Theorem 4 is proved.

6.1.1. Proof that (18) is at most $\frac{\tau^2}{10}$. We have that (18) $\leq \left|\frac{(1-\tau^2)^{1/2}}{\|x'\|} - 1\right|$. To bound this quantity, we will consider the largest value greater than 1 and smallest value less than 1 that $\frac{(1-\tau^2)^{1/2}}{\|x'\|}$ can take. Throughout the following bounds, we repeatedly use the fact that $0 < \tau < 0.1$.

We have that

$$\|x'\| > \sqrt{1 - \left(\tau + \frac{1}{2\sqrt{n}}\right)^2} > 1 - \frac{9\tau^2}{16},$$

where the first inequality is from (17) and the second is easily verified (recall that $\tau > K\sqrt{\rho} > \frac{1}{n^{1/4}}$). Since $(1 - \tau^2)^{1/2} < 1 - \frac{\tau^2}{2}$, we have $\frac{(1-\tau^2)^{1/2}}{\|x'\|} < \frac{1-\tau^2/2}{1-9\tau^2/16} = 1 + \frac{\tau^2/16}{1-9\tau^2/16} < 1 + \frac{\tau^2}{10}$.

On the other hand, from (17) we also have that $\|x'\| \leq \sqrt{1 - (\tau - \frac{1}{2\sqrt{n}})^2}$, and so consequently we have (writing b for $\frac{1}{2\sqrt{n}}$ for readability below)

$$\frac{(1-\tau^2)^{1/2}}{\|x'\|} \geq \sqrt{\frac{1-\tau^2}{1-(\tau-b)^2}} = \sqrt{1 - \frac{2b\tau - b^2}{1-(\tau-b)^2}} > 1 - \frac{3}{5} \cdot \frac{2b\tau - b^2}{1-(\tau-b)^2}.$$

Recalling that $b = \frac{1}{2\sqrt{n}}$ whereas $\frac{1}{n^{1/4}} < \tau < 0.1$, we see that $\frac{3}{5} \cdot \frac{2b\tau-b^2}{1-(\tau-b)^2}$ is greater than 0 but is easily less than $\frac{\tau^2}{10}$.

We thus have that $\left|\frac{(1-\tau^2)^{1/4}}{\|x'\|} - 1\right| < \frac{\tau^2}{10}$ as claimed.

7. Conclusions and future work. We have given an algorithm that learns (under distributional assumptions) a halfspace in the agnostic setting. It constructs a polynomial threshold function whose error rate on future examples is within an additive ϵ of the optimal halfspace, in time poly(n) for any constant $\epsilon > 0$, for the uniform distribution over $\{-1,1\}^n$ or unit sphere in $\mathbb{R}^n$, as well as any log-concave distribution in $\mathbb{R}^n$. It also agnostically learns Boolean disjunctions in time $2^{\tilde{O}(\sqrt{n})}$ with respect to *any* distribution. Our algorithm has can be viewed as a noise-tolerant arbitrary-distribution generalization of the well-known "low-degree" Fourier algorithm of Linial, Mansour, and Nisan.

There are many natural ways to extend our work. One promising direction is to try to develop a broader range of learning results over the sphere S^{n-1} using the Hermite polynomials basis, in analogy with the rich theory of uniform distribution learning that has been developed for the parity basis over $\{-1,1\}^n$. Another natural goal is to gain a better understanding of the distributions and concept classes for which we can use the polynomial regression algorithm as an agnostic learner. Is there a way to extend the analysis of the $d=1$ case of the polynomial regression algorithm (establishing Theorem 3) to obtain a stronger version of Theorem 1, part 1(b)? Another natural idea would be to use the "kernel trick" with the polynomial kernel to speed up the algorithm. Finally, it would be interesting to explore whether the polynomial regression algorithm can be used for other challenging noisy learning problems beyond agnostic learning, such as learning with malicious noise.

Appendix A. Solving L_1 polynomial regression in polynomial time. Let $\mathcal{S}$ denote the set of all indices of monomials of degree at most d over variables $x_1,\dots,x_n$, and so $|\mathcal{S}| \le n^{d+1}$. Our goal is to find $w_S \in \mathbb{R}$ for $S \in \mathcal{S}$ to minimize $\frac{1}{m}\sum_{i=1}^m |y^i - \sum_{S\in\mathcal{S}} w_S (x^i)_S|$, where x_S is the monomial indexed by S. This can be done by solving the following LP:

$$\min \sum_{i=1}^m z_i \quad \text{such that} \quad \forall i: z_i \ge y^i - \sum_{S\in\mathcal{S}} w_S (x^i)_S \qquad \text{and}$$
$$z_i \ge -\left(y^i - \sum_{S\in\mathcal{S}} w_S (x^i)_S\right).$$

Using a polynomial-time algorithm for linear programming, this can be solved exactly in $n^{O(d)}$ time. In fact, for our purposes it is sufficient to obtain an approximate minimum, and hence one can use even more efficient algorithms [9].

Appendix B. Proof of Theorem 6.

Proof of Theorem 6. We assume without loss of generality that $\theta \ge 0$; an entirely similar proof works for $\theta < 0$. First, suppose that $\theta > \sqrt{d}$. Then we claim that the constant polynomial $p(x) = -1$ will be a sufficiently good approximation of $\operatorname{sgn}(x-\theta)$. In particular, it will have error:

$$\int_\theta^\infty \frac{4e^{-x^2}}{\sqrt{\pi}}dx \le \int_{\sqrt{d}}^\infty \frac{4e^{-x}}{\sqrt{\pi}}dx = \frac{4e^{-\sqrt{d}}}{\sqrt{\pi}} \le \frac{4}{\sqrt{\pi d}}.$$

So the case that $\theta > \sqrt{d}$ is easy, and for the remainder we assume that $\theta \in [0,\sqrt{d}]$.

We use the Hermite polynomials H_d, $d = 0,1,\dots$, (H_d is a degree-d *univariate* polynomial) which are a set of orthogonal polynomials given the weighting $e^{-x^2}\pi^{-1/2}$. In particular,

$$\int_{-\infty}^\infty H_{d_1}(x)H_{d_2}(x)\frac{e^{-x^2}}{\sqrt{\pi}}dx = \begin{cases} 0 & \text{if } d_1 \ne d_2, \\ 2^{d_1} d_1! & \text{if } d_1 = d_2. \end{cases}$$

Hence these polynomials form an orthogonal basis of polynomials with respect to the inner product $\langle p, q\rangle = \int_{-\infty}^{\infty} p(x)q(x)e^{-x^2}\pi^{-1/2}dx$. The functions $\bar{H}_d(x) = H_d(x)/\sqrt{2^d d!}$ are an orthonormal basis.

Now the best degree-d approximation to the function $\operatorname{sgn}(x-\theta)$, in the sense of (5), for any d, can be written as $\sum_{i=0}^{d} c_i\bar{H}_i(x)$. The $c_i \in \mathbb{R}$ that minimize (5) are

$$\begin{aligned} c_i &= \int_{-\infty}^{\infty} \operatorname{sgn}(x-\theta)\bar{H}_i(x)\frac{e^{-x^2}}{\sqrt{\pi}}dx \\ &= \int_{\theta}^{\infty} \bar{H}_i(x)\frac{e^{-x^2}}{\sqrt{\pi}}dx - \int_{-\infty}^{\theta} \bar{H}_i(x)\frac{e^{-x^2}}{\sqrt{\pi}}dx \\ &= 2\int_{\theta}^{\infty} \bar{H}_i(x)\frac{e^{-x^2}}{\sqrt{\pi}}dx \qquad (\text{for } i \geq 1). \end{aligned} \tag{19}$$

The last step follows from the fact that $\int_{-\infty}^{\infty} \bar{H}_i(x)\frac{e^{-x^2}}{\sqrt{\pi}}dx = 0$ for $i \geq 1$ by orthogonality of $\bar{H}_i$ with $\bar{H}_0$. Next, to calculate our error, we use Parseval's identity:

$$\int_{-\infty}^{\infty}\left(\sum_{i=0}^{d} c_i\bar{H}_i(x) - \operatorname{sgn}(x-\theta)\right)^2 \frac{e^{-x^2}}{\sqrt{\pi}}dx = 1 - \sum_{i=0}^{d} c_i^2 = \sum_{i=d+1}^{\infty} c_i^2.$$

The above holds because $\int_{-\infty}^{\infty}\frac{e^{-x^2}}{\sqrt{\pi}} = 1$ and hence $\sum_{i=0}^{\infty} c_i^2 = 1$ ($\operatorname{sgn}(x) \in L^2(\mathbb{R}, e^{-x^2})$ and polynomials are dense in this set). It thus suffices for us to bound $\sum_{i=d+1}^{\infty} c_i^2$.

It is now easy to calculate each coefficient c_i using standard properties of the Hermite polynomials. It is well known [42] that the Hermite polynomials can be defined by

$$H_i(x)e^{-x^2} = (-1)^i\frac{d^n}{dx^n}e^{-x^2}, \qquad \text{which implies } \frac{d}{dx}H_i(x)e^{-x^2} = -H_{i+1}(x)e^{-x^2}.$$

In turn, this and (19) imply that, for $i \geq 1$,

$$\begin{aligned} c_i &= \frac{2}{\sqrt{\pi 2^i i!}}\int_{\theta}^{\infty} H_i(x)e^{-x^2}dx \\ &= \frac{2}{\sqrt{\pi 2^i i!}}\left(-H_{i-1}(x)e^{-x^2}\right)\Big|_{\theta}^{\infty} \\ &= \frac{2}{\sqrt{\pi 2^i i!}}H_{i-1}(\theta)e^{-\theta^2}. \end{aligned} \tag{20}$$

We must show that $\sum_{i=d+1}^{\infty} c_i^2 = O(1/\sqrt{d})$. To do this, it suffices to show that for each i we have $c_i^2 = O(i^{-3/2})$. From (20) we have, for $i \geq 1$,

$$c_i^2 = \frac{4}{\pi 2^i i!}(H_{i-1}(\theta))^2 e^{-2\theta^2}. \tag{21}$$

Now, conveniently, Theorem 1.i of [7] states that, for all $i \geq \theta^2$,

$$\frac{1}{2^i i!}H_i(\theta)^2 e^{-\theta^2} \leq \frac{C}{\sqrt{i}},$$

where C is some absolute constant. Since we have $\theta \leq \sqrt{d}$ by assumption, we have that, for $i \geq d+1$, $c_i^2 \leq \frac{4C}{2\pi i\sqrt{i-1}}$, which is of the desired form $O(i^{-3/2})$, and Theorem 6 is proved. $\square$

REFERENCES

[1] E. Baum, *The Perceptron algorithm is fast for nonmalicious distributions*, Neural Comput., 2 (1990), pp. 248–260.

[2] E. B. Baum and Y-D. Lyuu, *The transition to perfect generalization in perceptrons*, Neural Comput., 3 (1991), pp. 386–401.

[3] A. Blum, *Machine learning: A tour through some favorite results, directions, and open problems*, FOCS 2003 tutorial slides; available online from http://www-2.cs.cmu.edu/∼avrim/Talks/FOCS03/tutorial.ppt (2003).

[4] A. Blum, A. Frieze, R. Kannan, and S. Vempala, *A polynomial time algorithm for learning noisy linear threshold functions*, Algorithmica, 22 (1997), pp. 35–52.

[5] A. Blum, M. L. Furst, J. Jackson, M. J. Kearns, Y. Mansour, and S. Rudich, *Weakly learning dnf and characterizing statistical query learning using Fourier analysis*, in Proceedings of the 26th Annual ACM Symposium on Theory of Computing, 1994, pp. 253–262.

[6] A. Blum, A. Kalai, and H. Wasserman, *Noise-tolerant learning, the parity problem, and the statistical query model*, J. ACM, 50 (2003), pp. 506–519.

[7] S. Bonan and D. Clark, *Estimates of the Hermite and the Freud polynomials*, J. Approx. Theory, 63 (1990), pp. 210–224.

[8] N. Bshouty and C. Tamon, *On the Fourier spectrum of monotone functions*, J. ACM, 43 (1996), pp. 747–770.

[9] K. L. Clarkson, *Subgradient and sampling algorithms for l1 regression*, in Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2005, pp. 257–266.

[10] S. Decatur, *Statistical queries and faulty PAC oracles*, in Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory, 1993, pp. 262–268.

[11] T. Ericson and V. Zinoviev, *Codes on Euclidean Spheres*, North–Holland Math. Library 63, North–Holland, Amsterdam, 2001.

[12] V. Feldman, *Optimal hardness results for maximizing agreements with monomials*, in Proceedings of the 21st Annual IEEE Conference on Computational Complexity, 2006, pp. 226–236.

[13] V. Feldman, P. Gopalan, S. Khot, and A. Ponnuswami, *New results for learning noisy parities and halfspaces*, in Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, 2006, pp. 563–574.

[14] Y. Freund and R. Schapire, *A decision-theoretic generalization of on-line learning and an application to boosting*, J. Comput. System Sci., 55 (1997), pp. 119–139.

[15] Y. Freund and R. Schapire, *Large margin classification using the Perceptron algorithm*, in Proceedings of the Eleventh Annual ACM Conference on Computational Learning Theory, 1998, pp. 209–217.

[16] Y. Freund and R. Schapire, *A short introduction to boosting*, J. Japanese Soc. Artificial Intelligence, 14 (1999), pp. 771–780.

[17] S. Goldman, M. Kearns, and R. Schapire, *On the sample complexity of weakly learning*, Inform. Comput., 117 (1995), pp. 276–287.

[18] V. Guruswami and P. Raghavendra, *Hardness of learning halfspaces with noise*, in Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, 2006, pp. 543–552.

[19] A. Hajnal, W. Maass, P. Pudlak, M. Szegedy, and G. Turan, *Threshold circuits of bounded depth*, J. Comput. System Sci., 46 (1993), pp. 129–154.

[20] J. Jackson, *The Harmonic Sieve: A Novel Application of Fourier Analysis to Machine Learning Theory and Practice*, Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, 1995.

[21] J. Jackson, *An efficient membership-query algorithm for learning DNF with respect to the uniform distribution*, J. Comput. System Sci., 55 (1997), pp. 414–440.

[22] J. Jackson, A. Klivans, and R. Servedio, *Learnability beyond AC^0*, in Proceedings of the 34th Annual ACM Symposium on Theory of Computing, 2002, pp. 776–784.

[23] M. Kearns, *Efficient noise-tolerant learning from statistical queries*, J. ACM, 45 (1998), pp. 983–1006.

[24] M. Kearns and M. Li, *Learning in the presence of malicious errors*, SIAM J. Comput., 22 (1993), pp. 807–837.

[25] M. Kearns, R. Schapire, and L. Sellie, *Toward efficient agnostic learning*, Mach. Learn., 17 (1994), pp. 115–141.

[26] A. Klivans, R. O'Donnell, and R. Servedio, *Learning intersections and thresholds of halfspaces*, J. Comput. System Sci., 68 (2004), pp. 808–840.

[27] A. Klivans and R. Servedio, *Boosting and hard-core sets*, in Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science, 1999, pp. 624–633.

[28] A. R. Klivans and A. A. Sherstov, *Cryptographic hardness for learning intersections of half-*

spaces, in Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, 2006, pp. 553–562.

[29] W. Lee, P. Bartlett, and R. Williamson, *Efficient agnostic learning of neural networks with bounded fan-in*, IEEE Trans. Inform. Theory, 42 (1996), pp. 2118–2132.

[30] W. S. Lee, P. L. Bartlett, and R. C. Williamson, *On efficient agnostic learning of linear combinations of basis functions*, in Proceedings of the Eighth Annual ACM Conference on Computational Learning Theory, 1995, pp. 369–376.

[31] N. Linial, Y. Mansour, and N. Nisan, *Constant depth circuits, Fourier transform and learnability*, J. Assoc. Comput. Mach., 40 (1993), pp. 607–620.

[32] P. Long, *On the sample complexity of PAC learning halfspaces against the uniform distribution*, IEEE Trans. Neural Networks, 6 (1995), pp. 1556–1559.

[33] P. Long, *An upper bound on the sample complexity of pac learning halfspaces with respect to the uniform distribution*, Inform. Process. Lett., 87 (2003), pp. 229–234.

[34] L. Lovász and S. Vempala, *Logconcave functions: Geometry and efficient sampling algorithms*, in Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, 2003, pp. 650–659.

[35] Y. Mansour and M. Parnas, *Learning conjunctions with noise under product distributions*, Inform. Process. Lett., 68 (1998), pp. 189–196.

[36] N. Nisan and M. Szegedy, *On the degree of Boolean functions as real polynomials*, in Proceedings of the 24th Annual ACM Symposium on Theory of Computing, 1992, pp. 462–467.

[37] R. O'Donnell and R. Servedio, *New degree bounds for polynomial threshold functions*, in Proceedings of the 35th Annual ACM Symposium on Theory of Computing, 2003, pp. 325–334.

[38] R. Paturi, *On the degree of polynomials that approximate symmetric Boolean functions*, in Proceedings of the 24th Annual ACM Symposium on Theory of Computing, 1992, pp. 468–474.

[39] O. Regev, *On lattices, learning with errors, random linear codes, and cryptography*, in Proceedings of the 37th Annual ACM Symposium on Theory of Computing, 2005, pp. 84–93.

[40] R. Servedio, *On PAC learning using Winnow, Perceptron, and a Perceptron-like algorithm*, in Proceedings of the Twelfth Annual ACM Conference on Computational Learning Theory, 1999, pp. 296–307.

[41] J. Shawe-Taylor and N. Cristianini, *An Introduction to Support Vector Machines*, Cambridge University Press, Cambridge, UK, 2000.

[42] G. Szegö, *Orthogonal Polynomials*, Amer. Math. Soc. Colloq. Publ. 23, AMS, Providence, RI, 1989.

[43] L. Valiant, *A theory of the learnable*, Comm. ACM, 27 (1984), pp. 1134–1142.

[44] L. Valiant, *Learning disjunctions of conjunctions*, in Proceedings of the Ninth International Joint Conference on Artificial Intelligence, Morgan Kaufmann, San Francisco, 1985, pp. 560–566.

[45] V. Vapnik, *Statistical Learning Theory*, Wiley-Interscience, New York, 1998.

SIAM J. Comput.
Vol. 37, No. 6, pp. 1806–1841

LOWER BOUNDS FOR THE NOISY BROADCAST PROBLEM*

NAVIN GOYAL[†], GUY KINDLER[‡], AND MICHAEL SAKS[§]

Abstract. We prove the first nontrivial (superlinear) lower bound in the *noisy broadcast model*, defined by El Gamal in [*Open problems presented at the* 1984 *workshop on Specific Problems in Communication and Computation sponsored by Bell Communication Research*, in Open Problems in Communication and Computation, T. M. Cover and B. Gopinath, eds., Springer-Verlag, New York, 1987, pp. 60–62]. In this model there are $n+1$ processors $P_0, P_1, \dots, P_n$, each of which is initially given a private input bit x_i. The goal is for P_0 to learn the value of $f(x_1, \dots, x_n)$, for some specified function f, using a series of *noisy broadcasts*. At each step a designated processor broadcasts one bit to all of the other processors, and the bit received by each processor is flipped with fixed probability (independently for each recipient). In 1988, Gallager [*IEEE Trans. Inform. Theory*, 34 (1988), pp. 176–180] gave a noise-resistant protocol that allows P_0 to learn the entire input with constant probability in $O(n \log\log n)$ broadcasts. We prove that Gallager's protocol is optimal, up to a constant factor. Our lower bound follows by reduction from a lower bound for *generalized noisy decision trees*, a new model which may be of independent interest. For this new model we show a lower bound of $\Omega(n \log n)$ on the depth of a tree that learns the entire input. While the above lower bound is for an n-bit function, we also show an $\Omega(n \log\log n)$ lower bound for the number of broadcasts required to compute certain explicit *boolean-valued* functions, when the correct output must be attained with probability at least $1-n^{-\alpha}$ for a constant parameter $\alpha > 0$ (this bound applies to all threshold functions as well as any other boolean-valued function with linear sensitivity). This bound also follows by reduction from a lower bound of $\Omega(n \log n)$ on the depth of generalized noisy decision trees that compute the same functions with the same error. We also show a (nontrivial) $\Omega(n)$ lower bound on the depth of generalized noisy decision trees that compute such functions with small constant error. Finally, we show the first protocol in the noisy broadcast model that computes the Hamming weight of the input using a linear number of broadcasts.

Key words. distributed computation, computation in presence of noise

AMS subject classifications. 68Q10, 68Q17

DOI. 10.1137/060654864

1. Introduction.

The relationships between noise, communication, and computation have been studied extensively. A recurring problem, arising in both theory and practice, is to minimize the additional resources needed to obtain reliable results in the presence of noise. This problem was studied in the context of decision trees [12, 26, 10, 8, 22], formulas and circuits [24, 15, 29, 18, 9, 30], sorting networks [20], cellular automata [14], quantum computation [1], data structures [13, 4], various communication models [16, 28, 19, 27, 22], and other models [2, 7, 23, 17].

The *noisy broadcast model* was proposed by El Gamal [6] in 1984, and later popularized by Yao [33], as a simple model in which to study the effect of noise in

[*] Received by the editors March 22, 2006; accepted for publication (in revised form) February 20, 2007; published electronically March 26, 2008.
http://www.siam.org/journals/sicomp/37-6/65486.html

[†] Department of Computer Science, Rutgers University, Piscataway, NJ 08854 (ngoyal@cs.rutgers.edu). Current address: School of Computer Science, McGill University, Montreal H3A 2A7, QC, Canada (navin@cs.mcgill.ca). This author's work was supported in part by NSF grant CCR-9988526 and a Bevier Fellowship of Rutgers University.

[‡] Theory Group, Microsoft Research, Redmond, WA 98052 (gkindler@microsoft.com). Current address: Department of Applied Mathematics and Computer Science, Weizmann Institute of Science, Rehovot 76100, Israel (gkindler@weizmann.ac.il). This author's work was supported by NSF grants CCR-0324906 and DMS-0111298.

[§] Department of Mathematics, Rutgers University, Piscataway, NJ 08854 (saks@math.rutgers.edu). This author's work was supported in part by NSF grants CCR-9988526 and CCR-0515201.

a highly distributed system. This model considers n processors, $P_1, \ldots, P_n$, and a receiver P_0. Each processor P_i has a private input bit x_i, and the goal is for P_0 to evaluate a specified function $f(x_1, \ldots, x_n)$ using the smallest possible number of broadcasts. Communication is carried out in synchronous time steps: in each step a prespecified processor broadcasts a single bit to all other processors. For some fixed noise parameter $\varepsilon < 1/2$, each of the other processors independently receives the broadcast bit (with probability $1 - \varepsilon$) or the complement of the bit (with probability ε). After the final broadcast, P_0 determines the output of the protocol from the bits it has heard. If for every given input $x \in \{0,1\}^n$ the protocol outputs $f(x)$ with probability at least $1 - \delta$, we say that it computes f with error at most δ for noise parameter ε.

In this paper we study the case where the receiver, P_0, aims to output the entire input. Thus the function to be computed is the identity function, denoted *id*, which is clearly the hardest function to compute in this model. There is a simple protocol that computes *id* with error δ, for any fixed $\delta \in (0, 1/2)$, using $\Theta(n \log n)$ broadcasts: for each $i \in \{1, \ldots, n\}$, P_i broadcasts its bit $c \log n$ times (for some large enough constant $c = c(\varepsilon, \delta)$) and P_0 outputs the majority value of the copies of x_i it received. In 1988, Gallager [16] gave a protocol for *id* using only $O(n \log\log n)$ broadcasts (we give a simplified variant of his protocol in section 7), and this remains the best upper bound known. Previous to this paper, the only known lower bound was the trivial $\Omega(n)$. In this paper we prove an $\Omega(n \log\log n)$ lower bound, thereby showing that Gallager's upper bound is optimal up to a constant factor.

Related work. Kushilevitz and Mansour [19] showed that the majority function (or any other threshold function) can be computed using $O(n)$ broadcasts. This protocol takes advantage of a rather strong and unrealistic feature of the model: that the noise occurring in different receptions is independent and identically distributed. To consider protocols that are less dependent on the exact distribution of noise, Feige and Kilian [11] proposed a stronger adversarial model of noise. Roughly speaking, this model allows an omniscient adversary to cancel any of the errors introduced by the random noise, thus preventing the algorithm from taking advantage of stochastic regularities in the noise. Feige and Kilian showed that even against this stronger adversary OR can be computed in $O(n \log^* n)$ broadcasts. Newman [22] improved this to $O(n)$ broadcasts (Newman stated his result for a weaker adversarial model, but his result easily carries over to the stronger adversarial model of [11].)

Some results in closely related models are also worth mentioning: efficient error resilient protocols were given for the noisy two-party communication complexity by Schulman [28], and for noisy communication networks with small degree by Rajagopalan and Schulman [25].

1.1. Our results.

In this paper we prove that Gallager's $O(n \log\log n)$ protocol for *id* is optimal.

THEOREM 1. *Let $\varepsilon \in (0, 1/2)$ be any noise parameter, and let $\beta \geq 1$ and n be a positive integer. Let A be a (possibly randomized) noisy broadcast protocol for n processors using at most βn broadcasts. If A is executed with noise parameter ε on a random input X distributed uniformly in $\{0,1\}^n$, then the probability that the receiver outputs X is at most*

$$\sqrt{\frac{1}{n} + \frac{48\beta^2 \log(1/\varepsilon)}{\varepsilon^{4\beta} \log n}}.$$

This immediately gives the following corollary.

Corollary 2. *Let $\varepsilon \in (0, 1/2)$ be any constant. Then any noisy broadcast protocol which computes the identity function with error at most $1/3$ for noise parameter ε requires $\Omega(n \log \log n)$ broadcasts.*

Our lower bound is formulated for the original noise model of El Gamal; trivially, it also holds for the adversarial noise models as well. Previously, no superlinear lower bounds were known even for the adversarial noise model.

Generalized noisy decision trees. Our lower bound for the noisy broadcast model is obtained by reducing that model to a new model, which we call the *generalized noisy decision tree (*gnd*-tree) model.* This model, which may be of independent interest, considers a more centralized computational setting where a single decision tree attempts to evaluate a function of the boolean input x. The gnd-tree does not have direct access to x—instead it has access to an unlimited number of *noisy copies* of x. In each noisy copy of x each bit is complemented independently with some fixed probability $\varepsilon < 1/2$. At each step the tree selects one of the copies of x and asks for the evaluation of any boolean-valued function at that copy. (A precise definition of the model appears in section 2.)

The gnd-tree model can be seen as a generalization of the noisy decision tree (nd-tree) model introduced by Feige et al. [12]. The nd-tree model is obtained by restricting the gnd-tree model to *coordinate queries*, namely queries q of the form $q(y_1, \dots, y_n) = y_i$, $i \in \{1, \dots, n\}$. The computational power of these two models is not the same: the majority function requires $\Omega(n \log n)$ queries by an nd-tree [12], but can be computed with just $O(n)$ queries using a gnd-tree, by adapting the aforementioned noisy broadcast protocol for majority due to Kushilevitz and Mansour [19].

The proof of Theorem 1 has two parts. The first part is a reduction that takes a protocol in the noisy broadcast model and simulates it by a gnd-tree. The second (and more substantial) part is a lower bound on the depth of gnd-trees required to compute the identity function. It seems that proving lower bounds for gnd-trees is easier than proving them directly for the noisy broadcast model, because the former model is more centralized, making it easier to analyze the progress made in intermediate steps of the computation.

Note that *id* can be easily computed by a gnd-tree (indeed, even by an nd-tree) of depth $O(n \log n)$: the tree just queries all coordinates of $O(\log n)$ noisy copies of the input and outputs the bitwise majority of the copies. Our main lower bound for gnd-trees implies that this protocol is within a constant factor of optimal.

Theorem 3. *Let $\varepsilon \in (0, 1/2)$ be a noise parameter, and let n be a positive integer. If a (possibly randomized)* gnd*-tree T of depth d is run on an input X chosen uniformly from $\{0,1\}^n$, with noise parameter ε, then the probability that T outputs X is bounded above by*

$$\frac{1}{\sqrt{n}} + \frac{6 \log(1/\varepsilon)}{\varepsilon^2 \log n} \cdot \left(\frac{d}{n} + \sqrt{\frac{d}{n}} \right).$$

In particular, when $d \geq n$, the probability that T outputs X is bounded above by

$$\frac{1}{\sqrt{n}} + \frac{12 \log(1/\varepsilon)}{\varepsilon^2 \log n} \cdot \frac{d}{n}.$$

Decision functions. A significant problem left open by Theorem 1 is whether there are *decision functions* (functions that output a boolean value) that require a superlinear number of broadcasts in the noisy broadcast model. While it seems intuitively clear that there should be such functions, proving this seems difficult. Counting arguments do not seem useful here because the number of possible protocols grows too quickly as a function of the number of broadcasts. On the other hand, the linear upper bound on threshold functions of Kushilevitz and Mansour [19], as well as our linear protocol for computing Hamming weight mentioned below, indicate that it may be difficult to prove a superlinear lower bound for an explicit function.

Proving lower bounds for decision functions in the gnd-tree model is even more problematic than in the noisy broadcast model. It is trivial that in the noisy broadcast model every function that depends on all of its variables needs at least n broadcasts in the worst case (since the choice of who broadcasts is fixed in advance). For gnd-trees, however, we cannot rule out the existence of trees of sublinear depth even for functions whose (noiseless) decision-tree complexity is linear (also, there are highly nontrivial functions whose decision-tree complexity is sublinear and that can also be computed in sublinear depth by gnd-trees). We do not know of any simple arguments that yield linear lower bounds for gnd-trees computing decision functions, or even bounds of the form $n^{1-\alpha}$ for small positive $\alpha > 0$.

Our proof of Theorem 3 is based on the idea that it is hard for a gnd-tree or a noisy broadcast protocol to distinguish an input from its immediate neighbors. This suggests the parity function as a candidate for proving superlinear lower bounds. It turns out, however, that any boolean function whose output depends only on the Hamming weight $\sum_i x_i$ of the input can be computed in the noisy broadcast model with a linear number of broadcasts.

Theorem 4. *For any $\varepsilon, \delta \in (0, 1/2)$ there is a noisy broadcast protocol that computes the Hamming weight of the input with probability of error at most δ for noise parameter ε, which uses cn broadcasts for some constant $c = c(\varepsilon, \delta)$. Therefore every symmetric boolean function (namely a function which depends only on the Hamming weight of its input) can be computed using a linear number of broadcasts.*

Similarly, there exists a gnd-tree that computes $\sum_i x_i$ with error at most δ for noise parameter ε and has depth dn for some constant $d = d(\varepsilon, \delta)$.

Our protocol for computing the Hamming weight of the input borrows some of the ideas from the majority protocol of [19]. While it is somewhat surprising that such a protocol exists, it has the same drawback as the majority protocol: it relies heavily on the unrealistic assumption of the model that every bit received experiences independent noise precisely ϵ.

Lower bounds for decision functions. We are able to get some nontrivial lower bounds for computing decision functions, both in the noisy broadcast model and for gnd-trees, by adapting the techniques from the proof of Theorem 3. Our bounds for computing a boolean function f depend on its *sensitivity*, s_f. The sensitivity of a function f is the maximum, over all $x \in \{0, 1\}^n$, of the number of neighboring inputs y (inputs y that differ from x on one coordinate) for which $f(y) \neq f(x)$. In particular, $s(f) = n$ for AND, OR, and PAR, and is at least $n/2$ for any nontrivial symmetric boolean function. Our results, which are more accurately stated and proved in section 5, are summarized below.

Theorem 5. *Let $\varepsilon, \delta \in (0, 1/2)$ and $\alpha > 0$ be fixed constants. Let $f : \{0, 1\}^n \to \{0, 1\}$ be any decision function, and let $s = s(f)$ denote the sensitivity of f. Then*

- *the depth of any gnd-tree that computes f with error at most δ for noise parameter ε is at least $\Omega(s)$;*
- *the depth of any gnd-tree that computes f with error at most $n^{-\alpha}$ for noise parameter ε is at least $\Omega(s \log(s))$;*
- *any noisy broadcast protocol that computes f with error at most $n^{-\alpha}$ for noise*

parameter ε requires $\Omega(s \log\log(s))$ broadcasts.

We note that the first section in Theorem 5 implies that Theorem 4 is essentially tight. That is, a gnd-tree that computes a nontrivial symmetric function must be of at least a linear depth (a similar lower bound for the noisy broadcast model is trivially true).

1.2. Organization. In section 2 we formally define the noisy broadcast and gnd-tree models. We then state a reduction theorem between noisy broadcast protocols and gnd-trees, and use it to show that Theorem 3 implies Theorem 1. Section 3 contains some technical preliminaries for the proofs of the main results. The proof of Theorem 3 is given in section 4. Section 5 contains the precise statements and proofs for our lower bounds on decision functions. Section 6 proves the reduction theorem, showing how to simulate noisy broadcast protocols by gnd-trees. In section 7 we give our variant of Gallager's $O(n \log\log n)$ noisy broadcast protocol for *id*, and in section 8 we give our linear noisy broadcast protocol for computing the Hamming weight of the input. We conclude the paper with some open problems in section 9.

2. The noisy broadcast model and gnd-trees.

In this section we begin by defining the noisy broadcast and gnd-tree models. Next, we state a general reduction lemma from the first model to the second and use this lemma to derive Theorem 1 from Theorem 3.

2.1. Noisy copies. Let $\varepsilon \in (0, 1/2)$ be a noise parameter. An *ε-noise bit* is a $\{0,1\}$-valued random variable that takes value 1 with probability ε. An *ε-noise k-vector* N is a sequence of k independent ε-noise bits.

For a bit-vector $x \in \{0,1\}^k$, an *ε-noisy copy of* x is a random variable of the form $x \oplus N$, where $\oplus$ denotes the bitwise XOR, and N is an ε-noise k-vector. More generally, if X is any random variable taking values in $\{0,1\}^k$, an *ε-noisy copy of* X is a random variable of the form $X \oplus N$, where N is an ε-noise k-vector chosen independently from X.

2.2. Computation under noise. In standard computation models, a deterministic computation is determined by its input, and a randomized computation is determined by the input and some auxiliary independent unbiased random bits.

We will discuss several models for computing in the presence of noise. In such models, the computation also depends (in some specified way) on a boolean vector N that represents the noise that affects the computation. It is assumed that N is a ε-noise vector for some $\varepsilon \in (0, 1/2)$, and that N is independent of any random bits used by the algorithm.

For $\delta \in [0,1]$ we say that an algorithm computes a function f with error at most δ against ε-noise if, for each input x, the algorithm outputs $f(x)$ with probability at least $1 - \delta$, where the probability is taken with respect to the auxiliary random bits of the algorithm and the ε-noise vector N.

2.3. The noisy broadcast model. The *noisy broadcast model* considers one *receiver* P_0 and *n processors* $P_1, \ldots, P_n$. The input is a boolean vector x of length n, and each of the processors P_i initially has coordinate x_i. The goal is for P_0 to evaluate a specified function f at x. This goal is to be accomplished by a *noisy broadcast protocol.*

The specification of a noisy broadcast protocol consists of

- the number s of broadcasts used in the protocol,
- a sequence $i^1, \ldots, i^s$ of indices of processors (with repetitions allowed),

- a sequence $g^1, \dots, g^s$ of *broadcast functions*, where $g^j : \{0,1\}^j \longrightarrow \{0,1\}$,
- a *output function* h which is defined over the domain $\{0,1\}^s$.

Running a protocol. The execution of a noisy broadcast protocol A depends on the input x and a noise vector N. We will think of N as a concatenation of s independent noise vectors $N^1, \dots, N^s$, each of length $n+1$. In the jth step of the execution of A the processor P_{i^j} broadcasts a bit b^j, and each of the other processors receives an independent noisy copy of b^j. Formally, P_h receives $b^j_h = b^j \oplus N^j_h$; it will be convenient (but unimportant) to regard P_{i^j} as receiving a noisy copy $b^j_j = b^j \oplus N^j_j$ of his own message. The bit b^j broadcast by P_{i^j} at step j is g^j evaluated at the j-vector consisting of P_{i^j}'s input bit and the $j-1$ bits received by P_{i^j} during the first $j-1$ rounds. Thus $b^j = g^j(x_{i^j}, b^1_{i^j}, b^2_{i^j}, \dots, b^{j-1}_{i^j})$. The output of the protocol is $h(b^0_1, \dots, b^0_s)$, that is, the value of h on the s-vector of bits received by P_0.

The randomized version of the model is defined in the natural way. Each processor has access to a source that generates independent random bits where the processor has the ability to specify the bias of each successive bit. The function g^j determining the bit broadcast by P_{i_j} may depend on the random bits generated by P_{i_j}.

Our version of the noisy broadcast model is similar to that of Gallager [16]. Other minor variants of this model have been proposed, e.g., there is no receiver P_0, and the goal of the computation is for all of the processors to learn the correct value of $f(x)$. These differences are not significant, as protocols in one model can easily and efficiently be simulated in another.

This model enforces certain properties typically required of communication protocols in noisy environments. First, protocols must be *oblivious*: the sequence of processors who broadcast is fixed in advance and does not depend on the execution. Without this requirement, noise could lead to several processors speaking at the same time. Second, it rules out *communication by silence*: when it is the turn of a processor to speak, it must speak.

2.4. gnd-tree model. The gnd-tree model is a centralized computation model in which an algorithm seeks to evaluate f on input $x \in \{0,1\}^n$ by asking queries. The algorithm has no direct access to the input x. Instead, there is a collection $(y^\lambda : \lambda \in \Lambda)$ of independent ε-noisy copies of x; here Λ is an arbitrary index set. In each step, the algorithm is allowed to make an *arbitrary boolean-valued query* about any one of the noisy copies.

Formally, a gnd-tree algorithm is represented by a rooted labeled binary tree. Each internal node v is assigned a *copy type* $\lambda_v \in \Lambda$ and a *query function* $q_v : \{0,1\}^n \longrightarrow \{0,1\}$. The two arcs out of internal node v are labeled by 0 and 1. Each leaf v is labeled by an output value out_v.

The noisy copies $y^1, y^2, \dots$ of x determine a unique root-to-leaf path, called the *execution path*, as follows: start from the root r and follow the arc labeled by the output of q_r evaluated at noisy copy y^{λ_r} to a new node. Upon arriving at internal node v, evaluate $q_v(y^{i_v})$ and follow the indicated arc. The output of the computation is equal to the output value out_v labeling the leaf.

We also consider randomized gnd-trees. For our purposes, a randomized gnd-tree is simply a probability distribution over gnd-trees.

2.5. Proof of Theorem 1 via reduction. Let us now state a reduction theorem between the noisy broadcast model and the gnd-tree model, and show how it can be used to deduce Theorem 1 from Theorem 3.

To state the reduction theorem we need some notation. For $K \subseteq [n]$, we refer to a point in $\{0,1\}^K$ as a *partial assignment to* K. If ρ is a partial assignment to K and x is a partial assignment to $[n]-K$, we write $x\rho$ for the point in $\{0,1\}^n$ that agrees with ρ on K and with x on $[n]-K$.

THEOREM 6. *Let $\alpha > 1$ and $\varepsilon \in (0,1/2)$. Suppose $\mathcal{P}$ is a noisy broadcast protocol that uses k broadcasts. Then there is a subset $K \subseteq [n]$ of size at most n/α such that for any partial assignment ρ to K there is a* gnd*-tree T of depth $2k$ with input variables indexed by $[n]-K$ with the following property: for any $x \in \{0,1\}^{[n]-K}$, when T is run on x with noise parameter $\varepsilon^{\alpha k/n}$, the output distribution is exactly the same as that of the output of $\mathcal{P}$ when run on $x\rho$ with noise parameter ε.*

The proof of this theorem appears in section 6. Now we can deduce Theorem 1 from Theorem 3.

Proof of Theorem 1 *from Theorem* 3. Suppose that for some $\beta \geq 1$, A is a βn-step noisy broadcast protocol that computes id_n against ε-noise with some probability of correctness. Choosing $\alpha = 2$ in Theorem 6, we have a gnd-tree of depth $2k$ that computes $id_{\lceil n/2 \rceil}$ against $\varepsilon^{2\beta}$-noise with the same probability of correctness as A. By Theorem 3, the probability that this gnd-tree is correct is bounded above by

$$\begin{aligned} &\frac{1}{\sqrt{n}} + \frac{12\log(1/\varepsilon^{2\beta})}{\varepsilon^{4\beta}\log n}(2\beta) \\ &\leq \frac{1}{\sqrt{n}} + \frac{48\beta^2\log(1/\varepsilon)}{\varepsilon^{4\beta}\log n}. \end{aligned}$$

This completes the proof of Theorem 1. $\square$

3. Preliminaries to the proof of Theorem 3.

In this section we present some preliminary notation, conventions, and technical facts.

Some notation. We use log to denote logarithm in base 2. When n is implicit from the context, we use $\mathbf{e_i}$ to denote the point in $\{0,1\}^n$ whose ith cooridnate is 1 and whose other coordinates are 0. The point with all zero coordinates is denoted $\mathbf{0}$.

3.1. Entropy and relative entropy.

We use some basic notions from information theory. In that context, we consider terms of the form $0\log\frac{1}{0}$ or $0\log\frac{0}{0}$ to have value 0. For a random variable X taking values in a finite set A, the *binary entropy* of X is given by

$$\mathbb{H}[X] := \sum_{x\in A} \Pr[X=x]\log\frac{1}{\Pr[X=x]}. \quad (1)$$

If another random variable Y, taking values in a finite set B, is defined over the same probability space as X, the *conditional entropy* of X given Y is defined by

$$\mathbb{H}[X|Y] := \sum_{y\in B} \Pr[Y=y]\mathbb{H}[X|Y=y]. \quad (2)$$

For two probability measures p and q defined over a finite set A, the *relative entropy* between p and q, denoted $\mathsf{D}(p\|q)$, is defined by

$$\mathsf{D}(p\|q) := \sum_{x\in A} p(x)\log\frac{p(x)}{q(x)}$$

(if $q(x) = 0$ for an x where $p(x) \neq 0$, the relative entropy is infinite). Entropy and relative entropy satisfy the following inequalities (see, e.g., [5]).

Fact 7. *For X, p, q, and A as above, we have*

$$\mathsf{D}(p\|q) \geq 0,$$

$$\mathbb{H}[X] \leq \log|A|.$$

3.2. Some estimates for logarithms. The Taylor series

$$\ln(1+x) = \sum_{n\geq 1} \frac{(-1)^{n+1}x^n}{n},$$

which is valid for $x \in (-1,1)$, implies that the following functions are continuous and differentiable for all $x > -1$:

$$a(x) = \begin{cases} \frac{\ln(1+x)}{x} & \text{if } x \neq 0, \\ 1 & \text{if } x = 0, \end{cases} \tag{3}$$

$$b(x) = \begin{cases} \frac{\ln(1+x)-x}{x^2} & \text{if } x \neq 0, \\ -\frac{1}{2} & \text{if } x = 0. \end{cases} \tag{4}$$

Proposition 8. *The function a is positive and decreasing on $(-1,\infty)$, and the function b is negative and increasing on $(-1,\infty)$.*

Proof. For $x \in (-1,0) \cup (0,\infty)$, $a'(x) = \frac{g(x)}{x^2}$, where $g(x) = \frac{x}{1+x} - \ln(1+x)$. To prove the statement concerning the function a, it thus suffices to show that $g(x) \leq 0$ for $x > -1$. This follows from the fact that $g(0) = 0$, and that $g'(x) = \frac{-x}{(1+x)^2}$, which is positive for $x < 0$ and negative for $x > 0$.

Next we prove the statement about the function b. Since $x > \ln(1+x)$ for all $x \in (-1,0) \cup (0,\infty)$, it follows that $b(x) < 0$ for all $x \in (-1,\infty)$. To show that b is increasing, we show that b' is nonnegative. Using the Taylor expansion of $\ln(1+x)$ around 0, we have that for $x \in (-1,1)$,

$$b'(x) = \sum_{n=0}^{\infty} (-1)^n \frac{n+1}{n+3} x^n.$$

Thus $b'(0) = \frac{1}{3}$, and for $x < 0$, each term in the second sum is positive so that $b'(x) > 0$. For $x > 0$, it suffices to show that the numerator of $b'(x) = (x^2 + 2x - 2(1+x)\ln(1+x))/x^3(1+x)$ is positive. The numerator is 0 at $x = 0$ and has derivative $2(x - \ln(1+x))$, which is positive for all $x > 0$. □

Corollary 9. *Let $\varepsilon \in (0, \frac{1}{2})$ and $x \geq \varepsilon - 1$. Then*

1. $0 < a(x) \leq 2\ln(1/\varepsilon)$,
2. $0 > b(x) \geq -2\ln(1/\varepsilon)$.

Proof. Using Proposition 8 and $\varepsilon \leq 1/2$ we have

$$0 < a(x) \leq a(\varepsilon - 1) = \ln(1/\varepsilon)/(1-\varepsilon) \leq 2\ln(1/\varepsilon),$$

and thus the first part holds.

As for the second part, $0 > b(x)$ follows from Proposition 8. Also, since b is negative and increasing, it remains to show that $|b(\varepsilon - 1)| \leq 2\ln(1/\varepsilon)$. We consider two cases: if $\varepsilon \leq 1 - 1/\sqrt{2}$, then it is easy to see that

$$|b(\varepsilon - 1)| \leq \frac{\ln(1/\varepsilon)}{(1-\varepsilon)^2} \leq 2\ln\left(\frac{1}{\varepsilon}\right).$$

If $\varepsilon > 1 - 1/\sqrt{2}$, since a is decreasing we have $a(\varepsilon - 1) \leq a(-1/\sqrt{2})$ which by direct computation is less than 2. From this and $\varepsilon \leq 1/2$, we get

$$|b(\varepsilon - 1)| = \frac{a(\varepsilon-1)-1}{1-\varepsilon} = \frac{2a(\varepsilon-1)-2}{2(1-\varepsilon)} \leq \frac{a(\varepsilon-1)}{2(1-\varepsilon)} = \frac{\ln(1/\varepsilon)}{2(1-\varepsilon)^2} \leq 2\ln\left(\frac{1}{\varepsilon}\right). \qquad \square$$

PROPOSITION 10. *Let m be defined on $[0, 1]$ by*

$$m(x) = \begin{cases} x^2 \ln(1/x) & \text{if } x \neq 0, \\ 0 & \text{if } x = 0. \end{cases}$$

Then $m(x) \geq m(1 - x)$ for all $x \in [\frac{1}{2}, 1]$.

Proof. Let $d(x) = m(x) - m(1 - x)$ for $x \in [1/2, 1]$. Then d is continuous and twice differentiable on $[1/2, 1]$ with second derivative given by $d''(x) = 2(\ln((1-x)/x))$. Since this is negative for all $x \in (1/2, 1)$, and $m(1/2) = m(1) = 0$, we conclude that $m(x) \geq 0$ for $x \in [1/2, 1]$. $\square$

PROPOSITION 11. *For $x, y \in [0, 1]$, $x \log(x/y) + (1 - x) \log((1 - x)/(1 - y))$ is nonnegative.*

This a special case of the first assertion in Fact 7, where p and q are distributions over a two-point space, but we provide a direct proof for completeness.

Proof. Hold x fixed; it suffices to show that as a function of y, $g(y) = x \log y + (1 - x) \log(1 - y)$ has a maximum at $y = x$. Taking the derivative with respect to y and simplifying yields $g'(y) = (x - y)/y(1 - y)$, which is positive for $0 < y < x$ and negative for $1 > y > x$, implying g attains a maximum at $y = x$. $\square$

3.3. Tail bounds for sums of noise bits. We will need a standard type of tail bound for sums of noise bits.

LEMMA 12. *Let N be an ε-noisy n-vector. For any nonzero vector $\alpha \in \mathbb{R}^n$, and $t > 0$, we have*

$$\Pr_N\left[\sum_i \alpha_i(N_i - \varepsilon) > t\right] \leq e^{-2t^2/\|\alpha\|^2}.$$

Since we do not know a reference for this version of the bound, we provide a proof.

Proof. For any positive λ,

$$\begin{aligned} \Pr_N\left[\sum_i \alpha_i(N_i - \epsilon) > t\right] &\leq e^{-\lambda t}\mathbb{E}[e^{\lambda \sum_i \alpha_i(N_i-\varepsilon)}] \\ &= e^{-\lambda t}\prod_i \mathbb{E}[e^{\lambda\alpha_i(N_i-\varepsilon)}] = e^{-\lambda t}\prod_i \left(\varepsilon e^{\lambda\alpha_i(1-\varepsilon)} + (1-\varepsilon)e^{-\lambda\alpha_i\varepsilon}\right) \\ &\leq e^{-\lambda t}\prod_i e^{\alpha_i^2\lambda^2/8} \end{aligned}$$

(since for $p \in [0, 1]$, $pe^{\gamma(1-p)} + (1 - p)e^{-\gamma p} \leq e^{\gamma^2/8}$ (see Lemma A.1.6 of [3]))

$$= e^{-\lambda t + \lambda^2\|\alpha\|^2/8}.$$

Choosing $\lambda = 4t/\|\alpha\|^2$ yields the bound of the lemma. $\square$

4. Proof of Theorem 3. In this section we prove Theorem 3. For the rest of this section, fix T to be a gnd-tree that supposedly computes id for inputs $x \in \{0,1\}^n$. The *success probability* of T is the probability that T gives the correct output when run on a uniformly random input. Our goal is to give an upper bound on the success probability of T in terms of its depth $\mathsf{depth}(T)$, n, and the noise parameter ε. In the case that T is randomized, i.e., a probability distribution over deterministic gnd-trees, its success probability is the average (with respect to this distribution) of the success probabilities of various deterministic trees. Thus, any upper bound for deterministic trees extends to randomized trees.

Therefore, without loss of generality, we assume that T is deterministic.

The vertices of T. Let V be the set of vertices of T. Let Λ be the set of indices of the noisy copies of the input used by T. For an index $\lambda \in \Lambda$, V^λ will denote the set of (internal) vertices of T that query the noisy copy indexed by λ.

The execution space Υ. For noise parameter $\varepsilon \in (0, 1/2)$ we define the *ε-execution space* $\Upsilon = \Upsilon(T, \varepsilon)$ of T to be the probability space corresponding to the choice of a uniformly random input $X \in \{0,1\}^n$ and an indexed family of independent ε-noise n-vectors $N^\Lambda = (N^\lambda : \lambda \in \Lambda)$. For every $\lambda \in \Lambda$, Y^λ denotes the ε-noisy copy of X defined by $Y^\lambda = X \oplus N^\lambda$. $Y^\Lambda = (Y^\lambda : \lambda \in \Lambda)$ denotes the indexed family of all noisy copies.

We say that T is *executed on* Υ if it is run with input X and has access to Y^Λ as the noisy copies of the input. Note that when T is executed on Υ, the unique root-to-leaf execution path is a random variable that is completely determined by Y^Λ. We define the following random variables and events over the execution space:

- Let Π denote the leaf that terminates the execution path.
- Let $\mathsf{success}$ be the event that T correctly outputs the input X.
- For a vertex v, let $\mathsf{vis}(v)$ denote the event that v lies on the execution path. In this case we say that T *visits* v.
- When the meaning is clear from context, the event $X = x$ (for $x \in \{0,1\}^n$) is abbreviated as x, especially when writing the probability of events. For example, we may write $\Pr[x]$ instead of $\Pr[X = x]$ and $\Pr[\mathsf{success}|x]$ instead of $\Pr[\mathsf{success}|X = x]$.

The progress function. As with many lower bound proofs, our proof proceeds by defining a function that measures the progress of the computation towards its goal. We show that (1) for T to succeed with high probability, the expected value of the progress measure at the final leaf must be large; and (2) the aforementioned expected value is bounded by a function of $\mathsf{depth}(T)$, and thus for it to be large the depth of T must be large as well.

For a vertex v in T and an input vector $x \in \{0,1\}^n$ we define the following functions. The final function L serves as our progress measure:

$$\begin{aligned} p(v,x) &= \Pr\left[\mathsf{vis}(v)|X = x\right], \\ L_i(v,x) &= \log\left(\frac{p(v,x)}{p(v, x \oplus \mathbf{e_i})}\right) \qquad \text{for } i \in \{1,\dots,n\}, \\ \text{and} \quad L(v,x) &= \frac{1}{n}\sum_{i=1}^{n} L_i(v,x). \end{aligned} \tag{5}$$

Some intuition. Theorem 3 gives an upper bound on the success probability of T in terms of its depth or, equivalently, on the number of queries it makes. Let us consider the most interesting case, where the success probability is at least some constant, say $1/2$. In this case the theorem gives a lower bound of $\Omega(n \log n)$ on the number of queries.

To prove this lower bound using a progress measure, we want to show that in order to succeed with high probability, the expectation of the progress measure achieved by T at the completion of its computation must be higher than some threshold τ. We then want to show (roughly) that the progress measure achieved by any gnd-tree making $o(n \log n)$ queries is smaller than τ.

So how does one measure the progress made by T after the computation has reached a vertex v? One natural choice would be to look at $\mathbb{H}[X] - \mathbb{H}[X|\mathsf{vis}(v)]$, the reduction in entropy of the random input X provided by knowing $\mathsf{vis}(v)$. Initially, at the root, this is equal to 0, and it is not hard to show that for any gnd-tree that outputs X correctly with probability $1/2$ the expectation of this expression must be at least $\Omega(n)$. To prove our result, we would like to show that $\Omega(n \log n)$ queries are needed to reduce the entropy of X by $\Omega(n)$, but this is not true: if we query all n coordinates of a single noisy copy, then the expected reduction of entropy in X is already $\Omega(n)$.

The reason for this is that querying each coordinate once already gives us, with high probability, a guess for X which is correct on roughly $(1 - \varepsilon)$-fraction of coordinates. To go from that to getting all the coordinates right takes many more queries. Indeed, it turns out that in a sense, distinguishing between X and its neighbors (namely from inputs that differ from it on a singly coordinate) is as hard as figuring out all the bits of X from scratch. But although after all coordinates have been queried once, making further queries will provide less of an entropy reduction, this behavior is difficult to capture and use, since to show that the entropy reduction is slow one must use the fact that certain information about X is already known. Furthermore, other algorithms which perform different queries may have a different pattern of entropy reduction.

The progress measure L turns out to be more useful. For example, it behaves very nicely when we restrict ourselves to coordinate queries of noisy copies of X. Indeed, the change in L due to the response to a coordinate query in a noisy copy of X is context independent, i.e., does not depend on the answers to any of the previous queries. (The proof of this is easy, but since we do not need it explicitly, it is omitted.) A single coordinate query to bit i changes L_i by $\Theta(1)$ while L_j is unchanged, and so L changes by at most $O(1/n)$. It is easy to show that the final value of L must have expectation at least $\Omega(\log n)$, and thus using our progress measure it is relatively easy to get an $\Omega(n \log n)$ lower bound on the depth of a tree computing id and making only coordinate queries. In the case of more general queries we do not have context independence in progress measure gain, but its behavior is still nice enough for us to control its increase as a function of the depth of T.

By rewriting the expression for L we can get further intuition. Since x is uniformly distributed, we have $\Pr[x] = \Pr[x \oplus \mathbf{e_i}]$, and thus by standard laws of conditional probability the functions L_i can be written as

$$L_i(v, x) = \log\bigl(\Pr[x|\mathsf{vis}(v)]\bigr) - \log\bigl(\Pr[x \oplus \mathbf{e_i}|\mathsf{vis}(v)]\bigr). \tag{6}$$

The term $\Pr[y|\mathsf{vis}(v)]$ measures the conditional probability of the event $[X = y]$, given that T was run on the random input X and has reached the node v in the course of computation. It is therefore the "perceived probability" that T assigns to y being the value of the input, when it reaches v during the computation. Thus $L_i(v, x)$ compares the perceived probabilities of x and its neighbor $x \oplus \mathbf{e_i}$ and thus measures how well the computation has distinguished x from $x \oplus \mathbf{e_i}$, given that it has arrived at v. $L(v,x)$ gives an aggregate measure of how well the computation has distinguished x from all of its neighbors. Note that while it is also important that the computation distinguishes x from points other than its neighbors, L does not take this into account directly. Intuitively, the neighbors are the "hardest" points to distinguish x from, and so it is enough for L to consider only these.

Let us now get a rough estimate for the change in L at the completion of a successful algorithm when the true input is x. At the beginning of the computation, when v is the root, $L(v,x)$ is obviously zero. At the end of the computation, when T reaches a leaf π, we expect that T should assign a high perceived likelihood to x, say $\Omega(1)$. On the other hand, since the sum of the perceived likelihoods of all points is 1, the average perceived likelihoods of the neighbors of x is at most $\frac{1}{n}$, and the concavity of the logarithm yields that $L(\pi,x) \geq \log n - O(1)$.

A coarse intuition for the proof of Theorem 3 is that we bound the expected gain to L obtained from each query by $O(\frac{1}{n})$. Since the typical value of $L(\Pi,X)$ is roughly $\log n$ (this is the value of L at the leaf reached by the computation and the actual input on which T is run), it follows that at least $\Omega(n \log n)$ queries are typically required.

As noted above, if we restrict queries to noisy copies of individual input bits, it is easy to show that each query changes L by at most $O(\frac{1}{n})$. When we turn to general queries, this is no longer true; indeed, one can construct situations where a single general query changes L by much more. Nevertheless, we are able to formulate and prove a weaker statement (Lemma 14) that suffices for our bounds.

The relation to relative entropy. We note that the functions L_i and L are related to relative entropies. For example, looking at (5) one observes that $\mathbb{E}\big[L_i(\Pi,X) \bigm| x\big]$ is the relative entropy between the distribution of the leaf reached when T is run on x and that of the leaf reached when T is run on the ith neighbor of x.

How we proceed. The next two lemmas formalize the above intuition and imply Theorem 3 immediately. Lemma 13 states that for T to succeed with constant probability, the expected value of the progress measure at Π should be at least logarithmic in n; and Lemma 14 shows that each level of depth in T contributes at most $O(\frac{1}{n})$ to the expectation of the progress measure at Π (this is shown to be true not just for a random input but even if the input is arbitrarily fixed). The proof of Theorem 3 will be completed once we prove those lemmas. The relatively simple proof of Lemma 13 appears below. The proof of Lemma 14, which is considerably more involved, spans through the rest of this section.

LEMMA 13. *Let T be a* gnd*-tree with inputs in $\{0,1\}^n$. Then when T is executed over $\Upsilon(T,\varepsilon)$,*

$$\Pr[\mathsf{success}] \leq \frac{2}{\log n} \cdot \mathbb{E}\Big[|L(\Pi,X)|\Big] + \frac{1}{\sqrt{n}}.$$

LEMMA 14. *Let T be a* gnd*-tree with inputs in $\{0,1\}^n$. When T is run over $\Upsilon(T,\varepsilon)$, it holds for every $x \in \{0,1\}^n$ that*

$$\mathbb{E}\Big[|L(\Pi,X)| \Bigm| x\Big] \leq \frac{3\log(1/\varepsilon)}{\varepsilon^2} \cdot \frac{\mathsf{depth}(T)}{n} + \frac{5\log(1/\varepsilon)}{\varepsilon} \cdot \sqrt{\frac{\mathsf{depth}(T)}{n}}.$$

Proof of Theorem 3. Since the bound on conditional expectation in Lemma 14 holds for each x, it holds for the deconditioned expectation. Furthermore, since

$\varepsilon < 1/2$, we can use $5 < 3/\varepsilon$ to get

$$\mathbb{E}\Big[|L(\Pi, X)| \mid x\Big] \leq \frac{\log(1/\varepsilon)}{\varepsilon^2}\left(3\frac{\mathsf{depth}\,T}{n} + 3\sqrt{\frac{\mathsf{depth}\,T}{n}}\right).$$

Substituting this into Lemma 13 yields the theorem.

Proof of Lemma 13. We first show that for any leaf π and input x,

$$\Pr[x \wedge \pi] \leq \Pr[\pi]\frac{2^{L(\pi,x)}}{n}. \tag{7}$$

Taking logs, it suffices to show that

$$L(\pi, x) \geq \log(\Pr[\pi \wedge x]) - \log\left(\frac{1}{n}\Pr[\pi]\right). \tag{8}$$

Using (5), the fact that $\Pr[x] = \Pr[x \oplus \mathbf{e_i}]$, and the convexity of the logarithm function, we have

$$\begin{aligned} L(\pi, x) &= \frac{1}{n}\sum_{i=1}^{n}\big[\log(\Pr[\pi \wedge x]) - \log(\Pr[\pi \wedge (x \oplus \mathbf{e_i})])\big] \\ &= \log(\Pr[\pi \wedge x]) - \frac{1}{n}\sum_{i=1}^{n}\log(\Pr[\pi \wedge (x \oplus \mathbf{e_i})]) \\ &\geq \log(\Pr[\pi \wedge x]) - \log\left(\frac{1}{n}\sum_{i=1}^{n}\Pr[\pi \wedge (x \oplus \mathbf{e_i})]\right) \\ &\geq \log(\Pr[\pi \wedge x]) - \log\left(\frac{1}{n}\Pr[\pi]\right), \end{aligned}$$

as required to prove (8) and (7).

Now let $A(\pi, x)$ denote the condition $L(\pi, x) < \frac{1}{2}\log(n)$, and let $\bar{A}(\pi, x)$ denote the complementary condition. We have

$$\Pr[\mathsf{success}] \leq \Pr[\bar{A}(\Pi, X)] + \Pr[\mathsf{success} \wedge A(\Pi, X)]. \tag{9}$$

Using the general upper bound $\mathbb{E}[|Z|] \geq B\Pr[Z \geq B]$ for any random variable Z and positive real B, the first term of (9) satisfies

$$\Pr[\bar{A}(\Pi, X)] \leq \frac{2}{\log n}\mathbb{E}[|L(\Pi, X)|].$$

To bound the second term, let $\mathsf{out}(x)$ denote the set of leaves in T that output x. Using (7) we have

$$\begin{aligned} \Pr[\mathsf{success} \wedge A(\Pi, X)] &\leq \sum_{x}\sum_{\{\pi \in \mathsf{out}(x):\ A(\pi,x)\}}\Pr[\pi \wedge x] \\ &\leq \sum_{x}\sum_{\{\pi \in \mathsf{out}(x):\ A(\pi,x)\}}\frac{1}{\sqrt{n}}\Pr[\pi]. \end{aligned}$$

Since each leaf π belongs to exactly one set $\mathsf{out}(x)$ and $\sum_{\pi}\Pr[\pi] = 1$, this is less than or equal to $\frac{1}{\sqrt{n}}$. □

4.1. Proof of Lemma 14: Notation.

The remainder of this section is devoted to the proof of Lemma 14. For brevity we fix ε for the rest of the section, and work over the execution space Υ of T, which we also refer to as simply the execution space. Also, note that while Lemma 14 is stated for every $x \in \{0,1\}^n$, it suffices (by symmetry) to prove it in the case $x = \mathbf{0}$. We begin by defining some notation.

Restriction to $x = \mathbf{0}$. Since we prove Lemma 14 only for the case $x = \mathbf{0}$, we can use the following simplifications in our notation. For any $i \in [n]$, define

$$\begin{aligned} p_0(v) &= p(v,\mathbf{0}) = \Pr[\mathsf{vis}(v)|X=\mathbf{0}],\\ p_i(v) &= p(v,\mathbf{e_i}) = \Pr[\mathsf{vis}(v)|X=\mathbf{e_i}],\\ L_i(v) &= L_i(v,\mathbf{0}) = \log\left(\frac{p_0(v)}{p_i(v)}\right),\\ L(v) &= L(v,\mathbf{0}) = \frac{1}{n}\sum_i L_i(v). \end{aligned} \tag{10}$$

Progress measure for events. Let us now extend the above notation for any event A defined over the execution space. We denote

$$\begin{aligned} p_0[A] &= \Pr[A|X=\mathbf{0}],\\ p_i[A] &= \Pr[A|X=\mathbf{e_i}],\\ L_i(A) &= \log\left(\frac{p_0(A)}{p_i(A)}\right),\\ L(A) &= \frac{1}{n}\sum_i L_i(A). \end{aligned}$$

Progress measures for variables. Given any function F, such as L and L_i, that maps events in the execution space to real numbers, we define an operator $\tilde{F}$ that maps any random variable Z over the execution space to a real-valued random variable over the execution space. Let Z be a random variable taking on values from S. Viewing Z as a function from the execution space to S, we have that for each $s \in S$, $Z^{-1}(s)$ is an event. We define $\tilde{F}(Z)$ to be the real-valued random variable that gets value $F(Z^{-1}(s))$ when Z gets value s. We abuse notation by omitting the "$\sim$" and simply writing $F(Z)$.

Intuitively, the value of the random variable $L_i(Z)$ indicates how helpful the observed value of Z is for distinguishing between the zero input from $\mathbf{e_i}$. The value of $L(Z)$ indicates how well the observed value of Z is for distinguishing the zero input from a randomly chosen neighbor.

In the case that Z is the random variable Π, the notation $L(\Pi)$ as just defined coincides with the definition we get by using the progress measure for vertices defined in (10) and evaluating it at the random variable Π.

We define zero-conditioned expectation and entropy by

$$\begin{aligned} \mathbb{E}_0[Z] &= \mathbb{E}[Z|X=\mathbf{0}],\\ \mathbb{H}_0[Z] &= \mathbb{H}[Z|X=\mathbf{0}]. \end{aligned} \tag{11}$$

The goal of proving Lemma 14, formalized using the new notation, is to show that

$$\mathbb{E}_0\Big[|L(\Pi)|\Big] \le \frac{3\log(1/\varepsilon)}{\varepsilon^2}\cdot\frac{\mathsf{depth}(T)}{n} + \frac{5\log(1/\varepsilon)}{\varepsilon}\cdot\sqrt{\frac{\mathsf{depth}(T)}{n}}. \tag{12}$$

4.2. Analyzing a single event involving one noisy copy. To prove (12) we need to understand the behavior of $L(Z)$ for certain random variables Z (particularly, for the random variable Π). We begin by understanding its behavior on particularly simple events and variables.

Definition 15 (λ-events and λ-variables). *For an index $\lambda \in \Lambda$,*

(i) *an event over the execution space that depends only on Y^{λ} is called a* λ-event*;*

(ii) *a random variable that depends only on Y^{λ} is called a* λ-variable.

In this subsection we obtain bounds on $L(A)$ and $(L(A))^2$ for λ-events A, in terms of ε, $p_0[A]$, and the quantities $\delta_i[A]$ defined by $\delta_i[A] = p_i[A] - p_0[A]$. We then derive analogous bounds for expectations of λ-variables.

In the remainder of this subsection we use A to denote a λ-event, and for simplicity we often denote

$$\begin{aligned} p_0 &= p_0[A],\\ p_i &= p_i[A],\\ \delta_i &= p_i - p_0,\\ L_i &= \log(p_0/p_i) \end{aligned}$$

when A is implicit from the context.

The following is an easy bound on $p_i[A]/p_0[A]$ in terms of ε.

Lemma 16. *For any λ-event A, for all $i \in [n]$,*

$$\frac{\varepsilon}{1-\varepsilon} \le \frac{p_0[A]}{p_i[A]} \le \frac{1-\varepsilon}{\varepsilon}.$$

Proof. We have

$$\begin{aligned} p_0 &= \sum_{a\in A} p_0[Y^{\lambda} = a],\\ p_i &= \sum_{a\in A} p_i[Y^{\lambda} = a], \end{aligned}$$

and therefore

$$\min_{a\in A} \frac{p_0[Y^{\lambda} = a]}{p_i[Y^{\lambda} = a]} \le \frac{p_0}{p_i} \le \max_{a\in A} \frac{p_0[Y^{\lambda} = a]}{p_i[Y^{\lambda} = a]}.$$

Since

$$\frac{p_0[Y^{\lambda} = a]}{p_i[Y^{\lambda} = a]} = \begin{cases} \frac{1-\varepsilon}{\varepsilon} & \text{if } a_i = 0,\\ \frac{\varepsilon}{1-\varepsilon} & \text{if } a_i = 1, \end{cases}$$

the conclusion of the lemma follows. □

To prove the bounds on $L(A)$ and $L(A)^2$, we need the following upper bound for $\sum_i \delta_i{}^2$.

Lemma 17. *For any λ-event A,*

$$\sum_{i\in[n]} \delta_i^2 \le \frac{2}{\varepsilon^2} p_0^2 \ln\left(\frac{1}{p_0}\right) \tag{13}$$

(where, as usual, $p_0^2 \ln(1/p_0)$ is defined to be 0 for $p_0 = 0$.)

Proof. Since A depends only on Y^λ, A can be identified with the subset of $\{0,1\}^n$ of values for Y^λ that imply the event A. The bound of the lemma will be obtained by interpreting the quantities δ_i as *biased Fourier coefficients* (see [31]) of the characteristic function of A. For this we need some definitions and facts.

Biased Fourier transform. Let F denote the inner product space consisting of functions mapping $\{0,1\}^n$ to $\mathbb{R}$ with the inner product

$$\langle g, h\rangle = \mathbb{E}_N[g(N)h(N)],$$

where N is an ε-noise vector of length n, defined in section 2.1.

For $S \subseteq [n]$, the biased Fourier character $\chi_S \in F$ is defined by

$$\chi_S(x) = \prod_{i\in S} \frac{x_i - \varepsilon}{\sqrt{\varepsilon(1-\varepsilon)}}.$$

In particular $\chi_\emptyset$ is identically 1. For $f \in F$, the ε-biased fourier transform of f is the function $\hat{f}$ mapping subsets of $[n]$ to $\mathbb{R}$ defined for $S \subseteq [n]$ by

$$\hat{f}(S) = \langle f, \chi_S\rangle.$$

For ease of notation, we use $\hat{f}(i)$ instead of $\hat{f}(\{i\})$ and χ_i for $\chi_{\{i\}}$.

It is easily verified that $\{\chi_S : S \subseteq [n]\}$ is an orthonormal basis of F. It follows that for any $f \in F$

$$f = \sum_S \hat{f}(S)\chi_S$$

and that for any $g, h \in F$

$$\langle g, h\rangle = \sum_S \hat{g}(S)\hat{h}(S). \tag{14}$$

For $i \in [n]$, let T_i denote the linear transformation on F defined by $T_i f(x) = f(x \oplus \mathbf{e_i})$. Define the constant $d = \frac{1-2\varepsilon}{\sqrt{\varepsilon(1-\varepsilon)}}$. By direct computation, one obtains

$$\langle T_i\chi_S, \chi_\emptyset\rangle = \begin{cases} 1 & \text{if } S = \emptyset, \\ d & \text{if } S = \{i\}, \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

We now return to the proof of Lemma 17. Let f be the characteristic function of the set A. Using the definitions and the facts above we have

$$\begin{aligned} p_0 &= \langle f, \chi_\emptyset\rangle, \\ p_i &= \langle T_i f, \chi_\emptyset\rangle = \sum_S \hat{f}(S)\langle T_i(\chi_S), \chi_\emptyset\rangle = \hat{f}(\emptyset) + d\hat{f}(i), \\ \delta_i &= d\hat{f}(i). \end{aligned}$$

We thus have that $\sum_i \delta_i^2 = d^2 \sum \hat{f}(i)^2$. The proof of Lemma 17 will therefore be completed once we prove the following lemma.

Lemma 18. *Let $f : \{0,1\}^n \longrightarrow [0,1]$ be any function. Then*

$$\sum \hat{f}(i)^2 \le \frac{2}{\varepsilon(1-\varepsilon)} \hat{f}(\emptyset)^2 \ln\left(\frac{1}{\hat{f}(\emptyset)}\right).$$

Lemma 18 is a generalization to the biased case of a lemma from [32].

Proof. Let $S(f) = \sum \hat{f}(i)^2$, and let $g \in F$ be the function $g = \sum_i \hat{f}(i)\chi_i$. By (14),

$$S(f) = \langle f, g\rangle.$$

Defining $\ell \in F$ by $\ell(x) = \sum_i \hat{f}(i)x_i$ and $\mu = \mathbb{E}_N[\ell(N)] = \varepsilon \sum_i \hat{f}(i)$, we have that for all x,

$$g(x) = \frac{\ell(x) - \mu}{\sqrt{\varepsilon(1-\varepsilon)}}.$$

Consequently, we have from Lemma 12 that

$$\Pr[g(N) \ge s] = \Pr[\ell(x) - \mu \ge s\sqrt{\varepsilon(1-\varepsilon)}] \le e^{-2\varepsilon(1-\varepsilon)s^2/S(f)}.$$

Since $f(x) \in [0,1]$ for every x, it holds for any positive parameter t that

$$\begin{aligned}
S(f) &= \langle f, g\rangle \\
&\le t\mathbb{E}_N[f(N)] + \mathbb{E}_N[(g(N) - t)\mathbf{1}_{\{g(N)>t\}}] \\
&= t\hat{f}(\emptyset) + \int_{s=0}^{\infty} \Pr[g(N) - t > s]\mathrm{d}s \\
&= t\hat{f}(\emptyset) + \int_{s=t}^{\infty} \Pr[g(N) > s]\mathrm{d}s \\
&\le t\hat{f}(\emptyset) + \int_{s=t}^{\infty} e^{-2\varepsilon(1-\varepsilon)s^2/S(f)}\mathrm{d}s \\
&\le t\hat{f}(\emptyset) + \frac{1}{t}\int_{s=t}^{\infty} se^{-2\varepsilon(1-\varepsilon)s^2/S(f)}\mathrm{d}s \\
&= t\hat{f}(\emptyset) + \frac{S(f)}{4\varepsilon(1-\varepsilon)t}e^{-2\varepsilon(1-\varepsilon)t^2/S(f)}.
\end{aligned}$$

Choosing $t = \frac{\sqrt{S(f)}}{\sqrt{2\varepsilon(1-\varepsilon)}}\sqrt{\ln(1/\hat{f}(\emptyset))}$ in the last expression yields

$$S(f) \le \frac{\hat{f}(\emptyset)\sqrt{S(f)}}{\sqrt{2\varepsilon(1-\varepsilon)}}\left[\sqrt{\ln\left(\frac{1}{\hat{f}(\emptyset)}\right)} + \frac{1}{2\sqrt{\ln(1/\hat{f}(\emptyset))}}\right].$$

Now assume that $\hat{f}(\emptyset) \le 1/2$. In this case the second summand is bounded above by the first, and we get

$$S(f) \le \sqrt{\frac{2S(f)}{\varepsilon(1-\varepsilon)}}\hat{f}(\emptyset)\sqrt{\ln\left(\frac{1}{\hat{f}(\emptyset)}\right)},$$

which implies the desired inequality

$$S(f) \le \frac{2}{\varepsilon(1-\varepsilon)}\hat{f}(\emptyset)^2 \ln\left(\frac{1}{\hat{f}(\emptyset)}\right).$$

Now if $\hat{f}(\emptyset) > 1/2$ (it always holds that $\hat{f}(\emptyset) = \mathbb{E}[f] \leq 1$), take $h = 1 - f$. Then $\hat{h}(\emptyset) \leq 1/2$, and we have

$$S(f) = S(h) \leq \frac{2}{\varepsilon(1-\varepsilon)} \left(\hat{h}(\emptyset)^2 \ln \left(\frac{1}{\hat{h}(\emptyset)} \right) \right). \tag{16}$$

Since $\hat{f}(\emptyset) = 1 - \hat{h}(\emptyset)$, we may, using Proposition 10, replace $\hat{h}(\emptyset)$ by $\hat{f}(\emptyset)$ in (16). This completes the proof of Lemma 18. □

This also completes the proof of Lemma 17. □

We are now ready to state and prove the bound for λ-events.

LEMMA 19. *Let A be a λ-event. Then*

$$L\left(A\right) \leq \frac{3\log(1/\varepsilon)}{\varepsilon^2 n} \log\left(\frac{1}{p_0}\right) - \frac{\log e}{p_0 n} \sum_{i=1}^{n} \delta_i, \tag{17}$$

$$L^2\left(A\right) \leq \frac{6(\log(1/\varepsilon))^2}{\varepsilon^2 n} \log\left(\frac{1}{p_0}\right). \tag{18}$$

Proof. By definition, $L\left(A\right) = -\frac{1}{n}\sum_{i=1}^{n} \log\left(1 + \frac{\delta_i}{p_0}\right) = -\frac{\log e}{n}\sum_{i=1}^{n} \ln\left(1 + \frac{\delta_i}{p_0}\right)$. To obtain (17) we rewrite the expression for $L\left(A\right)$ in terms of the function b defined in (4), and use the second part of Corollary 9 together with the inequality $p_i/p_0 \geq \varepsilon$, and then use Lemma 17 to get

$$\begin{aligned} L\left(A\right) &= \frac{-\log e}{n}\left[\sum_{i=1}^{n}\left(\frac{\delta_i}{p_0}\right)^2 b\left(\frac{\delta_i}{p_0}\right) + \sum_{i=1}^{n}\frac{\delta_i}{p_0}\right] \\ &\leq \frac{\log e}{n}\left[\frac{2\ln(1/\varepsilon)}{p_0^2}\sum_{i=1}^{n}\delta_i^2 - \frac{1}{p_0}\sum_{i=1}^{n}\delta_i\right] \\ &\leq \frac{3\log(1/\varepsilon)}{\varepsilon^2 n}\log\left(\frac{1}{p_0}\right) - \frac{\log e}{p_0 n}\sum_{i=1}^{n}\delta_i. \end{aligned}$$

For (18) we rewrite the expression for $L\left(A\right)^2$ in terms of the function a defined in (3) and obtain the chain of inequalities

$$\begin{aligned} L\left(A\right)^2 &= \left(\frac{\log e}{n}\sum_{i=1}^{n}\left(\frac{\delta_i}{p_0}\right) a\left(\frac{\delta_i}{p_0}\right)\right)^2 \\ &\overset{(1)}{\leq} \left(\frac{2\log(1/\varepsilon)}{p_0}\frac{1}{n}\sum_{i=1}^{n}\delta_i\right)^2 \overset{(2)}{\leq} \left(\frac{2\log(1/\varepsilon)}{p_0}\sqrt{\frac{1}{n}\sum_{i=1}^{n}\delta_i^2}\right)^2 \\ &\leq \frac{4(\log(1/\varepsilon))^2}{p_0^2 n}\sum_{i-1}^{n}\delta_i^2 \\ &\overset{(3)}{\leq} \frac{4(\log(1/\varepsilon))^2}{p_0^2 n}\,\frac{2p_0^2\log(1/p_0)}{\varepsilon^2\log e} \leq \frac{6(\log(1/\varepsilon))^2}{\varepsilon^2 n}\log\left(\frac{1}{p_0}\right). \end{aligned}$$

Here (1) comes from the first part of Corollary 9, (2) follows from the Cauchy–Schwarz inequality, and (3) is obtained from Lemma 17. □

We now deduce an analogue of Lemma 19 for expectations of λ-variables.

Corollary 20. *Let Z be a λ-random variable. Then*

$$\mathbb{E}_0[L\,(Z)] \le \frac{3\log(1/\varepsilon)}{\varepsilon^2 n}\mathbb{H}_0[Z]$$
$$\text{and} \qquad \mathbb{E}_0[L\,(Z)^2] \le \frac{6(\log(1/\varepsilon))^2}{\varepsilon^2 n}\mathbb{H}_0[Z].$$

Proof. In the following expressions, the index z ranges over values of Z. Using the definition of $L\,(Z)$ and applying (17) we get

$$\begin{aligned}\mathbb{E}_0[L\,(Z)] &= \sum_z p_0[Z=z]L([Z=z])\\ &\le \sum_z \frac{3\log(1/\varepsilon)}{\varepsilon^2 n} p_0[Z=z]\log\left(\frac{1}{p_0[Z=z]}\right) + \sum_z \frac{\log e}{n}\sum_{i=1}^n \delta_i[Z=z]\\ &= \frac{3\log(1/\varepsilon)}{\varepsilon^2 n}\mathbb{H}_0[Z],\end{aligned}$$

where the last transition is obtained by noting that the second double sum is 0 since for each fixed i, $\sum_z p_i[Z=z] = \sum_z p_0[Z=z] = 1$, and so $\sum_z \delta_i[Z=z] = 0$.

The bound on $\mathbb{E}_0[L\,(Z)^2]$ follows by a similar (actually simpler) calculation using (18). □

4.3. Box events. In this subsection we define box events and box random variables, which are more general than λ-events and λ-variables. We then extend the bounds from the previous section to the box case, and apply these bounds to the random variable Π, which turns out to be a box random variable.

Box events. A *box event* is an intersection of λ-events for possibly different λ's. Any box event B can be uniquely written as $B = \bigcap_{\lambda\in\Lambda} B^\lambda$, where each B^λ is a λ-event. A box *random variable* is a random variable Z for which each of the events $[Z = s]$ is a box event.

Our interest in box events stems from the following immediate fact.

Fact 21. *For each vertex v in T the event* $\mathsf{vis}(v)$ *is a box event, and therefore* Π *is a box random variable.*

Let $B = \bigcap_{\lambda\in\Lambda} B^\lambda$ be a box event. The different λ-events $(B^\lambda : \lambda \in \Lambda)$ may well be dependent, as the variables Y^λ, each being correlated with X, are dependent. We observe, however, that once we condition on a *fixed value of* X they become mutually independent. This implies the following properties for box events.

Fact 22. *Every box event B satisfies*

$$p_0[B] = \prod_{\lambda\in\Lambda} p_0[B^\lambda].$$

Moreover, for every $i \in \{1,\ldots,n\}$,

$$\begin{aligned} p_i[B] &= \prod_{\lambda\in\Lambda} p_i[B^\lambda],\\ L_i(B) &= \sum_{\lambda\in\Lambda} L_i\left(B^\lambda\right),\\ \text{and} \qquad L(B) &= \sum_{\lambda\in\Lambda} L\left(B^\lambda\right).\end{aligned} \tag{19}$$

Contribution of copies to progress. Looking at (19) one observes that for box events the contribution of each noisy copy to the progress measure can be easily singled out. For a box event B and for $\lambda \in \Lambda$ we define

$$\begin{aligned} & L_i^\lambda(B) = L_i\left(B^\lambda\right) \\ \text{and} \qquad & L^\lambda(B) = L\left(B^\lambda\right). \end{aligned}$$

In the same way that we extended L and L_i from events to random variables, we extend L^λ and L_i^λ from box events to box random variables.

Some more notation. The following notation is used for the bounds for box random variables below. Throughout this subsection we use the letter y to denote points in $(\{0,1\}^n)^\Lambda$ representing an assignment to Y^Λ (recall that Y^Λ is the set of all noisy copies of the input). We use y^λ to denote a value of Y^λ. Also, writing $\hat{\lambda}$ for the set $\Lambda - \{\lambda\}$, we use $y^{\hat{\lambda}}$ to denote a value of $Y^{\hat{\lambda}}$, namely an assignment to all noisy copies but Y^λ. In a context where a point $y \in (\{0,1\}^n)^\Lambda$ is specified, we use $y^{\hat{\lambda}}$ and y^λ to denote the restrictions of y to the λ coordinate or the set $\hat{\lambda}$ of coordinates, respectively. We also write $y = (y^\lambda, y^{\hat{\lambda}})$.

Lemma 23. *Let Z be a box random variable. Then*

$$\begin{aligned} \mathbb{E}_0[L^\lambda(Z)] &\le \frac{3\log(1/\varepsilon)}{\varepsilon^2 n}\mathbb{H}_0[Z|Y^{\hat{\lambda}}], \\ \mathbb{E}_0[(L^\lambda(Z))^2] &\le \frac{6(\log(1/\varepsilon))^2}{\varepsilon^2 n}\mathbb{H}_0[Z|Y^{\hat{\lambda}}]. \end{aligned}$$

Proof. Since Z is a box random variable, its value is determined by the value of Y^Λ. We can therefore write $Z = Z(y) = Z(y^\lambda, y^{\hat{\lambda}})$. Lemma 23 is obtained by applying Corollary 20 to restrictions of Z of the form $Z(Y^\lambda, y^{\hat{\lambda}})$, which are λ-random variables. Let us therefore denote the random variable $Z(Y^\lambda, y^{\hat{\lambda}})$ by $Z_{y^{\hat{\lambda}}}$. Letting y range over all values of Y^Λ, we have

$$\begin{aligned} \mathbb{E}_0[L^\lambda(Z)] &= \sum_y p_0[Y^\Lambda = y] L^\lambda([Z = Z(y)]) \\ &= \sum_y p_0[Y^\Lambda = y] L([Z = Z(y)]^\lambda) \\ &= \sum_y p_0[Y^\Lambda = y] L([Z(Y^\lambda, y^{\hat{\lambda}}) = Z(y)]) \\ &= \sum_{y^{\hat{\lambda}}} p_0[Y^{\hat{\lambda}} = y^{\hat{\lambda}}] \mathbb{E}_0\left[L(Z_{y^{\hat{\lambda}}})\right] \\ &\le \sum_{y^{\hat{\lambda}}} p_0[Y^{\hat{\lambda}} = y^{\hat{\lambda}}] \frac{3\log(1/\varepsilon)}{\varepsilon^2 n}\mathbb{H}_0[Z|Y^{\hat{\lambda}} = y^{\hat{\lambda}}] \qquad \text{(by Corollary 20)} \\ &= \frac{3\log(1/\varepsilon)}{\varepsilon^2 n}\mathbb{H}_0[Z|Y^{\hat{\lambda}}]. \end{aligned}$$

The bound on $\mathbb{E}_0[(L^\lambda[Z])^2]$ is obtained similarly. □

Lemma 24. *For any box random variable Z,*

$$\mathbb{H}_0[Z] = \sum_\lambda \mathbb{H}_0[Z|Y^{\hat{\lambda}}].$$

Proof. As in the proof of Lemma 23 we can write $Z = Z(y)$, since Z depends only on the value of Y^{Λ}. We denote the box events $[Z = Z(y)]$ by $B(y)$, and let $\mathcal{B} = \{B(y)\}_y$. In the sums below, y ranges over the values of Y^{Λ} and $y^{\hat{\lambda}}$ ranges over the values of $Y^{\hat{\lambda}}$:

$$\begin{aligned}
-\mathbb{H}_0[Z] &= \sum_{B\in\mathcal{B}} p_0[B]\log p_0[B]\\
&= \sum_{B\in\mathcal{B}} p_0[B]\sum_{\lambda}\log p_0[B^{\lambda}]\\
&= \sum_{\lambda}\sum_{B\in\mathcal{B}} p_0[B]\log p_0[B^{\lambda}]\\
&= \sum_{\lambda}\sum_{y} p_0[Y^{\Lambda}=y]\log p_0[B(y)^{\lambda}]\\
&= \sum_{\lambda}\sum_{y^{\hat{\lambda}}}\Big(p_0[Y^{\hat{\lambda}}=y^{\hat{\lambda}}]\cdot\sum_{y^{\lambda}} p_0[Y^{\lambda}=y^{\lambda}]\log p_0[B(y)^{\lambda}]\Big)\\
&= \sum_{\lambda}\sum_{y^{\hat{\lambda}}}\Big(p_0[Y^{\hat{\lambda}}=y^{\hat{\lambda}}]\cdot\sum_{y^{\lambda}} p_0[Y^{\lambda}=y^{\lambda}|Y^{\hat{\lambda}}=y^{\hat{\lambda}}]\log p_0[B(y)|Y^{\hat{\lambda}}=y^{\hat{\lambda}}]\Big)\\
&= -\sum_{\lambda}\sum_{y^{\hat{\lambda}}}\Big(p_0[Y^{\hat{\lambda}}=y^{\hat{\lambda}}]\cdot\sum_{y^{\lambda}}\mathbb{H}_0[Z|Y^{\hat{\lambda}}=y^{\hat{\lambda}}]\Big)\\
&= -\sum_{\lambda}\mathbb{H}_0[Z|Y^{\hat{\lambda}}]. \qquad \square
\end{aligned}$$

Since Π is a box random variable, we obtain the following as an immediate consequence of Lemmas 23 and 24.

COROLLARY 25. *Let T be a* gnd*-tree. Then the random variable* Π *defined over the execution space of T satisfies*

$$\mathbb{E}_0[L(\Pi)] \le \frac{3\log(1/\varepsilon)}{\varepsilon^2 n}\mathbb{H}_0[\Pi] \tag{20}$$

$$\text{and} \qquad \mathbb{E}_0\left[\sum_{\lambda}(L^{\lambda}(\Pi))^2\right] \le \frac{6(\log(1/\varepsilon))^2}{\varepsilon^2 n}\mathbb{H}_0[\Pi]. \tag{21}$$

4.4. Dealing with correlations. Since $\mathbb{H}_0[\Pi]$ is obviously bounded from above by $\mathsf{depth}(T)$, we would be happy to prove a bound on $\mathbb{E}_0[|L(\Pi)|]$ of the form $\mathbb{H}_0[\Pi]\cdot \mathrm{polylog}(1/\varepsilon)/n$. Corollary 25 comes close: noting that $L(\Pi)=\sum_{\lambda}L^{\lambda}(\Pi)$ we see that if only the variables $L^{\lambda}(\Pi)$ were uncorrelated, (21) would have given a bound on the second moment of $L(\Pi)$, and together with (20) we would have easily obtained the desired bound (since $L(\Pi)$ can be negative, (20) by itself is not sufficient to get the bound).

However, in general the variables $L^{\lambda}(\Pi)$ can be correlated. For example, T can decide to query one noisy copy only if queries to a different copy have yielded very little gain to the progress measure. This would introduce correlation between the contributions to the progress measure of those two copies. To handle the correlations we define two random variables, $Q = Q(\Pi)$ and $R = R(\Pi)$. Random variable Q is the sum over nodes on the path to Π, of the expected progress due to the query at that node (a precise definition appears below), and $R = L(\Pi) - Q$.

We show that Q has the same expectation as $L(\Pi)$ but is nonnegative, and thus $\mathbb{E}_0[|Q|] = \mathbb{E}_0[Q] = \mathbb{E}_0[L(\Pi)]$, which we already know is suitably small from (20). It is then left to show a bound on $\mathbb{E}_0[|R|]$. We show that R can be written as a sum $R = \sum_{\lambda \in \Lambda} R^\lambda$ of contributions from the individual noisy copies, but unlike the variables $L^\lambda(\Pi)$, the R^λ's are uncorrelated. The bound on $\mathbb{E}_0[|R|]$ is then obtained by proving a bound on $\mathbb{E}_0[(R^\lambda)^2]$ for all $\lambda \in \Lambda$.

The variables $\boldsymbol{L}$, $\boldsymbol{Q}$, and $\boldsymbol{R}$. In this subsection we write L to denote the random variable $L(\Pi)$, and L^λ for $L^\lambda(\Pi)$ (we still use the notation $L(v)$ and $L^\lambda(v)$ for vertices other than Π). Recall that V is the set of vertices of T, and define a partial order on V by writing $v \leq w$ if v lies on the path from the root to w. We say that v and w are *incomparable* if neither $v \leq w$ nor $w \leq v$. For an internal vertex v, let $v0$ and $v1$ denote its children.

In order to define Q, we first decompose L as a sum of random variables $\{L_v\}_{v \in V}$, where L_v represents the contribution of the query at vertex v to L. For vertices $v, w \in T$ we define

$$L_v(w) = \begin{cases} L(v0) - L(v) & \text{if } v0 \leq w, \\ L(v1) - L(v) & \text{if } v1 \leq w, \\ 0 & \text{otherwise.} \end{cases}$$

$L_v(w)$ measures the contribution of the query made at v to $L(w)$. It easily follows from the definition that $L(w) = \sum_{v \in V} L_v(w)$. Moreover, one can verify that

$$L^\lambda(w) = \sum_{v \in V^\lambda} L_v(w)$$

(where V^λ is the set of vertices in T that query the noisy copy indexed by λ). For every v we define the random variable $L_v = L_v(\Pi)$. We further define the following random variables:

$$\begin{aligned} Q_v &= \begin{cases} \mathbb{E}_0[L_v | \mathsf{vis}(v)] & \text{if } v \leq \Pi, \\ 0 & \text{otherwise,} \end{cases} \\ R_v &= L_v - Q_v, \\ Q &= \sum_{v \in V} Q_v, \qquad \text{and} \quad Q^\lambda = \sum_{v \in V^\lambda} Q_v \quad \text{for every } \lambda \in \Lambda, \\ R &= \sum_{v \in V} R_v, \qquad \text{and} \quad R^\lambda = \sum_{v \in V^\lambda} R_v \quad \text{for every } \lambda \in \Lambda. \end{aligned}$$

Q_v can be thought of as an a priori approximation to L_v, and R_v can be thought of as the a posteriori correction to Q_v. The following are some properties of Q_v and R_v.

Proposition 26. *For any internal vertices v and w,*

1. $\mathbb{E}_0[Q_v] = \mathbb{E}_0[L_v]$*;*
2. *if* $v \geq w$*,* $\mathbb{E}_0[Q_v | \mathsf{vis}(w)] = \mathbb{E}_0[L_v | \mathsf{vis}(w)]$*;*
3. *if* $v \geq w$*,* $\mathbb{E}_0[R_v | \mathsf{vis}(w)] = 0$*;*
4. $Q_v \geq 0$.

Proof. Part 1 is immediate from the definitions of Q_v and L_v. For part 2, since $Q_v = L_v = 0$ when w is not visited, we have $\mathbb{E}_0[Q_v | \mathsf{vis}(w)] = \frac{1}{\Pr[\mathsf{vis}(w)]} \mathbb{E}_0[Q_v] = \frac{1}{\Pr[\mathsf{vis}(w)]} \mathbb{E}_0[L_v] = \mathbb{E}_0[L_v | \mathsf{vis}(w)]$. Part 3 follows immediately from part 2. For part 4, fix v, and for $j \in \{0,1\}$ and $i \in \{0,\dots,n\}$ let $a_i(j) = p_i(vj)/p_i(v)$. Observe that $a_i(0) + a_i(1) = 1$ for each i. Now, if v is not visited, then by definition $Q_v = 0$; otherwise, if v is visited, we have

$$Q_v = \frac{1}{n}\sum_{i=1}^{n}\left(a_0(0)\log\frac{a_0(0)}{a_i(0)} + a_0(1)\log\frac{a_0(1)}{a_i(1)}\right),$$

and each term of the sum is nonnegative by Proposition 11. □

PROPOSITION 27. *For any vertices $v, w \in V$,*

1. *if v and w are incomparable, then $\mathbb{E}_0[Q_vQ_w] = 0$;*
2. *if $v \le w$, then $\mathbb{E}_0[Q_vQ_w] = \mathbb{E}_0[Q_vL_w]$;*
3. *if $v \ne w$, then $\mathbb{E}_0[R_vR_w] = 0$.*

Proof. If v and w are incomparable, then on any execution of T either v or w is not visited, and so $Q_vQ_w = R_vR_w = 0$ and we have part 1. Part 2 follows since both Q_w and L_w are 0 if w is not visited, and conditioned on w being visited, Q_v is constant. Formally,

$$\begin{aligned}\mathbb{E}_0[Q_vQ_w] &= \Pr[\mathsf{vis}(w)]\mathbb{E}_0[Q_vQ_w|\mathsf{vis}(w)]\\ &= \Pr[\mathsf{vis}(w)]\mathbb{E}_0[Q_v|\mathsf{vis}(w)]\mathbb{E}_0[Q_w|\mathsf{vis}(w)]\\ &= \Pr[\mathsf{vis}(w)]\mathbb{E}_0[Q_v|\mathsf{vis}(w)]\mathbb{E}_0[L_w|\mathsf{vis}(w)]\\ &= \Pr[\mathsf{vis}(w)]\mathbb{E}_0[Q_vL_w|\mathsf{vis}(w)] = \mathbb{E}_0[Q_vL_w].\end{aligned}$$

For part 3, if v and w are incomparable, then R_vR_w is 0. If $v < w$, then the product is 0 unless w is visited. But conditioned on w being visited, R_v is a constant, and $\mathbb{E}_0[R_w|\mathsf{vis}(w)] = 0$. □

As an immediate corollary of part 3 of Proposition 27 we have the following.

COROLLARY 28. *For $\lambda \ne \kappa \in \Lambda$, $\mathbb{E}_0[R^\lambda R^\kappa] = 0$.*

Bounds for *L*, *Q*, and *R*. Having defined L, Q, R, and related variables, and proven their basic properties, we now prove some bounds for these variables that ultimately yield (12). The first bound is on the contribution that the subtree of T below a given vertex can make to L^λ.

PROPOSITION 29. *Let $v \in V^\lambda$. Then $|\sum_{w\in V^\lambda : w\ge v} L_w| \le 2\log(1/\varepsilon)$.*

Proof. The sum inside the absolute value equals 0 unless $\Pi \ge v$. If $\Pi \ge v$, then the sum is equal to $|L^\lambda(\Pi) - L^\lambda(v)| \le |L^\lambda(\Pi)| + |L^\lambda(v)| \le 2\log(1/\varepsilon)$ by Lemma 16. □

Next we show that the second moment of R^λ cannot be much higher than the second moment of L^λ.

LEMMA 30. *For each $\lambda \in \Lambda$, $\mathbb{E}_0[(R^\lambda)^2] \le \mathbb{E}_0[(L^\lambda)^2] + 6\log(1/\varepsilon)\mathbb{E}_0[L^\lambda]$.*

Proof.

$$\begin{aligned}\mathbb{E}_0[(R^\lambda)^2] &= \mathbb{E}_0[(L^\lambda - Q^\lambda)^2]\\ &\le \mathbb{E}_0[(L^\lambda)^2] + 2|\mathbb{E}_0[L^\lambda Q^\lambda]| + \mathbb{E}_0[(Q^\lambda)^2]. \qquad (22)\end{aligned}$$

Since by Lemma 16 we have $|L^\lambda| \le \log(1/\varepsilon)$, we can bound the second term in (22) by

$$2|\mathbb{E}_0[L^\lambda Q^\lambda]| \le 2\log(1/\varepsilon)\mathbb{E}_0[|Q^\lambda|] = 2\log(1/\varepsilon)\mathbb{E}_0[L^\lambda].$$

For the third term in (22), we have

$$\begin{aligned}
\mathbb{E}_0[(Q^\lambda)^2] &= \sum_{v,w\in V^\lambda} \mathbb{E}_0[Q_v Q_w] \\
&\overset{(1)}{\leq} 2 \sum_{v,w\in V^\lambda : v\leq w} \mathbb{E}_0[Q_v Q_w] \\
&\overset{(2)}{=} 2 \sum_{v,w\in V^\lambda : v\leq w} \mathbb{E}_0[Q_v L_w] \\
&= 2 \sum_{v\in V^\lambda} \mathbb{E}_0\left[Q_v \cdot \sum_{w\in V^\lambda : w\geq v} L_w\right] \\
&\overset{(3)}{\leq} 2\mathbb{E}_0\left[\sum_{v\in V^\lambda} Q_v\right] \cdot 2\log(1/\varepsilon) \\
&= 4\log(1/\varepsilon)\mathbb{E}_0[Q^\lambda] \overset{(4)}{=} 4\log(1/\varepsilon)\mathbb{E}_0[L^\lambda],
\end{aligned}$$

where (1) follows from part 1 of Proposition 27, (2) follows from part 2 of Proposition 27, (3) follows from Proposition 29, and (4) follows from part 1 of Proposition 26. Combining the bounds for the second and third terms in (22), the lemma is obtained. □

Completing the proof. We are now ready to prove (12), which implies Lemma 14, which in turn completes the proof of Theorem 3.

Since $L = Q + R$, we have

$$\mathbb{E}_0[|L|] \leq \mathbb{E}_0[|Q|] + \mathbb{E}_0[|R|]. \tag{23}$$

Using parts 1 and 4 of Proposition 26, we have that $\mathbb{E}_0[|Q|] = \mathbb{E}_0[Q] = \mathbb{E}_0[L]$. Hence using Corollary 20 and the immediate fact that $\mathbb{H}_0[\Pi] \leq \mathsf{depth}(T)$, we can bound the first term in the right-hand side (r.h.s.) of (23) by

$$\mathbb{E}_0[|Q|] = \mathbb{E}_0[L] \leq \frac{3\log(1/\varepsilon)}{\varepsilon^2 n} \cdot \mathbb{H}_0[\Pi] \leq \frac{3\log(1/\varepsilon)}{\varepsilon^2} \cdot \frac{\mathsf{depth}(T)}{n}. \tag{24}$$

For the second term of the r.h.s. of (23), we have using the Cauchy–Schwarz inequality

$$\begin{aligned}
\mathbb{E}_0[|R|]^2 &\leq \mathbb{E}_0[R^2] \\
&= \sum_\lambda \mathbb{E}_0[(R^\lambda)^2] \qquad \text{(by part 3 of Proposition 27)} \\
&\leq \sum_\lambda \left(\mathbb{E}_0[(L^\lambda)^2] + 6\log\left(\frac{1}{\varepsilon}\right)\mathbb{E}_0[L^\lambda]\right) \qquad \text{(by Lemma 30)} \\
&= \left(\sum_\lambda \mathbb{E}_0[(L^\lambda)^2]\right) + 6\log\left(\frac{1}{\varepsilon}\right)\mathbb{E}_0[L] \\
&\leq \frac{24(\log(1/\varepsilon))^2}{\varepsilon^2 n}\mathbb{H}_0[\Pi] \qquad \text{(by Corollary 25)} \\
&\leq \frac{24(\log(1/\varepsilon))^2}{\varepsilon^2}\frac{\mathsf{depth}(T)}{n}.
\end{aligned}$$

Combining the bounds for Q and for R we get

$$\mathbb{E}_0[|L|] \le \frac{3\log(1/\varepsilon)}{\varepsilon^2} \cdot \frac{\mathsf{depth}(T)}{n} + \frac{5\log(1/\varepsilon)}{\varepsilon} \cdot \sqrt{\frac{\mathsf{depth}(T)}{n}},$$

which completes the proof. □

5. Lower bound for decision functions.

In this section we prove lower bounds for computing *decision functions* (functions that output a single boolean value), as stated in Theorem 5. Our lower bounds are stated in terms of the *sensitivity* of the function to be computed, which is defined as follows.

Definition 31. *Let $f : \{0,1\}^n \to \{0,1\}$ be any function. The sensitivity of f at input $x \in \{0,1\}^n$, denoted $s_x(f)$, is the number of indices $i \in [n]$ such that $f(x) \neq f(x \oplus \mathbf{e_i})$. The sensitivity of f is the maximum of $s_x(f)$ over all x.*

For example, AND, OR, and PAR all have sensitivity n, and every other symmetric boolean function has sensitivity at least $n/2$.

We have the following lower bound.

Theorem 32. *Let $\varepsilon \in (0, 1/2)$ and $\delta \in (0, 1/16)$, and let f be an n-variate boolean function. Any randomized gnd-tree that, for every input x, outputs $f(x)$ with probability $1 - \delta$ when run with noise parameter ε satisfies*

$$\mathsf{depth}(T) \ge \frac{\varepsilon^2 \log(1/4\delta)}{200\log^2(1/\varepsilon)} \cdot s(f).$$

Thus gnd-trees that compute symmetric boolean functions with small constant probability of error require linear depth. While this fact is trivial for noisy broadcast protocols (since the protocol must consider all bits of the input), it is not so for gnd-trees. In fact, we do not know how to achieve linear lower bounds for gnd-trees without applying the full power of the techniques developed in section 4.

Theorem 32 also implies that gnd-trees for symmetric functions that are correct with probability $1 - n^{\Omega(1)}$ must have depth $\Omega(n \log n)$. Before proving Theorem 32, we derive a corollary for noisy broadcast protocols which shows that they require $\Omega(n \log\log n)$ broadcasts to compute symmetric functions with polynomially small error probability.

Corollary 33. *Let $\varepsilon \in (0, 1/2)$ and $\beta > 0$. Let f be an n-variate boolean function. Any randomized noisy broadcast protocol for f that, for every input x, outputs $f(x)$ with probability $1 - n^{-\beta}$ when run with noise parameter ε uses at least $\Omega(s(f) \cdot \frac{\log\log(n^\beta)}{\log(1/\varepsilon)})$ broadcasts.*

Proof. Suppose that A is a noisy broadcast protocol for f that on any input x has error probability $1 - n^{-\beta}$, and suppose A uses $t \cdot s(f)$ broadcasts. Applying Theorem 6 with $\alpha = \frac{2n}{s(f)}$ yields a subset K of $s(f)/2$ variables given by the theorem. Let z be an input with sensitivity $s(f)$. If we fix the variables of K according to z, the resulting function f' with variables indexed by $[n] - K$ has sensitivity at least $s(f)/2$. Furthermore, by the conclusion of Theorem 6, there is a gnd-tree T that uses $2t \cdot s(f)$ queries and for all inputs $y \in \{0,1\}^{[n]-K}$ outputs $f'(y)$ with probability at least $1 - n^{-\beta}$ when run with noise parameter ε^{2t}. By Theorem 32, we have

$$2t \cdot s(f) \ge \frac{\varepsilon^{4t} \log(n^\beta/4)}{50\log^2(1/\varepsilon^{2t})} \cdot s(f),$$

from which the claimed lower bound easily follows. □

The decision-function execution space Υ^z. Let T be a gnd-tree. To prove Theorem 32, it suffices to prove a suitable upper bound on the probability that T is correct, when T is executed on an input selected at random from some distribution. Rather than use the uniform distribution as we did for Theorem 3, we use a distribution tailored to f.

Let $z \in \{0,1\}^n$ be chosen such that $s_z(f) = s(f)$, and let $I \subseteq [n]$ be such that $f(z) \neq f(z \oplus \mathbf{e_i})$ for $i \in I$. The ε-execution space for f, denoted $\Upsilon^z = \Upsilon^z(T, \varepsilon)$, is similar to Υ defined in section 4, except that X is set to z with probability 1/2 and is set to $z \oplus \mathbf{e_i}$ with probability $1/2s$ for each $i \in I$.

Without loss of generality, we may assume that $z = \mathbf{0}$ and that $f(\mathbf{0}) = 0$ (we can replace f by the function mapping x to $f(x \oplus z) \oplus f(z)$ and make the analogous change to T). Furthermore, we may assume that $s(f) = n$ by fixing all variables outside I to 0 in both f and T. We therefore have that the execution space for f is $\Upsilon^{\mathbf{0}}$, and it satisfies that the input X is equal to $\mathbf{0}$ with probability 1/2 (in which case $f(X) = 0$) and to any of the vectors $\mathbf{e_i}$ with probability $1/2n$ (in which case $f(X) = 1$). As in the proof of Theorem 3, since we are proving a lower bound with respect to an input distribution, we may now assume that T is deterministic.

The progress measure. We use the same progress measure L as was used in section 4. Note that once X is fixed to be $\mathbf{0}$ the spaces $\Upsilon(T, \varepsilon)$ and $\Upsilon^{\mathbf{0}}(T, \varepsilon)$ become identical, and therefore the values of $p_0(v)$, $p_i(v)$, $L_i(v)$, and $L(v, \mathbf{0})$, as defined at the beginning of section 4.1, remain unchanged when defined over $\Upsilon^{\mathbf{0}}(T, \varepsilon)$. For the same reason, Lemma 14 holds for $x = \mathbf{0}$ with Υ replaced by $\Upsilon^{\mathbf{0}}$. Using the notation for zero-conditioned expectation as defined in (11) (again, the definition is the same whether it is done over Υ or over $\Upsilon^{\mathbf{0}}$ since it fixes X), we have that (12) holds over $\Upsilon^{\mathbf{0}}(T, \varepsilon)$.

Adaptation of notation. As in section 4, we denote $L(v) = L(v, \mathbf{0})$. We also define the following random variables and events over $\Upsilon^{\mathbf{0}}(T, \varepsilon)$:

- Let Π denote the leaf that terminates the execution of T over $\Upsilon^{\mathbf{0}}(T, \varepsilon)$.
- Let err denote the event that T does not output $f(X)$ (when this occurs we say that T errs).

5.1. The analogue of Lemma 13. Having observed that Lemma 14 applies in the decision-function case (for $x = \mathbf{0}$), we now need an analogue of Lemma 13. The intuition here is that Lemma 14 bounds how well a gnd-tree can distinguish the zero input from its neighbors, as a function of its depth. We now need to show that since on the neighbors of $\mathbf{0}$ the gnd-tree must return different values than on $\mathbf{0}$, it must in fact distinguish well between $\mathbf{0}$ and its neighbors, or else it will err.

The analogue of Lemma 13 for the decision-function case is given in Lemma 35, but first we derive a lower bound on the probability that T errs, in terms of the probability that on input $\mathbf{0}$, T arrives at a leaf π where $L(\pi)$ is small. Define, for every $t \geq 1$,

$$A_t = \{\pi \text{ leaf of } T \mid L(\pi) \leq \log t\}. \tag{25}$$

Lemma 34. *For every $t \geq 1$,*

$$\Pr[\mathsf{err}] \geq \frac{\Pr[A_t | X = 0]}{2t}.$$

Proof. For every leaf π we have

$$L(\pi) = \log\left(\Pr[\Pi = \pi | X = 0]\right) - \frac{1}{n}\sum_i \log\left(\Pr[\Pi = \pi | X = e_i]\right) \qquad \text{(by definition)}$$

$$\geq \log\left(\Pr[\Pi=\pi|X=0]\right) - \log\left(\frac{1}{n}\sum_i \Pr[\Pi=\pi|X=e_i]\right) \qquad \text{(concavity of log)}$$
$$= \log\left(\Pr[\Pi=\pi|X=0]\right) - \log\left(\Pr[\Pi=\pi|X\neq 0]\right) \qquad \text{(by definition of } \Upsilon^{\mathbf{0}}\text{)}.$$

Exponentiating both sides of the inequality with base 2, we get

$$\Pr\left[\Pi=\pi|X\neq 0\right] \geq \frac{\Pr\left[\Pi=\pi|X=0\right]}{2^{L(\pi)}}. \tag{26}$$

Now let $A_t^0 = \{\pi \in A_t | out_\pi = 0\}$ denote the set of leaves in A_t where T outputs zero, and similarly, let $A_t^1 = \{\pi \in A_t | out_\pi = 1\}$. Since $f(\mathbf{0}) = 0$ and $f(\mathbf{e_i}) = 1$ for all $i \in [n]$, we have

$$\begin{aligned}\Pr[\mathsf{err}] &= \frac{1}{2}\Pr[\mathsf{err}|X=0] + \frac{1}{2}\Pr[\mathsf{err}|X\neq 0]\\ &\geq \frac{1}{2}\Pr\left[A_t^1|X=0\right] + \frac{1}{2}\Pr\left[A_t^0|X\neq 0\right]\\ &\geq \frac{1}{2}\Pr\left[A_t^1|X=0\right] + \frac{1}{2}\Pr\left[A_t^0|X=0\right]\cdot\frac{1}{t} \quad \text{(by (26) and definition of } A_t\text{)}\\ &\geq \frac{1}{2t}\cdot\left(\Pr\left[A_t^1|X=0\right] + \Pr\left[A_t^0|X=0\right]\right)\\ &= \frac{\Pr\left[A_t|X=0\right]}{2t},\end{aligned}$$

concluding the proof. □

The following lemma is the analogue of Lemma 13 for the decision-function case.

LEMMA 35. *Denote* $\delta = \Pr[\mathsf{err}]$. *Then if* $\delta \leq 1/4$,

$$\mathbb{E}\left[|L(\Pi)||X=0\right] \geq \frac{1}{2}\log\left(\frac{1}{4\delta}\right).$$

Proof. Take $t = 1/4\delta$. Then by Lemma 34 we have

$$\delta = \Pr[\mathsf{err}] \geq \frac{\Pr[A_t|X=0]}{2t} = 2\delta\cdot\Pr[A_t|X=0],$$

and thus

$$\Pr[A_t|X=0] \leq 1/2.$$

Now on the complement of A_t with respect to the set of leaves in T, the values of L are greater than $\log t$. We therefore have

$$\mathbb{E}\left[|L(\Pi)||X=0\right] \geq (1-\Pr[A_t|X=0])\cdot\log(t) \geq \frac{1}{2}\log\left(\frac{1}{4\delta}\right),$$

as desired. □

5.2. Proof of Theorem 32. We are now ready to complete the proof of Theorem 32.

Proof of Theorem 32. Let us consider the execution of T over $\Upsilon^{\mathbf{0}}(T,\varepsilon)$. It is given that $\Pr[\mathsf{err}] = \delta < \frac{1}{16}$. Applying Lemma 35 and (12), we thus have that

$$\frac{3\log(1/\varepsilon)}{\varepsilon^2}\cdot\frac{\mathsf{depth}(T)}{n} + \frac{5\log(1/\varepsilon)}{\varepsilon}\cdot\sqrt{\frac{\mathsf{depth}(T)}{n}} \geq t, \tag{27}$$

where $t = \frac{1}{2}\log\left(\frac{1}{4\delta}\right) \geq 1$. Therefore either the first or the second summand in (27) is greater than $\frac{t}{2}$, and by isolating $\mathsf{depth}(T)$ in the resulting inequalities we obtain

$$\mathsf{depth}(T) \geq \min\left\{\frac{\varepsilon^2 tn}{6\log(1/\varepsilon)}\ ,\ \frac{\varepsilon^2 t^2 n}{100\log^2(1/\varepsilon)}\right\} \geq \frac{\varepsilon^2 tn}{100\log^2(1/\varepsilon)},$$

as desired. □

6. Proof of the reduction theorem (Theorem 6).

Let us now prove Theorem 6. We are given a noisy broadcast protocol $\mathcal{P}$ that uses k broadcasts. We are also given a noise parameter ε and another parameter $\alpha > 1$. Our goal is to select a subset K of size n/α and show that for each partial assignment ρ that fixes K there is a (possibly randomized) gnd-tree T that uses $2k$ queries and, for any input $x \in \{0,1\}^{[n]-K}$, when T is run on x with noise parameter $\varepsilon^{\alpha k/n}$ it gives the same output that $\mathcal{P}$ gives when run on the input $x\rho \in \{0,1\}^n$ with noise parameter ε.

We select K to be the index set of processors that broadcast more than $\alpha k/n$ times in $\mathcal{P}$. Since there are k broadcasts in $\mathcal{P}$, $|K| < n/\alpha$.

Now fix a partial assignment ρ to K. The gnd-tree T is obtained by a sequence of reductions. These reductions involve two intermediate models of noisy computation which we now define.

The *seminoisy broadcast model* $SNB(\varepsilon)$ is similar to the noisy broadcast model. In this model there are n *input processors* $Q_1, \ldots, Q_n$, a receiver P_0, and a collection $\{P_\lambda : \lambda \in \Lambda\}$ of *auxiliary processors* (where Λ is an arbitrary index set). Q_i initially has input x_i and is restricted to making ε-noisy broadcasts of x_i. An auxiliary processor P_λ may broadcast any boolean function of the bits it heard previously, and this broadcast is noise-free, i.e., is received by every other processor with no error.

In the *noisy-copy broadcast model* $NCB(\varepsilon)$, there is a receiver P_0 and an indexed collection $\{P_\lambda : \lambda \in \Lambda\}$ of processors. Each of the P_i initially gets an (independent) ε-noisy copy of the *entire* input. All broadcasts are noise-free.

Starting from $\mathcal{P} =: \mathcal{P}_1$, we construct a sequence of protocols:

- $\mathcal{P}_2$ in the model $SNB(\varepsilon)$ takes input in $\{0,1\}^n$.
- $\mathcal{P}_3$ in the model $SNB(\varepsilon)$ takes input in $\{0,1\}^{[n]-K}$.
- $\mathcal{P}_4$ runs in the model $NCB(\varepsilon^{\alpha k/n})$ and takes input in $\{0,1\}^{[n]-K}$.
- T is a gnd-tree that takes input in $\{0,1\}^{[n]-K}$.

As stated, there are k broadcasts in $\mathcal{P}$. Using the notation from section 2.3, we have, for $1 \leq j \leq k$, the following:

- i^j is the index of the processor broadcasting at step j.
- g^j is the boolean function (depending on x_j and the bits received by P_j prior to step j) that determines broadcast j.
- b^j is the bit broadcast at step j (obtained by evaluating g^j).
- For $h \in [n]$, b_h^j is the noisy copy of b^j received by P_h.

We also introduce the notation g_0^j and g_1^j for the functions obtained from g^j by setting x_j to 0 and to 1, respectively.

From $\mathcal{P}_1$ *to* $\mathcal{P}_2$. Given $\mathcal{P}_1$, the protocol $\mathcal{P}_2$ in the model $SNB(\varepsilon)$ will consist of k stages. Stage j simulates broadcast j of $\mathcal{P}_1$ and consists of three broadcasts.

The index set Λ for auxiliary processors is $[n]$. In $\mathcal{P}_2$, we will maintain the invariant that after each stage j the following holds:

> For $0 \leq i \leq n$, P_i has constructed a sequence $b_1^i, \ldots, b_j^i$ of bits and the collection $\{b_h^i : 0 \leq i \leq n,\ 1 \leq h \leq j\}$ has the same joint probability distribution as it does in $\mathcal{P}_1$.

P_0 will compute its output exactly as in $\mathcal{P}_1$, and the invariant ensures that the outputs of the two protocols have the same distribution.

Assume that the invariant holds inductively after stage $j-1$. We define stage j of $\mathcal{P}_2$ so as to maintain the invariant. In stage j, Q_{i^j} broadcasts x_{i^j} (which is all it can do in this model) and P_{i^j} evaluates both g_0^j and g_1^j at $b_1^{i^j}, \ldots, b_{j-1}^{i^j}$ and broadcasts the two values.

Each P_i must generate b_i^j. P_i receives three bits (x, a_0, a_1), where x is a noisy copy of x_{i^j} broadcast by Q_{i^j}, and a_0 and a_1 are the exact bits sent by P_{i^j}. If $a_0 \neq a_1$, P_i sets b_j^i to a_x, and if $a_0 = a_1$, P_i uses its private randomness to generate an ε-noise bit N and sets b_j^i to $a_0 \oplus N$. It is easy to check that this choice maintains the invariant.

The total number of broadcasts in $\mathcal{P}_2$ is $3k$, of which k are by input processors and $2k$ are by auxiliary processors.

From $\mathcal{P}_2$ to $\mathcal{P}_3$. We now construct a protocol $\mathcal{P}_3$ for the restricted function $f\lceil_\rho$ in the model $SNB(\varepsilon)$. For this function, the input (and the Q_i) are indexed by $[n]-K$. The index set Λ of auxiliary processors is chosen to be $[n]$.

$\mathcal{P}_3$ simulates $\mathcal{P}_2$ in a step by step fashion. The only difficulty is that the processors Q_i for $i \in K$ do not exist when running $\mathcal{P}_3$. In $\mathcal{P}_2$, processor Q_i would send its input bit, which is now fixed to ρ_i. In $\mathcal{P}_3$, the value ρ_i can be "hardwired" and each processor simply simulates the reception of ρ_i as $\rho_i \oplus N$, where N is an ε-noise bit that it generates locally. Every other step is done exactly as in $\mathcal{P}_2$.

The total number of broadcasts in $\mathcal{P}_3$ is $3k$, of which at most k are by input processors and the remaining are by other processors. Furthermore, no input processor sends its bit more than $\alpha k/n$ times.

From $\mathcal{P}_3$ to $\mathcal{P}_4$. We now construct a protocol $\mathcal{P}_4$ in the $NCB(\gamma)$ model for $\gamma = \varepsilon^{\alpha k/n}$ that simulates $\mathcal{P}_3$.

In $\mathcal{P}_3$, each input bit x_i is broadcast at most αk times, and so each P_j receives at most αk noisy copies of x_i. In $\mathcal{P}_4$, P_j starts with a γ-noisy copy of x_i. It suffices to show that $\alpha k/n$ independent ϵ-noisy copies of x_i can be generated from a single γ-noisy copy of x_i.

LEMMA 36. *Let t be an arbitrary integer, $\varepsilon \in (0, 1/2)$, and $\gamma = \varepsilon^t$. There is a randomized algorithm that takes as input a single bit b and outputs a sequence of t bits and has the property that if the input is a γ-noisy copy of 0 (resp., of 1), then the output is a sequence of independent ε-noisy copies of 0 (resp., of 1).*

Proof. The algorithm is specified by two probability distributions q_0 and q_1 over $\{0,1\}^t$. On input b, the algorithm outputs a string according to the distribution q_b.

If the input b to the algorithm is a γ-noisy copy of 0 (resp., 1), then the output will be generated according to the distribution function $r_0 = (1-\gamma)q_0 + \gamma q_1$ (resp., $r_1 = (1-\gamma)q_1 + \gamma q_0$).

Let p_0 (resp., p_1) be the distribution on $\{0,1\}^t$ obtained by taking t independent ε-noisy copies of 0 (resp., 1). We want to choose q_0 and q_1 so that $r_0 = p_0$ and $r_1 = p_1$. For each $s \in \{0,1\}^t$, we need

$$(1-\gamma)q_0(s) + \gamma q_1(s) = p_0(s),$$
$$(1-\gamma)q_1(s) + \gamma q_0(s) = p_1(s).$$

Solving for $q_0(s)$ and $q_1(s)$ we get

$$q_0(s) = \frac{(1-\gamma)p_0(s) - \gamma p_1(s)}{1-2\gamma},$$

$$q_1(s) = \frac{(1-\gamma)p_1(s) - \gamma p_0(s)}{1-2\gamma}.$$

It suffices to show that q_0 and q_1 are probability distributions; i.e., they are nonnegative and sum to 1. That they sum to 1 follows from the fact that p_0 and p_1 each sum to 1. Nonnegativity follows immediately from the easy fact that for any $s \in \{0,1\}^t$, $p_0(s)/p_1(s) \geq p_0(1^t)/p_1(1^t) \geq \gamma/(1-\gamma)$ and $p_1(s)/p_0(s) \geq p_1(0^t)/p_0(0^t) \geq \gamma/(1-\gamma)$. □

From $\mathcal{P}_4$ *to* T. $\mathcal{P}_4$ is randomized; i.e., each processor uses its own internal source of random bits in the protocol. We construct a random gnd-tree that first simulates the random choices of all the processors in $\mathcal{P}_4$. Once these are fixed, $\mathcal{P}_4$ reduces to a deterministic protocol. So it is enough to show how to simulate a deterministic protocol $\mathcal{Q}$ in the $NCB(\gamma)$ model by a gnd-tree.

Let $\lambda^1, \ldots, \lambda^{2k}$ be the sequence of indices of processors that broadcast in $\mathcal{Q}$, and let $b^1, \ldots, b^{2k}$ be the bits broadcast. The gnd-tree makes $2k$ queries and will be chosen so that the answer to query j is b^j. Query j is made to copy λ^j. Given that the sequence of answers to the first $j-1$ is $b^1, \ldots, b^{j-1}$, the question asked of copy i^j is, What would processor P_{i^j} broadcast in $\mathcal{Q}$ given that the string of bits received during the first $j-1$ rounds is $b^1, \ldots, b^{j-1}$? The answer to this is a boolean function depending only on copy i^j. Since all broadcasts in $\mathcal{Q}$ are noise-free, the sequence of answers in the gnd-tree has exactly the same distribution as the sequence of bits broadcast by $\mathcal{Q}$.

This completes the proof of Theorem 6. □

7. A protocol for identity.

In this section we give a protocol for computing *id* with $O(n \log\log n)$ broadcasts. Our protocol is similar to, but simpler than, the protocol of Gallager mentioned earlier. Gallager's protocol and ours work in all three models of noise discussed in the introduction.

Gallager gave an $O(n \log\log n)$ broadcast protocol for PAR and used that protocol to construct one for *id*. Our protocol uses constant rate error correcting codes, which are well known to exist (random codes are good) but nontrivial to construct explicitly. Gallager's protocol has the advantage of being self-contained and explicit.

Error correcting codes are an efficient way to communicate large blocks of data on a noisy channel. In the noisy broadcast model, each processor has only a single bit to start, and so error correcting codes are not directly applicable.

For our protocol, we divide the processors into teams of size $t = \log n$. The processors in each team work together to transmit their bits to the receiver. Processors ignore transmissions by processors in other teams.

For the analysis, we will need to assume $\varepsilon < 1/12$. This is without loss of generality since given a protocol $\mathcal{P}$ that works for some constant noise parameter ε' we can get a protocol that works for larger noise parameter $\varepsilon < 1/2$ by a routine amplification: every broadcast in $\mathcal{P}$ is repeated C times for some appropriate constant $C = C(\varepsilon, \varepsilon')$, and each receiver decodes the broadcast by majority vote.

We will describe a protocol for a single team, denoted $T = \{Q_1, \ldots, Q_t\}$. The protocol uses $O(t \log t)$ broadcasts. We show that the probability that the receiver fails to recover the team's entire input is at most $2(4^{-t}) = 2/n^2$. Repeating the same protocol for each team yields a protocol for *id* that fails with probability less than $2/(n \log n)$.

Let $y = (y_1, \ldots, y_t)$ denote the team's input. The protocol works in two phases.

Phase 1. For each $i \in [t]$, Q_i broadcasts y_i $s = c_1 \log t$ times for some constant c_1 to be chosen according to Lemma 37. Every other processor tries to recover y_i by taking a majority vote of the s copies it received.

Let y^i be the copy of y recovered by Q_i. Say that Q_i is *successful* if $y^i = y$. Let U denote the set of unsuccessful processors.

LEMMA 37. *There exists a constant* $c_1 = c_1(\varepsilon)$ *such that*

$$\Pr[[|U| \geq \varepsilon t] \leq 4^{-t}].$$

Proof. Processor Q_i decodes y_j incorrectly if at least $1/2$ of the copies of y_j that Q_i received were corrupted by noise. By Lemma 12 with each $\alpha_i = 1$ and t (in the lemma) equal to $(1/2 - \varepsilon)s$, the probability that Q_i decodes y_j incorrectly is at most $e^{-2(1/2-\varepsilon)^2 s}$. Summing over the t bits of y, $\Pr[Q_i \in U]$ is at most $q = te^{-2(1/2-\varepsilon)^2 c_1 \log t}$.

For $S \subseteq T$, $\Pr[U = S] \leq q^{|S|}$. Summing over subsets S of size at least εt, we obtain

$$\Pr[|U| \geq \varepsilon t] \leq q^{\varepsilon t} 2^t = (2q^{\varepsilon})^t.$$

We want that this is at most 4^{-t}, and so it suffices that $q < (1/8)^{1/\varepsilon}$. It is now easy to choose c_1 sufficiently large (depending on ε) so that this last condition holds. □

Phase 2. In this phase the processors of T work together to transmit an encoding of y based on an error correcting code. For boolean strings v, w of the same length, let $d(v, w)$ denote the number of coordinates where they differ. The following well-known result states what we need about the existence of good codes.

LEMMA 38. *For* $\gamma \in (0, 1/2)$ *there is an integer* $K_1 = K_1(\gamma)$ *such that for each positive integer* t *and each* $K \geq K_1$*, there is a subset* C_t *of* $\{0,1\}^{Kt}$ *having size* 2^t*, such that for all* $v, w \in C_t$ *with* $v \neq w$*,* $d(v, w) \geq \gamma K t$*.*

This is easily proved by the probabilistic method [3]. For γ sufficiently small, there are explicit constructions, e.g., Justesen codes (see, e.g., [21]).

Set γ to be 6ε (here is where we need $\varepsilon < 1/12$). Let $K = \max\{K_1(\gamma), 1/\varepsilon^2\}$, and let C_t be given by the lemma. Fix a bijection σ from $\{0,1\}^t$ to C_t. Divide $\sigma(y)$ into t blocks of size K, and denote the jth block by $\sigma_j(y)$.

In Phase 2, Q_i broadcasts $\sigma_i(y^i)$ (recall that y^i is the copy of y recovered by Q_i). Let r_i be the noisy version of this heard by the receiver, and let r be the concatenation of $r_1, \ldots, r_{\log n}$. The receiver chooses $w \in C_t$ to minimize $d(r, w)$ and outputs $\sigma^{-1}(w)$.

If $d(r, \sigma(y)) < \gamma K t/2 = 3\varepsilon t$, then by the choice of C_t, $w = \sigma(y)$, and the receiver correctly outputs y.

Since every processor that was successful in Phase 1 broadcasts a part of $\sigma(y)$, r differs from $\sigma(y)$ only on coordinates sent by unsuccessful processors and coordinates that were flipped due to noise. Let $I \subseteq [Kt]$ be the index set of the coordinates that were flipped by noise. Thus $d(r, \sigma(y)) \leq K|U| + |I|$.

By Lemma 37, $\Pr[K|U| \geq K\varepsilon t] \leq 4^{-t}$. By Lemma 12, $\Pr[|I| \geq 2K\varepsilon t] \leq e^{-2\varepsilon^2 K t}$, which is at most 4^{-t} for $K \geq 1/\varepsilon^2$. Thus the probability that the receiver does not output the input of the team is at most $2 \times 4^{-t} \leq 2/n^2$. Summing over all $n/\log n$ teams, the probability that the receiver fails is at most $2/(n \log n)$.

8. A linear noisy broadcast protocol computing weight.

In this section we prove Theorem 4 by giving a protocol for computing $\mathsf{weight}(x)$ which uses a linear number of broadcasts in the noisy broadcast model and showing a gnd-tree of linear depth that computes it. This immediately implies linear protocols or gnd-trees for

PAR, or any other boolean function whose value at x depends only on $\mathsf{weight}(x)$. Our protocol does not work under any of the adversarial noise models.

We begin by giving a gnd-tree T of linear depth for $\mathsf{weight}(x)$, and then we convert it to a protocol in the noisy broadcast model.

8.1. A gnd-tree for computing weight. We need some additional notation. For $\theta \in [0, n]$, the threshold function $f_\theta : \{0, 1\}^n \to \{0, 1\}$ is defined to be 1 if $\mathsf{weight}(x) \geq \theta$. All queries made by T are threshold functions. We define the real function $\rho : [0, n] \longrightarrow [0, n]$ by $\rho(a) = (1 - \varepsilon)a + \varepsilon(n - a)$. Observe that an ε-noisy copy of 0 has expectation ε and an ε-noisy copy of 1 has expecation $1 - \varepsilon$. Therefore, for fixed x, the expected weight of a noisy copy of x satisfies

$$\mathbb{E}[\mathsf{weight}(x \oplus N)] = \rho(\mathsf{weight}(x)).$$

The gnd-tree T works in two phases. During the first phase, T does a modified binary search to identify an interval $[a, b]$ of length at most $O(\sqrt{n})$ that contains $\mathsf{weight}(x)$ with high probability. During the second phase, T repeats the threshold query $f_{(a+b)/2}$ on $O(n)$ different copies and determines $\mathsf{weight}(x)$ from this with high probability.

Phase I. During this phase T does a modified binary search to identify an interval of length $O(\sqrt{n})$ that contains $\mathsf{weight}(x)$ with high probability. Initially the interval is $[a_0, b_0] = [0, n]$. The phase consists of stages. During stage i, the interval is reduced to $2/3$ of its previous length. The interval after stage i is denoted $[a_i, b_i]$. Phase 1 ends when $b_i - a_i \leq c(\varepsilon)\sqrt{n}$, for $c(\varepsilon) = 6/(1 - 2\varepsilon)$. This uses $O(\log n)$ stages.

The algorithm for a stage depends on the interval $[a, b]$ at the beginning of the stage and is denoted $S_{a,b}$. Let $m = (a + b)/2$ be the midpoint. $S_{a,b}$ consists of $k = 16 \ln n$ threshold queries $f_{\rho(m)}$ to distinct copies of the input. If the number of 0 answers is more than $k/2$, then the interval $[a', b']$ output by $S_{a,b}$ is $[a, \frac{a+2b}{3}]$, and otherwise it is $[\frac{2a+b}{3}, b]$.

We say that stage $S_{a,b}$ *fails* if $\mathsf{weight}(x)$ belongs to $[a, b]$ but not to $[a', b']$.

LEMMA 39. *Let $c(\varepsilon) = 6/(1 - 2\varepsilon)$. Let $x \in \{0, 1\}^n$, and suppose $a, b \in [0, n]$ satisfy $b - a > c(\varepsilon)\sqrt{n}$ and $\mathsf{weight}(x) \in [a, b]$. The probability that $S_{a,b}$ fails is at most $1/n^2$.*

Proof. We divide the analysis according to which of the intervals $[a, \frac{2a+b}{3})$, $[\frac{2a+b}{3}, \frac{a+2b}{3}]$, and $(\frac{a+2b}{3}, b]$ contains $\mathsf{weight}(x)$.

If $\mathsf{weight}(x) \in [\frac{2a+b}{3}, \frac{a+2b}{3}]$, then $\mathsf{weight}(x) \in [a', b']$ for either of the two possible outputs.

We next consider the case $\mathsf{weight}(x) \in [a, \frac{2a+b}{3})$; the analysis in the other case is very similar and is omitted. For such an x, we say that the answer to a query to $f_{\rho(m)}$ is correct if it returns 0.

Let q denote the probability a single noisy query of $f_{\rho(m)}$ is incorrect. For $i \in [k]$, let Z_i be the random variable that is 1 if the answer to the ith noisy query is incorrectly 1. The stage fails if $\sum_i Z_i \geq k/2$. Using Lemma 12 with $\varepsilon = q$ and all of the $\alpha_i = 1$, the probability that the stage fails is at most

$$\Pr\left[\sum(Z_i - q) \geq k\left(\frac{1}{2} - q\right)\right] \leq e^{-2(1/2-q)^2 k} = e^{-32(1/2-q)^2 \ln n}.$$

If $q \leq 1/4$, then this is at most $1/n^2$. So we complete the proof by showing that $q \leq 1/4$.

Let X be a noisy copy of x and N the associated noise vector. For $i \in [n]$, let $a_i = 1 - 2x_i$. $X_i = x_i + a_i N_i = \rho(x_i) + a_i(N_i - \varepsilon)$ and $\sum_i X_i = \rho(\mathsf{weight}(x)) + \sum a_i(N_i - \varepsilon)$. Note that under the case assumption we have $\rho(m) - \rho(\mathsf{weight}(x)) = (m - \mathsf{weight}(x))(1-2\varepsilon) \geq (b-a)(1-2\varepsilon)/6 \geq c(\varepsilon)\sqrt{n}(1-2\varepsilon)/6 \geq \sqrt{n}$. Lemma 12 implies

$$\begin{aligned} q = \Pr\left[\sum_i X_i \geq \rho(m)\right] &= \Pr\left[\sum_i a_i(N_i - \varepsilon) \geq \rho(m - \mathsf{weight}(x))\right] \\ &\leq \Pr\left[\sum_i a_i(N_i - \varepsilon) \geq \sqrt{n}\right] \leq e^{-2}. \qquad \square \end{aligned}$$

Observe that the total number of queries in Phase 1 is $O((\log n)^2)$.

Bias differences. After successfully completing the first phase T "knows" that the weight of x is in a strip $[a_r, b_r]$ of length roughly $\sqrt{n}$. In the second phase T simply makes a linear number of queries of the form $f_{\rho(\theta)}$ to distinct noisy copies of x, where $\theta = \frac{a_r+b_r}{2}$. The weight of x is estimated based on the number of "1" answers. Intuitively, the larger $\mathsf{weight}(x)$ is, the more 1's we expect to see (note that the distribution on the number of "1" answers depends only on the weight of x). We make this observation more formal in the next definition and lemma.

DEFINITION 40. *Let $\varepsilon \in (0, 1/2)$ be a noise parameter, and let $\theta \in [0, n]$. For any $x \in \{0,1\}^n$ we define*

$$\beta_{n,\varepsilon}(\omega, \theta) = \Pr\left[f_{\rho(\theta)}(x \oplus N) = 1\right],$$

where $\omega = \mathsf{weight}(x)$, and N is an ε-noise vector.

LEMMA 41. *There exists a global constant $C > 0$ which satisfies the following. Let $\varepsilon \in (0, 1/2)$ be a noise parameter, and let $c(\varepsilon)$ be as in Lemma* 39. *Then for n large enough it holds for every $\theta \in [0, n]$ and every $\omega \in [\theta - c(\varepsilon)\sqrt{n},\, \theta + c(\varepsilon)\sqrt{n}]$ that*

$$\beta_{n,\varepsilon}(\omega+1, \theta) - \beta_{n,\varepsilon}(\omega, \theta) \geq \frac{\exp\left(-\frac{C}{\varepsilon(1-2\varepsilon)^4}\right)}{C\sqrt{n}}. \tag{28}$$

Lemma 41 follows in a straightforward way by writing $\beta_{n,\varepsilon}(\omega+1,\theta) - \beta_{n,\varepsilon}(\omega,\theta)$ as an expression involving binomial coefficients and powers of ε and of $(1-\varepsilon)$, and then applying simple estimations using Stirling's formula. We omit the full derivation.

Phase II. For brevity, let us denote the expression in the r.h.s. of (28) by $\Delta(n, \epsilon)$. In the second phase, T applies the threshold tree $T_{k,\theta}$ to x, where $\theta = \frac{a_r+b_r}{2}$ and $k = 17/\Delta^2(n, \epsilon)$. That is, T queries $f_{\rho(\theta)}$ on k distinct noisy copies of x (note that k is linear in n, and thus this adds only a linear depth to T).

To determine the output of T, let Z denote the number of queries of the threshold tree for which the result was 1. Also, for every $\omega \in \{0, 1, \ldots, n\}$ define a segment I_ω by

$$I_\omega = \left[k \cdot \beta_{n,\epsilon}(\omega, \theta) - 2\sqrt{k}\,,\, k \cdot \beta_{n,\epsilon}(\omega, \theta) + 2\sqrt{k}\right].$$

As we show below, the segments I_ω are all disjoint for $\omega \in [a_r, b_r]$. If $Z \in I_\omega$ for such an ω, T outputs ω. Otherwise T declares failure in computing $\mathsf{weight}(x)$.

Analysis of Phase II. As noted above, the depth that the second phase contributes to T is linear in n (the total depth of T is therefore linear, as required). To show that the output is really $\mathsf{weight}(x)$ with high probability, we have to first note that the output is well defined, namely that the segments I_ω are disjoint for all $\omega \in [a_r, b_r]$. Indeed, since each I_ω is a segment of length $4\sqrt{k}$ centered around $k \cdot \beta_{n,\varepsilon}(\omega, \theta)$, the choice of k and Lemma 41 directly imply that these segments are disjoint.

Now suppose $\mathsf{weight}(x) = \omega$ for some $\omega \in [a_r, b_r]$. By definition, each query of the threshold tree applied in the second phase returns the value 1 with probability $\beta_{n,\varepsilon}(\omega, \theta)$, and thus

$$\mathbb{E}[Z] = k \cdot \beta_{n,\varepsilon}(\omega, \theta).$$

Also, since Z is the sum of k independent random variables (the indicators of events of the form "the ith query being 1") each with variance at most 1, it follows that the variance of Z is at most $\sqrt{k}$. Therefore by Chebyshev's inequality,

$$\Pr[Z \in I_\omega] = \Pr\left[|Z - \mathbb{E}[Z]| \leq 2\sqrt{k}\right] \geq \Pr\left[|Z - \mathbb{E}[Z]| \leq 2\sqrt{\mathbb{V}[Z]}\right] \geq 3/4.$$

It follows that, assuming the first phase succeeds, T outputs $\mathsf{weight}(x)$ with probability at least 3/4. The overall probability that T computed $\mathsf{weight}(x)$ correctly is therefore at least 2/3.

8.2. Translating to a noisy broadcast protocol.

We have constructed a gnd-tree T of linear depth in n, which computes $\mathsf{weight}(x)$. Let us sketch how T can be transfomed into a protocol in the noisy broadcast network model. As each query that T makes is applied to a distinct ε-noisy copy of the input, we first need to get hold of that many noisy copies of x.

Obtaining noisy copies of x. Let $\varepsilon \in (0, 1/2)$ be the noisy parameter in a noisy broadcast network, and let $x \in \{0,1\}^n$ be an input. The first step in simulating T is to obtain noisy copies of the input. To achieve $d \cdot n$ noisy copies of x, say, we can make each processor broadcast its own bit d times. After this step (which takes $d \cdot n$ broadcasts) is completed, each processor has d noisy copies of each player's bits. In the protocol described below we need only a linear number of noisy copies, and therefore this step takes only a linear number of broadcasts.

Phase I *simulation.* Once a sufficient number of noisy copies of x has been obtained, we can start by simulating the first phase of T. We can preassign $\Theta(\log^2 n)$ processors to perform the queries originally applied by T in the first phase. To avoid errors, we can have each processor repeat the result of the query it makes, broadcasting it $\Theta(\log n)$ times, so that with very high probability all the other processors will get the result of the query correctly. At the end of the first phase simulation we have that with probability at least $1 - 1/\mathrm{poly}(n)$ all processors agree on the same segment $[a_r, b_r]$, the same as would have been acheived by T using the same noisy copies.

Since T makes $\Theta(\log^2 n)$ queries in the first phase, and since we simulate each query by $\Theta(\log n)$ broadcasts, the simulation of the first phase requires $\Theta(\log^3 n)$ broadcasts.

Phase II *simulation.* The second phase in T consists of counting the number of 1's obtained in a series of threshold queries. In the simulation of this phase the receiver, P_0, will do the counting and output the result accordingly. The actual threshold queries will be applied by the other processors: each of them, in turn, will apply $f_{\rho(\theta)}$ queries to the noisy copies under its possession and broadcast the results.

A slight problem arises from the fact that P_0 may receive the wrong result for some of the queries due to noise. If $\mathsf{weight}(x) = \omega$, the probability that P_0 will receive the answer 1 for a query of type $f_{\rho(\theta)}$ is not $\beta_{n,\varepsilon}(\omega,\theta)$ as before, but rather it is

$$\beta'_{n,\varepsilon}(\omega,\theta) = \varepsilon + (1-2\varepsilon)\beta_{n,\varepsilon}(\omega,\theta).$$

But it is obvious that Lemma 41 still holds for $\beta'_{n,\epsilon}(\omega,\theta)$, perhaps with a different constant C, and thus the simulation of the second phase can still be carried out.

9. Open problems.

The main questions left open by this paper concern lower bounds for decision functions.

1. Can every decision function be computed by a linear noisy broadcast protocol if constant error is allowed? The same question can be asked for gnd-trees, and we believe these two questions should have the same answer, and that the answer should be negative. A random function would seem to be a natural candidate for a hard function.
2. What other interesting classes of decision functions besides symmetric functions have linear noisy broadcast protocols and/or gnd-trees?
3. Are there any functions that can be computed by a gnd-tree whose depth is significantly smaller than their randomized decision-tree complexity, namely the depth required to compute them by a (noiseless) randomized decision tree?
4. The same questions as above can be asked about the adverserial noise model of [11]. For this model, the only nontrivial upper bound known for both the noisy broadcast and gnd-tree models is the protocol for OR due to [22], which is linear in both models. What other nontrivial functions have linear protocols? Does the parity function have linear cost protocols/gnd-trees? We conjecture that the answer to the latter question is negative.
5. In the adversarial noise model, we do not know of any examples of decision functions where gnd-trees do better than ordinary nd-trees. Are these models equivalent for adversarial noise?

REFERENCES

[1] D. Aharonov and M. Ben-Or, *Fault tolerant quantum computation with constant error*, in Proceedings of the 29th Annual ACM Symposium on Theory of Computing, 1997, pp. 176–188.

[2] M. Ajtai, *The invasiveness of off-line memory checking*, in Proceedings of the 34th Annual ACM Symposium on Theory of Computing, 2002, pp. 504–513.

[3] N. Alon and J. Spencer, *The Probabilistic Method*, 2nd ed., Wiley, New York, 2000.

[4] Y. Aumann and M. A. Bender, *Fault tolerant data structures*, in Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science, 1996, pp. 580–589.

[5] T. Cover and J. Thomas, *Elements of Information Theory*, Wiley, New York, 1991.

[6] A. El Gamal, *Open problems presented at the* 1984 *workshop on Specific Problems in Communication and Computation sponsored by Bell Communication Research*, in Open Problems in Communication and Computation, T. M. Cover and B. Gopinath, eds., Springer-Verlag, New York, 1987, pp. 60–62.

[7] W. Evans, C. Kenyon, Y. Peres, and L. J. Schulman, *Broadcasting on trees and the Ising model*, Ann. Appl. Probab., 10 (2000), pp. 410–433.

[8] W. Evans and N. Pippenger, *Average-case lower bounds for noisy Boolean decision trees*, SIAM J. Comput., 28 (1998), pp. 433–446.

[9] W. S. Evans and L. J. Schulman, *Signal propagation and noisy circuits*, IEEE Trans. Inform. Theory, 44 (1998), pp. 1299–1305.

[10] U. Feige, *On the complexity of finite random functions*, Inform. Process. Lett., 44 (1992), pp. 295–296.

[11] U. Feige and J. Kilian, *Finding OR in a noisy broadcast network*, Inform. Process. Lett., 73 (2000), pp. 69–75.

[12] U. Feige, P. Raghavan, D. Peleg, and E. Upfal, *Computing with noisy information*, SIAM J. Comput., 23 (1994), pp. 1001–1018.

[13] I. Finocchi and G. F. Italiano, *Sorting and searching in the presence of memory faults (without redundancy)*, in Proceedings of the 36th Annual ACM Symposium on Theory of Computing, 2004, pp. 101–110.

[14] P. Gács, *Reliable cellular automata with self-organization*, in Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science, 1997, pp. 90–99.

[15] P. Gács and A. Gál, *Lower bounds for the complexity of reliable Boolean circuits with noisy gates*, IEEE Trans. Inform. Theory, 40 (1994), pp. 579–583.

[16] R. G. Gallager, *Finding parity in simple broadcast networks*, IEEE Trans. Inform. Theory, 34 (1988), pp. 176–180.

[17] A. Kalai and R. Servedio, *Boosting in the presence of noise*, in Proceedings of the 35th Annual ACM Symposium on Theory of Computing, 2003, pp. 195–205.

[18] D. J. Kleitman, F. T. Leighton, and Y. Ma, *On the design of reliable Boolean circuits that contain partially unreliable gates*, J. Comput. System Sci., 55 (1997), pp. 385–401.

[19] E. Kushilevitz and Y. Mansour, *Computation in noisy radio networks*, in Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 1998, pp. 236–243.

[20] F. T. Leighton and Y. Ma, *Tight bounds on the size of fault-tolerant merging and sorting networks with destructive faults*, SIAM J. Comput., 29 (1999), pp. 258–273.

[21] J. H. van Lint, *Introduction to Coding Theory*, 3rd ed., Springer-Verlag, Berlin, 1999.

[22] I. Newman, *Computing in fault tolerance broadcast networks*, in Proceedings of the 19th Annual IEEE Conference on Computational Complexity, 2004, pp. 113–122.

[23] A. Pelc, *Searching games with errors—fifty years of coping with liars*, Theoret. Comput. Sci., 270 (2002), pp. 71–109.

[24] N. Pippenger, *On the networks of noisy gates*, in Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science, 1985, pp. 30–38.

[25] S. Rajagopalan and L. J. Schulman, *A coding theorem for distributed computing*, in Proceedings of the 26th Annual ACM Symposium on Theory of Computing, 1994, pp. 790–799.

[26] R. Reischuk and B. Schmeltz, *Reliable computation with noisy circuits and decision trees—a general $n \log n$ lower bound*, in Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science, 1991, pp. 602–611.

[27] A. Russell, M. Saks, and D. Zuckerman, *Lower bounds for leader election and collective coin-flipping in the perfect information model*, SIAM J. Comput., 31 (2002), pp. 1645–1662.

[28] L. Schulman, *Coding for interactive communication*, IEEE Trans. Inform. Theory, 42 (1996), pp. 1745–1756.

[29] D. A. Spielman, *Highly fault-tolerant parallel computation*, in Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science, 1996, pp. 154–163.

[30] M. Szegedy and X. Chen, *Computing Boolean functions from multiple faulty copies of input bits*, in LATIN 2002: Theoretical Informatics, Springer-Verlag, Berlin, pp. 539–553.

[31] M. Talagrand, *On Russo's approximate zero-one law*, Ann. Probab., 22 (1994), pp. 1576–1587.

[32] M. Talagrand, *How much are increasing sets positively correlated?*, Combinatorica, 16 (1996), pp. 243–258.

[33] A. Yao, *On the complexity of communication under noise*, in Proceedings of the 5th Israel Symposium on Theory of Computing and Systems, 1997.

SIAM J. COMPUT.
Vol. 37, No. 6, pp. 1842–1864

THE SYMMETRIC GROUP DEFIES STRONG FOURIER SAMPLING*

CRISTOPHER MOORE†, ALEXANDER RUSSELL‡, AND LEONARD J. SCHULMAN§

Abstract. The dramatic exponential speedups of quantum algorithms over their best existing classical counterparts were ushered in by the technique of *Fourier sampling*, introduced by Bernstein and Vazirani and developed by Simon and Shor into an approach to the hidden subgroup problem. This approach has proved successful for abelian groups, leading to efficient algorithms for factoring, extracting discrete logarithms, and other number-theoretic problems. We show, however, that this method cannot resolve the hidden subgroup problem in the symmetric groups, even in the weakest, information-theoretic sense. In particular, we show that the GRAPH ISOMORPHISM problem cannot be solved by this approach. Our work implies that any quantum approach based upon the measurement of coset states must depart from the original framework by using entangled measurements on multiple coset states.

Key words. quantum computing, hidden subgroup problem, graph isomorphism, Fourier sampling

AMS subject classifications. 68Q17, 81R05, 43A65

DOI. 10.1137/050644896

1. Introduction: The hidden subgroup problem.

Many problems of interest in quantum computing can be reduced to an instance of the *hidden subgroup problem* (HSP). We are given a group G and a function f with the promise that, for some subgroup $H \subseteq G$, f is invariant precisely under translation by H; that is, f is constant on the left cosets of H and takes distinct values on distinct cosets. We then wish to determine the subgroup H by querying f. Most algorithms for the HSP use the following approach, referred to as the *standard method* or *Fourier sampling* [5].

Step 1. Prepare two registers, the first in a uniform superposition over the elements of G and the second with the value zero, yielding the state

$$|\psi_1\rangle = \frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle \otimes |0\rangle \quad .$$

Step 2. Query (or calculate) the function f defined on G and XOR it with the second register. This entangles the two registers and results in the state

$$|\psi_2\rangle = \frac{1}{\sqrt{|G|}} \sum_{g \in G} |g\rangle \otimes |f(g)\rangle \quad .$$

Step 3. Measure the second register. This puts the first register in a uniform superposition over one of f's level sets, i.e., one of the left cosets of H, and

*Received by the editors November 11, 2005; accepted for publication (in revised form) November 12, 2007; published electronically March 26, 2008. This work was supported by NSF grants CCR-0093065, PHY-0200909, PHY-0456720, EIA-0218443, EIA-0218563, CCR-0220070, CCR-0220264, CCF-0524828, and CCF-0524613, and ARO grants W911NF-04-R-0009 and W911NF-05-1-0294.

http://www.siam.org/journals/sicomp/37-6/64489.html

†Department of Computer Science, University of New Mexico, Albuquerque, NM 87131, and the Santa Fe Institute, Santa Fe, NM 87501 (moore@cs.unm.edu).

‡Department of Computer Science and Engineering, University of Connecticut, Storrs, CT 06269 (acr@cse.uconn.edu).

§Computer Science Department, California Institute of Technology, Pasadena, CA 91125 (schulman@cs.caltech.edu).

disentangles it from the second register. If we observe the value $f(c)$, we have the state $\psi_3 \otimes |f(c)\rangle$, where

$$|\psi_3\rangle = |cH\rangle = \frac{1}{\sqrt{|H|}} \sum_{h \in H} |ch\rangle \quad .$$

Alternately, we can view the first register as being in a mixed state with density matrix

$$\rho = \frac{1}{|G|} \sum_{g \in G} |cH\rangle \langle cH| \quad .$$

Step 4. Carry out the quantum Fourier transform on $|\psi_3\rangle$ and measure the result; that is, observe the "frequency" corresponding to one of the Fourier basis functions.

For example, in Simon's problem [35], $G = \mathbb{Z}_2^n$ and f is an oracle such that, for some y, $f(x) = f(x+y)$ for all x; in this case $H = \{0, y\}$ and we wish to identify y. In Shor's factoring algorithm [34], G is essentially the group $\mathbb{Z}_n^*$, where n is the number we wish to factor, $f(x) = c^x \bmod n$ for a random $c < n$, and H is the subgroup of $\mathbb{Z}_n^*$ whose index is the multiplicative order of c. (However, Shor's algorithm does not operate on $\mathbb{Z}_n^*$ directly—indeed, knowing $|\mathbb{Z}_n^*|$ would provide an efficient classical algorithm. Instead, it performs the quantum Fourier transform over $\mathbb{Z}_q$ for some $q = \text{poly}(n)$; see [34] or [13, 14].)

In both Simon's and Shor's algorithms, the group G is abelian and finite. It is not hard to see that, in this case, a polynomial number (i.e., polynomial in $\log |G|$) of experiments of this type determine H. In a cyclic group, for instance, the observed frequency is a random multiple of the index of H, so we can determine this index with high probability by taking the greatest common divisor of these frequencies. More generally, each experiment yields a random element of the dual space $H^\perp$ perpendicular to H's characteristic function. After $O(\log |G|)$ such experiments, with high probability these elements span $H^\perp$, and we can determine H via linear algebra.

While the *nonabelian* HSP appears to be much more difficult, it has very attractive applications. In particular, solving the HSP for the symmetric group S_n would provide an efficient quantum algorithm for the Graph Automorphism and Graph Isomorphism problems (see, e.g., Jozsa [21] for a review). Another important motivation is the relationship between the HSP over the dihedral group with hidden shift problems [7] and cryptographically important cases of the shortest lattice vector problem [29].

So far, algorithms for the HSP are known for only a few families of nonabelian groups, including groups whose commutator subgroup is of polynomial size [30, 20]; "smoothly solvable" groups [10]; and some semidirect products of abelian groups [28, 18, 3]. Ettinger and Høyer [8] provided another type of result by showing that Fourier sampling can solve the HSP for the dihedral groups D_n in an *information-theoretic* sense. That is, a polynomial number of experiments gives enough information to reconstruct the subgroup, though it is unfortunately not known how to determine H from this information in polynomial time.

Extending the notion of Fourier sampling to nonabelian groups requires that we define a nonabelian version of the Fourier transform. For abelian groups, the Fourier basis functions are simply the homomorphisms $\phi : G \to \mathbb{C}$ such as the familiar exponential function $\phi_k(x) = e^{2\pi i k x/n}$ for the cyclic group $\mathbb{Z}_n$. In the nonabelian case, there are not enough such homomorphisms to provide a basis for all $\mathbb{C}$-valued functions on G. To create such a basis, we generalize to the *representations* of the group, namely, homomorphisms $\rho : G \to \mathsf{U}(V)$, where $\mathsf{U}(V)$ is the group of unitary matrices acting on some $\mathbb{C}$-vector space V of dimension d_ρ. It suffices to consider *irreducible* representations, namely, those for which no nontrivial subspace of V is fixed by the various operators $\rho(g)$; Fourier analysis over abelian groups then corresponds to the special case where all irreducible representations have dimension one, the single entry in these 1×1 matrices being the values of the Fourier basis functions. In general, once a basis for each irreducible ρ is chosen, the matrix elements ρ_{ij} provide an orthogonal basis for the vector space of all $\mathbb{C}$-valued functions on G.

The quantum Fourier transform then consists of transforming (unit-length) vectors in $\mathbb{C}[G] = \{\sum_{g \in G} \alpha_g |g\rangle \mid \alpha_g \in \mathbb{C}\}$ from the basis $\{|g\rangle \mid g \in G\}$ to the basis $\{|\rho, i, j\rangle\}$, where ρ is the name of an irreducible representation and $1 \leq i, j \leq d_\rho$ index a row and a column (in a chosen basis for V). Note, however, that a representation $\rho : G \to \mathsf{U}(V)$ does not intrinsically distinguish any specific basis for the underlying space V and, for high-dimensional representations, this appears to require a rather dramatic choice on the part of the transform designer. For instance, in a group such as S_n, in most bases a typical representation $\rho(g)$ is a dense matrix of exponential size, but for a carefully chosen basis it is sparse and highly structured. Making such choices of bases allows us to efficiently carry out the quantum Fourier transform for a wide variety of groups [4, 17, 27].

Since the work of [15, 12], the most fundamental question concerning the HSP has been whether there is a basis for the irreducible representations of a given group such that measuring coset states in this basis provides enough information to determine H and, if so, whether this information can be extracted by an efficient algorithm. This framework is known as *strong Fourier sampling*. In this article, we answer this question in the negative for the symmetric group S_n, showing that this process cannot distinguish relevant subgroups from each other, or from the trivial subgroup, even information-theoretically. Indeed, we show that no measurement whatsoever, including arbitrary positive operator-valued measurements (POVMs), on single coset states can succeed. We remark that the subgroups on which we focus are among the most important special cases of the HSP, as they are those to which Graph Isomorphism naturally reduces.

Related work. The terminology "strong Fourier sampling" [12] was invented to distinguish this approach from the natural variant, called *weak Fourier sampling*, where one only measures the name of the representation ρ and ignores the row and column information. Weak Fourier sampling is basis-independent, making it attractive from the standpoint of analysis; however, it cannot distinguish conjugate subgroups from each other, and Hallgren, Russell, and Ta-Shma [15] showed that it cannot distinguish the trivial subgroup from an order-2 subgroup of S_n consisting of $n/2$ disjoint transpositions. Specifically, they used character bounds to show that the probability distributions obtained on representation names for the trivial and order-2 subgroups are exponentially close in total variation distance: thus one needs an exponential number of such experiments to distinguish them. Kempe and Shalev [22] have generalized this result to other conjugacy classes and conjectured that one can do no better than classical computation with this approach.

In an effort to shed light on the power of strong Fourier sampling, Grigni et al. [12] showed that, for groups such as S_n, measuring in a *random* basis yields an exponentially small amount of information. This can be explained, roughly, by the fact that projecting a vector into a sufficiently high-dimensional random subspace results in tightly concentrated length. On the other hand, Moore et al. [28] showed that for the affine groups and some q-hedral groups, measuring in a well-chosen basis can solve the HSP in cases where random bases cannot.

Our contribution. In this paper we show that strong Fourier sampling, in an arbitrary basis of the algorithm designer's choice, cannot solve the HSP for S_n. As in [15] we focus on order-2 subgroups of the form $\{1, m\}$, where m is an involution consisting of $n/2$ disjoint transpositions. We show that strong Fourier sampling—and more generally, arbitrary measurements of single coset states—cannot distinguish most subgroups of this form from each other, or from the trivial subgroup, without an exponential number of experiments.

The motivation for looking at this case of the HSP is as follows. If we fix two rigid connected graphs of size n, then the automorphism group H of their disjoint union is a subgroup of S_{2n}. If they are isomorphic, then H is of the form $\{1, m\}$, where m is the involution that swaps the two graphs, while if they are nonisomorphic, then H is trivial. This yields a classical reduction from Graph Isomorphism to Graph Automorphism, and our results preclude a quantum algorithm for the latter problem that works by reducing to the HSP on the symmetric group.

However, the involutions m which switch the two graphs are not generic elements of the conjugacy class in S_{2n} consisting of n disjoint transpositions, since they switch the first n vertices with the last n vertices. The set of such elements forms a conjugacy class in the *wreath product* $S_n \wr \mathbb{Z}_2 \subset S_{2n}$, and it is the HSP on this group, rather than all of S_{2n}, to which Graph Isomorphism naturally reduces. To address the possibility of a quantum algorithm that uses this reduction, we present an additional result showing that this case of the HSP also requires an exponential number of experiments.

We remark that our results do not preclude the existence of an efficient quantum algorithm for the HSP on S_n or $S_n \wr \mathbb{Z}_2$. Rather, they force us to either abandon coset states or consider *multiregister* algorithms, in which we prepare multiple coset states and subject them to entangled measurements, rather than performing a product measurement where each coset state is measured independently. Some progress in this direction has been made: Ettinger, Høyer, and Knill [9] showed that the HSP on arbitrary groups can be solved information-theoretically with a polynomial number of coset states, and two of the present authors have shown how to carry out such a measurement in the Fourier basis [25]. Kuperberg [24] devised a subexponential ($2^{O(\sqrt{\log n})}$) algorithm for the HSP on the dihedral group D_n that uses entangled measurements, and Alagic, Moore, and Russell [1] obtained a similar algorithm for the HSP on groups of the form G^n. Bacon, Childs, and van Dam determined the *optimal* multiregister measurement for the dihedral group [2] (see also [26]) and used this approach to construct an algorithm for a class of semidirect product groups [3].

Whether a similar approach can be applied to the symmetric group is a major open question. Hallgren et al. [16] have shown, however, that no family of measurements across $o(n \log n)$ coset states can distinguish $H = \{1, m\}$ from the trivial group in S_n with a polynomial number of repetitions. We remark that in light of the upper bounds of [9, 25], $O(n \log n)$ coset states do, at least information-theoretically, determine the answer. Constructing such highly entangled measurements poses a major conceptual challenge, and it is far from clear in what cases they can be carried out efficiently. Of course, it is also possible that a completely different approach—one which does not use coset states, or which does not start by reducing to the HSP—will provide an efficient quantum algorithm for Graph Isomorphism.

The paper is organized as follows. In section 2 we give a brief introduction to representation theory and nonabelian Fourier analysis. In section 3 we discuss the general structure of quantum measurements on coset states and show that the optimal measurement takes the form of strong Fourier sampling. In section 4 we show how to bound the variance of the resulting probability distributions with respect to the choice of hidden subgroup. In section 5 we record some specific facts about the representations of the symmetric group, and in section 6 we use these facts to show that an exponential number of measurements are necessary. Finally, in section 7 we adapt the argument for the specific family of involutions relevant to Graph Isomorphism.

2. Fourier analysis over finite groups.

We briefly discuss the elements of the representation theory of finite groups. Our treatment is primarily for the purposes of setting down notation; we refer the reader to [11, 33] for complete accounts.

Let G be a finite group. A *representation* ρ of G is a homomorphism $\rho : G \to \mathsf{U}(V)$, where V is a finite-dimensional Hilbert space and $\mathsf{U}(V)$ is the group of unitary operators on V. The *dimension* of ρ, denoted d_ρ, is the dimension of the vector space V. By choosing a basis for V, we can then identify $\rho(g)$ with a unitary $d_\rho \times d_\rho$ matrix so that for every $g, h \in G$, $\rho(gh) = \rho(g) \cdot \rho(h)$.

Fixing a representation $\rho : G \to \mathsf{U}(V)$, we say that a subspace $W \subset V$ is *invariant* if $\rho(g)W \subset W$ for all $g \in G$. We say ρ is *irreducible* if it has no invariant subspaces other than the trivial space $\{\mathbf{0}\}$ and V. If two representations ρ and σ are the same up to a unitary change of basis, we say that they are *equivalent*. It is a fact that any finite group G has a finite number of distinct irreducible representations up to equivalence, and, for a group G, we let $\widehat{G}$ denote a set of representations containing exactly one from each equivalence class. The irreducible representations of G give rise to the Fourier transform. Specifically, for a function $f : G \to \mathbb{C}$ and an element $\rho \in \widehat{G}$, define the *Fourier transform of f at ρ* to be

$$\hat{f}(\rho) = \sqrt{\frac{d_\rho}{|G|}} \sum_{g \in G} f(g)\rho(g) \enspace .$$

The leading coefficients are chosen to make the transform unitary, so that it preserves inner products:

$$\langle f_1, f_2 \rangle = \sum_g f_1^*(g) f_2(g) = \sum_{\rho \in \widehat{G}} \mathbf{tr} \left(\hat{f}_1(\rho)^\dagger \cdot \hat{f}_2(\rho) \right) \enspace .$$

Given a representation ρ and pair of integers $1 \leq i, j \leq d_\rho$, we can associate a basis vector $|\rho, i, j\rangle$, which assigns the matrix entry $\rho(g)_{i,j}$ to each element g. As described above, these form an orthonormal basis for $\mathbb{C}[G]$, which implies

$$\sum_{\rho \in \widehat{G}} d_\rho^2 = |G| \enspace .$$

In the case when ρ is *not* irreducible, it can be decomposed into a direct sum of irreducible representations, each of which operates on an invariant subspace. We write $\rho = \sigma_1 \oplus \cdots \oplus \sigma_k$ and, for the σ_i appearing at least once in this decomposition, $\sigma_i \prec \rho$. In general, a given σ can appear multiple times, in the sense that ρ can have an invariant subspace isomorphic to the direct sum of a_σ^ρ copies of σ. In this case a_σ^ρ is called the *multiplicity* of σ in ρ, and we write $\rho = \bigoplus_{\sigma \prec \rho} a_\sigma^\rho \sigma$.

For a representation ρ we define its *character* as the trace $\chi_\rho(g) = \mathbf{tr}\, \rho(g)$. Since the trace is invariant under conjugation, characters are constant on the conjugacy classes, and if $\mathfrak{m}$ is a conjugacy class, we write $\chi_\rho(\mathfrak{m}) = \chi_\rho(m)$, where m is any element of $\mathfrak{m}$. Characters are a powerful tool for reasoning about the decomposition of reducible representations. In particular, for $\rho, \sigma \in \widehat{G}$, we have the orthogonality conditions

$$\langle \chi_\rho, \chi_\sigma \rangle_G = \frac{1}{|G|} \sum_{g \in G} \chi_\rho(g) \chi_\sigma(g)^* = \begin{cases} 1, & \rho = \sigma \ , \\ 0, & \rho \neq \sigma \ . \end{cases}$$

If ρ is reducible, we have $\chi_\rho = \sum_{\sigma \prec \rho} a_\sigma^\rho \chi_{\sigma_i}$, and so the multiplicity a_σ^ρ is given by

$$a_\sigma^\rho = \langle \chi_\rho, \chi_\sigma \rangle_G \ .$$

If ρ is irreducible, *Schur's lemma* asserts that the only matrices which commute with $\rho(g)$ for all g are the scalars, $\{c\mathbb{1} \mid c \in \mathbb{C}\}$. Therefore, for any A we have

$$\frac{1}{|G|} \sum_{g \in G} \rho(g)^\dagger A \rho(g) = \frac{\mathbf{tr}\, A}{d_\rho} \mathbb{1}_{d_\rho} \tag{2.1}$$

since conjugating this sum by $\rho(g)$ simply permutes its terms. In particular, consider the average of ρ over a conjugacy class $\mathfrak{m}$, which we denote $\rho(\mathfrak{m})$:

$$\rho(\mathfrak{m}) = \mathrm{Exp}_{m \in \mathfrak{m}}\, \rho(m) = \mathrm{Exp}_g\, \rho(g^{-1} m g) = \mathrm{Exp}_g\, \rho(g)^\dagger \rho(m) \rho(g) \ .$$

Then since $\mathbf{tr}\, \rho(m) = \chi_\rho(\mathfrak{m})$, we have

$$\rho(\mathfrak{m}) = \frac{\chi(\mathfrak{m})}{d_\rho} \mathbb{1}_{d_\rho} \ . \tag{2.2}$$

Similarly, if ρ is reducible, $\rho(\mathfrak{m})$ is scalar in each irreducible subspace, giving

$$\rho(\mathfrak{m}) = \sum_{\sigma \prec \rho} \frac{\chi_\sigma(\mathfrak{m})}{d_\sigma} \Pi_\sigma^\rho \ , \tag{2.3}$$

where Π_σ^ρ projects onto the subspace $a_\sigma^\rho \sigma$ spanned by copies of σ. We use these facts below.

There is a natural product operation on representations: if $\rho : G \to \mathsf{U}(V)$ and $\sigma : G \to \mathsf{U}(W)$ are representations of G, we may define a new representation $\rho \otimes \sigma : G \to \mathsf{U}(V \otimes W)$ by extending the rule $(\rho \otimes \sigma)(g) : \mathbf{u} \otimes \mathbf{v} \mapsto \rho(g)\mathbf{u} \otimes \sigma(g)\mathbf{v}$. In general, the representation $\rho \otimes \sigma$ is not irreducible, even when both ρ and σ are. This leads to the *Clebsch–Gordan problem*, that of decomposing $\rho \otimes \sigma$ into irreducibles. For example, since $\chi_{\rho \otimes \sigma}(g) = \chi_\rho(g) \cdot \chi_\sigma(g)$, the multiplicity of τ in $\rho \otimes \sigma$ is $\langle \chi_\tau, \chi_\rho \chi_\sigma \rangle_G$.

Group elements can act on each other on the left or right. Thus we can consider subspaces of $\mathbb{C}[G]$ that are invariant under left multiplication, right multiplication, or both; these subspaces are called *left-*, *right-*, or *bi-invariant*, respectively. Each $\rho \in \widehat{G}$ corresponds to a d_ρ^2-dimensional bi-invariant subspace of $\mathbb{C}[G]$. We can think of the bi-invariant subspace as a single d_ρ^2-dimensional representation, consisting of the space of $d_\rho \times d_\rho$ matrices A. If $\rho(g)$ acts on A by left or right multiplication, the left- and right-invariant subspaces correspond to A's columns and rows, respectively; for instance, each column of A is acted on independently by left multiplication by $\rho(g)$, and the space of matrices A which are nonzero only in this column form a d_ρ-dimensional left-invariant subspace. Thus, each bi-invariant subspace can be decomposed into

d_ρ d_ρ-dimensional left-invariant subspaces, or (transversely) d_ρ d_ρ-dimensional right-invariant subspaces.

However, this decomposition is not unique. If we think of A as the space of linear operators on the same d_ρ-dimensional vector space V on which ρ acts, changing the orthonormal basis for V transforms the matrices A. Thus, each orthonormal basis B of V gives a way to divide the bi-invariant subspace into left-invariant columns and right-invariant rows, and each such subspace is associated with some basis vector $\mathbf{b} \in B$.

3. The structure of the optimal measurement. In this section we show that starting with a single coset state, the optimal measurement for the HSP is precisely an instance of strong Fourier sampling (possibly in an overcomplete basis). This has been pointed out several times in the past, at varying levels of explicitness [19, 24]; we state it here for completeness. Everything we say in this section is true for the HSP in general. However, for simplicity we focus on the special case of the HSP called the *hidden conjugate problem* in [28]: there is a (nonnormal) subgroup H, and we are promised that the hidden subgroup is one of its conjugates, $H^g = g^{-1}Hg$ for some $g \in G$.

We may treat the states arising after Step 3 of the procedure above as elements of the group algebra $\mathbb{C}[G]$. We use the notation $|g\rangle = 1 \cdot g \in \mathbb{C}[G]$ so that the vectors $|g\rangle$ form an orthonormal basis for $\mathbb{C}[G]$. Given a set $S \subset G$, $|S\rangle$ denotes a uniform superposition over the elements of S, $|S\rangle = (1/\sqrt{|S|}) \sum_{s \in S} |s\rangle$.

3.1. The optimal POVM consists of strong Fourier sampling. The most general type of measurement allowed in quantum mechanics is a POVM. A POVM with a set of possible outcomes J consists of a set of positive operators $\{M_j \mid j \in J\}$ subject to the completeness condition,

$$\sum_j M_j = \mathbb{1} \quad . \tag{3.1}$$

Since positive operators are self-adjoint, they can be orthogonally diagonalized, and since their eigenvalues are positive, they can be written as a positive linear combination of projection operators (see, e.g., [32, sect. 10]). Any POVM may thus be refined so that each $M_j = a_j \mu_j$, where μ_j is a projection operator and a_j is positive and real.

The result of this refined measurement on the state $|\psi\rangle$ is a random variable, taking values in J, that is equal to $j \in J$ with probability $P_j = a_j \langle\psi| \mu_j |\psi\rangle$. Note that the outcomes j need not correspond to subgroups directly; the algorithm designer is free to carry out t of these experiments (where t is, ideally, polynomial), observing outcomes $j_1, \ldots, j_t$, and then apply some additional computation to find the most likely subgroup given these observations.

If g is chosen from G uniformly so that the hidden subgroup is a uniformly random conjugate of H, we wish to find a POVM that maximizes the probability of correctly identifying g from the coset state $|H^g\rangle$. (Of course, to identify a conjugate H^g, we need only specify g up to an element of the normalizer of H.) Since a random left coset of H^g can be written $cgH^g = cHg$ for a random $c \in G$, the probability we observe outcome j is

$$P_j = a_j \frac{1}{|G|} \sum_{c \in G} \langle cHg| \mu_j |cHg\rangle \quad . \tag{3.2}$$

Ip [19] observed that in the special case that each outcome j corresponds to a subgroup, maximizing the probability that j is correct subject to the constraint (3.1) gives a semidefinite program. Since such programs are convex, the optimum is unique and is a fixed point of any symmetries possessed by the problem.

However, our proof relies on an elementary "symmetrization" argument. Given a group element $x \in G$, let $L_x \left| g \right\rangle = \left| xg \right\rangle$ denote the unitary matrix corresponding to left group multiplication by x. In particular, applying L_x maps one left coset onto another: $\left| cHg \right\rangle = L_c \left| Hg \right\rangle$. Writing

$$P_j = a_j \frac{1}{|G|} \sum_{c \in G} \left\langle cHg \right| \mu_j \left| cHg \right\rangle = a_j \left\langle Hg \left| \frac{1}{|G|} \sum_{c \in G} L_c^\dagger \mu_j L_c \right| Hg \right\rangle \ ,$$

we conclude that replacing μ_j for each j with the symmetrization

$$\mu_j' = \frac{1}{|G|} \sum_{g \in G} L_g^\dagger \mu_j L_g$$

does not change the resulting probability distribution P_j. Since μ_j' commutes with L_x for every $x \in G$ and provides exactly the same information as the original μ_j, we may assume without loss of generality that the optimal POVM commutes with L_x for every $x \in G$.

It is easy to see that any projection operator that commutes with left multiplication projects onto a left-invariant subspace of $\mathbb{C}[G]$, and we can further refine the POVM so that each μ_j projects onto an *irreducible* left-invariant subspace. Each such space is contained in the bi-invariant subspace corresponding to some irreducible representation ρ, in which case we write $\mathbf{im}\,\mu_j \subseteq \rho$. As discussed in section 2, a given irreducible left-invariant subspace corresponds to some unit vector $\mathbf{b}$ in the vector space V on which ρ acts. Thus we can write

$$\mu_j = \left| \mathbf{b}_j \right\rangle \left\langle \mathbf{b}_j \right| \otimes \mathbb{1}_{d_\rho} \ ,$$

where $\mathbb{1}_{d_\rho}$ is the identity operator on that left-invariant subspace. Let $B = \{\mathbf{b}_j \mid \mathbf{im}\,\mu_j \in \rho\}$; then (3.1) implies a completeness condition for each $\rho \in \widehat{G}$,

$$\sum_{\mathbf{b}_j \in B} a_j \left| \mathbf{b}_j \right\rangle \left\langle \mathbf{b}_j \right| = \mathbb{1}_{d_\rho} \ , \tag{3.3}$$

and so B is a (possibly overcomplete) basis for V. In other words, the optimal POVM consists of first measuring the representation name ρ and then performing a POVM on the vector space V with the set of possible outcomes B. Another way to see this is to regard the choice of coset as a mixed state; then its density matrix is block-diagonal in the Fourier basis, and so as Kuperberg puts it [24], measuring the representation name "sacrifices no entropy."

We note that in the special case that this POVM is a von Neumann measurement—that is, when B is an orthonormal basis for V—it corresponds to measuring the column of ρ in that basis, which is how strong Fourier sampling is usually defined. (As pointed out in [12], nothing is gained by measuring the row, since we have a random left coset cHg and left-multiplying by a random element c in an irreducible representation completely mixes the probability across the rows in each column. Here this is reflected by the fact that each μ_j is a scalar in its left-invariant subspace.) However, in general the optimal measurement might consist of an overcomplete basis, or *frame*, in each ρ, consisting of vectors $\mathbf{b}_j$ with weights a_j.

Now that we know μ_j takes this form, let us change notation. Given $\rho \in \widehat{G}$ acting on a vector space V and a unit vector $\mathbf{b} \in V$, let $\Pi^\rho_{\mathbf{b}} = |\mathbf{b}\rangle\langle\mathbf{b}| \otimes \mathbb{1}_{d_\rho}$ denote the projection operator onto the left-invariant subspace corresponding to $\mathbf{b}$. Then $\mu_j = \Pi^\rho_{\mathbf{b}_j}$, and (3.2) becomes

$$P_j = a_j \frac{1}{|G|} \sum_{c \in G} \left\| \Pi^\rho_{\mathbf{b}_j} |cHg\rangle \right\|^2 = a_j \left\| \Pi^\rho_{\mathbf{b}_j} |Hg\rangle \right\|^2 \quad . \tag{3.4}$$

We can write this as the product of the probability $P(\rho)$ that we observe ρ, times the conditional probability $P(\rho, \mathbf{b}_j)$ that we observe $\mathbf{b}_j$. Note that by (3.3),

$$\Pi^\rho = \sum_{\mathbf{b}_j \in B} a_j \Pi^\rho_{\mathbf{b}_j}$$

is the projection operator onto the bi-invariant subspace corresponding to ρ. Then

$$P_j = P(\rho) P(\rho, \mathbf{b}_j) \ ,$$

where

$$P(\rho) = \| \Pi^\rho |H\rangle \|^2 \quad , \tag{3.5}$$

$$P(\rho, \mathbf{b}_j) = a_j \left\| \Pi^\rho_{\mathbf{b}_j} |Hg\rangle \right\|^2 \Big/ P(\rho) \quad . \tag{3.6}$$

Note that $P(\rho, \mathbf{b}_j)$ depends on g, but $P(\rho)$ does not, which is why weak sampling is incapable of distinguishing conjugate subgroups.

3.2. The probability distribution for a conjugate subgroup. Now let us use the fact that $|H\rangle$ is a superposition over a subgroup and calculate $P(\rho)$ and $P(\rho, \mathbf{b}_j)$ as defined in (3.5) and (3.6). This will set the stage for asking whether we can distinguish different conjugates of H from each other or from the trivial subgroup.

Fix an irreducible representation ρ that acts on a vector space V. Then Fourier transforming the state

$$|H\rangle = \frac{1}{\sqrt{|H|}} \sum_{h \in H} |h\rangle$$

yields the coefficient

$$\widehat{H}(\rho) = \sqrt{\frac{d_\rho}{|H||G|}} \sum_{h \in H} \rho(h) = \sqrt{\frac{d_\rho |H|}{|G|}} \, \Pi_H \quad ,$$

where $\Pi_H = (1/|H|) \sum_{h \in H} \rho(h)$ is a projection operator onto a subspace of V. The probability that we observe ρ is then the norm squared of this coefficient,

$$P(\rho) = \left\| \widehat{H}(\rho) \right\|^2 = \frac{d_\rho |H|}{|G|} \, \mathbf{rk}\, \Pi_H \quad , \tag{3.7}$$

and, as stated above, this is the same for all conjugates H^g. The conditional probability that we observe the vector $\mathbf{b}_j$, given that we observe ρ, is then

$$P(\rho, \mathbf{b}_j) = a_j \frac{\left\| \Pi^\rho_{\mathbf{b}_j} |H\rangle \right\|^2}{P(\rho)} = a_j \frac{\left\| \widehat{H}(\rho) \mathbf{b}_j \right\|^2}{P(\rho)} = a_j \frac{\| \Pi_H \mathbf{b}_j \|^2}{\mathbf{rk}\, \Pi_H} \quad . \tag{3.8}$$

In the case where H is the trivial subgroup, $\Pi_H = \mathbb{1}_{d_\rho}$ and $P(\rho, \mathbf{b}_j)$ is given by

$$P(\rho, \mathbf{b}_j) = \frac{a_j}{d_\rho} \ . \tag{3.9}$$

We call this the *natural distribution* on the frame $B = \{\mathbf{b}_j\}$. In the case that B is an orthonormal basis, $a_j = 1$ and $P(\rho, \mathbf{b}_j)$ is simply the uniform distribution on B.

This probability distribution over B changes for a conjugate H^g in the following way. The Fourier transform of $|Hg\rangle$ is

$$\widehat{Hg}(\rho) = \sqrt{\frac{d_\rho |H|}{|G|}}\, \Pi_H \rho(g) \ ,$$

and we have

$$P(\rho, \mathbf{b}_j) = a_j \frac{\|\Pi_H(g\mathbf{b}_j)\|^2}{\mathbf{rk}\, \Pi_H} \ , \tag{3.10}$$

where we write $g\mathbf{b}$ for $\rho(g)\mathbf{b}$.

Our goal is to understand, for each fixed $\mathbf{b}$, to what extent $P(\rho, \mathbf{b})$ varies with g, and so to what extent measurements of this type can distinguish the conjugates H^g from each other. Regarding this as a random variable over the choice of g, its expectation is easy to calculate: we have

$$\operatorname{Exp}_g \|\Pi_H(g\mathbf{b})\|^2 = \operatorname{Exp}_g \left\langle \mathbf{b}, \rho(g)^\dagger \Pi_H \rho(g) \mathbf{b} \right\rangle = \left\langle \mathbf{b}, \left(\operatorname{Exp}_g \rho(g)^\dagger \Pi_H \rho(g)\right) \mathbf{b} \right\rangle = \frac{\mathbf{rk}\, \Pi_H}{d_\rho} \ ,$$

where we used (2.1), $\|\mathbf{b}\|^2 = 1$, and the fact that the trace of a projection operator is its rank. Combining this with (3.10), the expected probability is simply the natural distribution (3.9),

$$\operatorname{Exp}_g P(\rho, \mathbf{b}_j) = \frac{a_j}{d_\rho} \ .$$

We wish to show that $\|\Pi_H(g\mathbf{b})\|^2$, and therefore $P(\rho, \mathbf{b}_j)$, is in fact very close to its expectation for most conjugates. In the next section, we present our primary technical contribution, which is a method for establishing concentration results for this random variable.

4. The variance of projection through a random involution. In this section we focus on the case where $H = \{1, m\}$ for an element m chosen uniformly at random from a fixed conjugacy class $\mathfrak{m}$ of involutions. (Observe that order is preserved under conjugation so that if m is an involution, then so are all elements of $\mathfrak{m}$.) Given an irreducible representation $\rho : G \to \mathsf{U}(V)$ and a vector $\mathbf{b} \in V$, we bound the variance, over the choice of $m \in \mathfrak{m}$, of the probability $P_m(\rho, \mathbf{b})$ that $\mathbf{b}$ is observed given that we observed ρ. Our key insight is that this variance depends on how the tensor product representation $\rho \otimes \rho^*$ decomposes into irreducible representations σ, and how the vector $\mathbf{b} \otimes \mathbf{b}^*$ projects into these constituent subspaces.

Recall that, if a representation ρ is reducible, it can be written as an orthogonal direct sum of irreducibles $\rho = \bigoplus_{\sigma \prec \rho} a^\rho_\sigma \sigma$, where a^ρ_σ is the multiplicity of σ. We let Π^ρ_σ denote the projection operator whose image is $a^\rho_\sigma \sigma$, that is, the span of all the irreducible subspaces isomorphic to σ.

Lemma 4.1. *Let ρ be a representation of a group G acting on a space V and let $\mathbf{b} \in V$. Let m be an element chosen uniformly from a conjugacy class $\mathfrak{m}$ of involutions. If ρ is irreducible, then*

$$\mathrm{Exp}_{m \in \mathfrak{m}} \langle \mathbf{b}, m\mathbf{b} \rangle = \frac{\chi_\rho(\mathfrak{m})}{d_\rho} \|\mathbf{b}\|^2 \quad .$$

If ρ is reducible, then

$$\mathrm{Exp}_{m \in \mathfrak{m}} \langle \mathbf{b}, m\mathbf{b} \rangle = \sum_{\sigma \prec \rho} \frac{\chi_\sigma(\mathfrak{m})}{d_\sigma} \|\Pi_\sigma^\rho \mathbf{b}\|^2 \quad .$$

Proof. Let $\rho(\mathfrak{m})$ denote the average of ρ over the conjugacy class $\mathfrak{m}$. Using (2.2), we have

$$\mathrm{Exp}_m \langle \mathbf{b}, m\mathbf{b} \rangle = \langle \mathbf{b}, \rho(\mathfrak{m})\mathbf{b} \rangle = \frac{\chi_\rho(\mathfrak{m})}{d_\rho} \|\mathbf{b}\|^2 \quad .$$

Similarly, if ρ is reducible, by (2.3) we have

$$\mathrm{Exp}_m \langle \mathbf{b}, m\mathbf{b} \rangle = \langle \mathbf{b}, \rho(\mathfrak{m})\mathbf{b} \rangle = \sum_{\sigma \prec \rho} \frac{\chi_\sigma(\mathfrak{m})}{d_\sigma} \langle \mathbf{b}, \Pi_\sigma^\rho \mathbf{b} \rangle = \sum_{\sigma \prec \rho} \frac{\chi_\sigma(\mathfrak{m})}{d_\sigma} \|\Pi_\sigma^\rho \mathbf{b}\|^2 \quad . \qquad \square$$

Turning now to the second moment of $\langle \mathbf{b}, m\mathbf{b} \rangle$, we observe that

$$|\langle \mathbf{b}, m\mathbf{b} \rangle|^2 = \langle \mathbf{b}, m\mathbf{b} \rangle \langle \mathbf{b}, m\mathbf{b} \rangle^* = \langle \mathbf{b} \otimes \mathbf{b}^*, m\mathbf{b} \otimes m\mathbf{b}^* \rangle = \langle \mathbf{b} \otimes \mathbf{b}^*, m(\mathbf{b} \otimes \mathbf{b}^*) \rangle \quad ,$$

where the action of m on the vector $\mathbf{b} \otimes \mathbf{b}^*$ is precisely given by the action of G in the representations $\rho \otimes \rho^*$. This will allow us to express the second moment of the inner product $\langle \mathbf{b}, m\mathbf{b} \rangle$ in terms of the projections of $\mathbf{b} \otimes \mathbf{b}^*$ into the irreducible constituents of the tensor product representation $\rho \otimes \rho^*$.

Lemma 4.2. *Let ρ be a representation of a group G acting on a space V and let $\mathbf{b} \in V$. Let m be an element chosen uniformly at random from a conjugacy class $\mathfrak{m}$ of involutions. Then*

$$\mathrm{Exp}_{m \in \mathfrak{m}} |\langle \mathbf{b}, m\mathbf{b} \rangle|^2 = \sum_{\sigma \prec \rho \otimes \rho^*} \frac{\chi_\sigma(\mathfrak{m})}{d_\sigma} \left\| \Pi_\sigma^{\rho \otimes \rho^*} (\mathbf{b} \otimes \mathbf{b}^*) \right\|^2 \quad .$$

Proof. We write the second moment as a first moment over the product representation $\rho \otimes \rho^*$: as above, $|\langle \mathbf{b}, m\mathbf{b} \rangle|^2 = \langle \mathbf{b} \otimes \mathbf{b}^*, m(\mathbf{b} \otimes \mathbf{b}^*) \rangle$ so that

$$\mathrm{Exp}_m |\langle \mathbf{b}, m\mathbf{b} \rangle|^2 = \mathrm{Exp}_m \langle \mathbf{b} \otimes \mathbf{b}^*, m(\mathbf{b} \otimes \mathbf{b}^*) \rangle \quad ,$$

and applying Lemma 4.1 completes the proof. $\square$

Now let $\Pi_m = \Pi_H$ denote the projection operator given by

$$\Pi_m \mathbf{v} = \frac{\mathbf{v} + m\mathbf{v}}{2} \quad .$$

For a given vector $\mathbf{b} \in B$, we will focus on the expectation and variance of $\|\Pi_m \mathbf{b}\|^2$. These are given by the following lemma.

Lemma 4.3. *Let ρ be an irreducible representation acting on a space V and let $\mathbf{b} \in V$. Let m be an element chosen uniformly at random from a conjugacy class $\mathfrak{m}$ of involutions. Then*

$$\operatorname{Exp}_{m\in\mathfrak{m}} \|\Pi_m \mathbf{b}\|^2 = \frac{1}{2}\|\mathbf{b}\|^2 \left(1 + \frac{\chi_\rho(\mathfrak{m})}{d_\rho}\right) , \tag{4.1}$$

$$\operatorname{Var}_{m\in\mathfrak{m}} \|\Pi_m \mathbf{b}\|^2 \le \frac{1}{4} \sum_{\sigma \prec \rho\otimes\rho^*} \frac{\chi_\sigma(\mathfrak{m})}{d_\sigma} \left\|\Pi_\sigma^{\rho\otimes\rho^*}(\mathbf{b}\otimes\mathbf{b}^*)\right\|^2 . \tag{4.2}$$

Proof. For the expectation,

$$\begin{aligned}\operatorname{Exp}_m \|\Pi_m \mathbf{b}\|^2 &= \operatorname{Exp}_m \langle \mathbf{b}, \Pi_m \mathbf{b}\rangle \\ &= \frac{1}{2}\operatorname{Exp}_m \left(\langle \mathbf{b},\mathbf{b}\rangle + \langle \mathbf{b}, m\mathbf{b}\rangle\right) \\ &= \frac{1}{2}\|\mathbf{b}\|^2\left(1 + \frac{\chi_\rho(\mathfrak{m})}{d_\rho}\right) ,\end{aligned}$$

where the last equality follows from Lemma 4.1.

For the variance, we first calculate the second moment,

$$\begin{aligned}\operatorname{Exp}_m \|\Pi_m \mathbf{b}\|^4 &= \operatorname{Exp}_m |\langle \mathbf{b}, \Pi_m \mathbf{b}\rangle|^2 \\ &= \frac{1}{4}\operatorname{Exp}_m |\langle \mathbf{b},\mathbf{b}\rangle + \langle \mathbf{b}, m\mathbf{b}\rangle|^2 \\ &= \frac{1}{4}\operatorname{Exp}_m \left(|\langle \mathbf{b},\mathbf{b}\rangle|^2 + 2\operatorname{Re}\langle \mathbf{b},\mathbf{b}\rangle\langle \mathbf{b}, m\mathbf{b}\rangle + |\langle \mathbf{b}, m\mathbf{b}\rangle|^2\right) \\ &= \frac{1}{4}\left(\|\mathbf{b}\|^4 + 2\|\mathbf{b}\|^4 \frac{\chi_\rho(\mathfrak{m})}{d_\rho} + \sum_{\sigma\prec\rho\otimes\rho^*} \frac{\chi_\sigma(\mathfrak{m})}{d_\sigma}\left\|\Pi_\sigma^{\rho\otimes\rho^*}(\mathbf{b}\otimes\mathbf{b}^*)\right\|^2\right) ,\end{aligned}$$

where in the last line we applied Lemmas 4.1 and 4.2 and the fact that any character evaluated at an involution is real. Then

$$\begin{aligned}\operatorname{Var}_m \|\Pi_m \mathbf{b}\|^2 &= \operatorname{Exp}_m \|\Pi_m \mathbf{b}\|^4 - \left(\operatorname{Exp}_m \|\Pi_m \mathbf{b}\|^2\right)^2 \\ &= \frac{1}{4}\left[\sum_{\sigma\prec\rho\otimes\rho^*} \frac{\chi_\rho(\mathfrak{m})}{d_\rho}\left\|\Pi_\sigma^{\rho\otimes\rho^*}(\mathbf{b}\otimes\mathbf{b}^*)\right\|^2 - \|\mathbf{b}\|^4\left(\frac{\chi_\rho(\mathfrak{m})}{d_\rho}\right)^2\right] . \qquad (4.3)\end{aligned}$$

Ignoring the second term, which is negative, gives the stated result. □

Finally, we point out that since

$$\operatorname{Exp}_m \|\Pi_m \mathbf{b}\|^2 = \|\mathbf{b}\|^2 \frac{\mathbf{rk}\,\Pi_m}{d_\rho} ,$$

we have

$$\frac{\mathbf{rk}\,\Pi_m}{d_\rho} = \frac{1}{2}\left(1 + \frac{\chi_\rho(\mathfrak{m})}{d_\rho}\right) , \tag{4.4}$$

a fact which we will use below.

5. The representation theory of the symmetric group. In this section we record the particular properties of S_n and its representation theory which we apply in the proofs of our main results. The irreducible representations of S_n are labeled by Young diagrams or, equivalently, by integer partitions of n,

$$\lambda = (\lambda_1, \ldots, \lambda_t) \ ,$$

where $\sum_i \lambda_i = n$ and $\lambda_i \geq \lambda_{i+1}$ for all i. The number of Young diagrams, equal to the number of conjugacy classes in S_n, is the partition number $p(n)$, which obeys

$$p(n) = (1 + o(1)) \frac{1}{4\sqrt{3} \cdot n} \, e^{\delta\sqrt{n}} < e^{\delta\sqrt{n}}, \quad \text{where} \quad \delta = \pi\sqrt{2/3} \ . \tag{5.1}$$

We identify each irreducible representation with its Young diagram λ, and denote its character χ_λ and its dimension d_λ. In particular, λ is the trivial or parity representation if λ is a single row (n) or a single column $(1, \ldots, 1)$, respectively. Given λ, its *conjugate* λ' is obtained by flipping λ about the diagonal: $\lambda' = (\lambda'_1, \ldots, \lambda'_{\lambda_1})$, where $\lambda'_j = |\{i \mid \lambda_i \geq j\}|$. In particular, $\lambda'_1 = t$. The representation λ' is the (tensor) product of λ with the parity representation.

The dimension of λ is given by the remarkable *hook length formula*:

$$d_\lambda = \frac{n!}{\prod_c \mathrm{hook}(c)} \ ,$$

where this product runs over all cells of the Young diagram associated with λ and hook(c) is the number of cells appearing in either the same column or row as c, excluding those that are above or to the left of c.

For example, the partition $\lambda = (\lambda_1, \lambda_2, \lambda_3, \lambda_4) = (6, 5, 3, 2)$ is associated with the diagram shown in Figure 5.1 below. The hook associated with the cell $(2, 2)$ in this diagram appears in Figure 5.2; it has length 6.

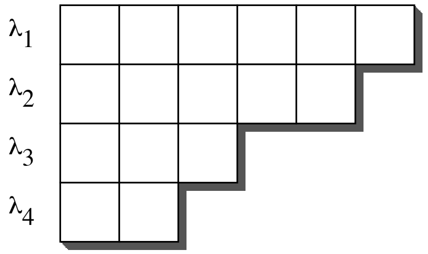


FIG. 5.1. *The Young diagram for* $\lambda = (6, 5, 3, 2)$.

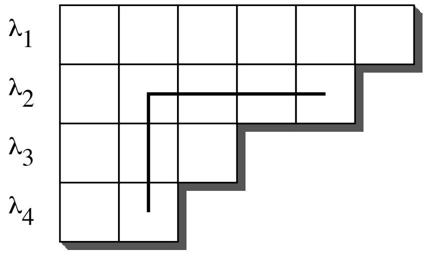


FIG. 5.2. *A hook of length* 6.

The symmetric groups have the property that every representation λ possesses a basis in which its matrix elements are real, and so all its characters are real. However, in a given basis λ might be complex, so we will refer below to its complex conjugate, the representation λ^* (not to be confused with λ').

The study of the asymptotic properties of the representations of S_n typically focuses on the *Plancherel* distribution (see, e.g., Kerov's monograph [23]). For a general group G, this is the probability distribution obtained on $\widehat{G}$ by assigning ρ the probability density $d_\rho^2/|G|$. One advantage of this distribution is that the density at

ρ is proportional to its contribution, dimensionwise, to the group algebra $\mathbb{C}[G]$. Note that in the context of the HSP, the Plancherel distribution is exactly the one obtained by performing weak Fourier sampling on the trivial hidden subgroup.

In the symmetric groups a fair amount is known about representations chosen according to the Plancherel distribution. In particular, Vershik and Kerov [36] have given the following result, showing that with high probability they have dimension equal to $e^{\Theta(\sqrt{n})}\sqrt{n!}$.

THEOREM 5.1 (see [36]). *Let λ be chosen from $\widehat{S_n}$ according to the Plancherel distribution. Then there exist positive constants c_1 and c_2 for which*

$$\lim_{n\to\infty} \Pr\left[e^{-c_1\sqrt{n}}\sqrt{n!} \leq d_\lambda \leq e^{-c_2\sqrt{n}}\sqrt{n!}\right] = 1 \ .$$

Vershik and Kerov have also obtained estimates for the *maximum* dimension of a representation in $\widehat{S_n}$.

THEOREM 5.2 (see [36]). *There exist positive constants $\check{c}$ and $\hat{c}$ such that for all $n \geq 1$,*

$$e^{-\check{c}\sqrt{n}}\sqrt{n!} \leq \max_{\lambda\in\widehat{S_n}} d_\lambda \leq e^{-\hat{c}\sqrt{n}}\sqrt{n!} \ .$$

Along with these estimates, we will use the following (one-sided) large-deviation versions of Theorem 5.1.

LEMMA 5.3. *Let λ be chosen according to the Plancherel distribution on $\widehat{S_n}$.*

1. *Let $\delta = \pi\sqrt{2/3}$ as in* (5.1). *Then for sufficiently large n,*

$$\Pr\left[d_\lambda \leq e^{-\delta\sqrt{n}}\sqrt{n!}\right] < e^{-\delta\sqrt{n}} \ .$$

2. *Let $0 < c < 1/2$. Then there is a constant $\gamma > 0$ such that*

$$\Pr[d_\lambda \leq n^{cn}] < n^{-\gamma n} \ .$$

Proof. For the first bound, setting $d = e^{-\delta\sqrt{n}}\sqrt{n!}$ and using (5.1), we have

$$\sum_{\lambda : d_\lambda \leq d} \frac{d_\lambda^2}{n!} \leq p(n)\frac{d^2}{n!} < e^{-\delta\sqrt{n}} \ .$$

For the second bound, recalling Stirling's approximation $n! > n^n e^{-n}$, we have

$$\sum_{\lambda : d_\lambda \leq n^{cn}} \frac{d_\lambda^2}{n!} \leq \frac{p(n)n^{2cn}}{n!} = n^{-(1-2c)n}e^{O(n)} \ ,$$

and setting $\gamma < 1 - 2c$ completes the proof. ∎

Finally, we will also apply Roichman's estimates [31] for the characters of the symmetric group.

DEFINITION 5.4. *For a permutation $\pi \in S_n$, define the* support *of π, denoted* supp(π), *to be the cardinality of the set $\{k \in [n] \mid \pi(k) \neq k\}$.*

THEOREM 5.5 (see [31]). *There exist constants $b > 0$ and $0 < q < 1$ so that for $n > 4$, for every conjugacy class C of S_n, and for every irreducible representation λ of S_n,*

$$\left|\frac{\chi_\lambda(C)}{d_\lambda}\right| \leq \left(\max\left(q, \frac{\lambda_1}{n}, \frac{\lambda_1'}{n}\right)\right)^{b\cdot\mathrm{supp}(C)} \ ,$$

where $\operatorname{supp}(C) = \operatorname{supp}(\pi)$ *for any* $\pi \in C$.

In our application, we take n to be even and consider involutions m in the conjugacy class of elements consisting of $n/2$ disjoint transpositions, $\mathfrak{m} = \mathfrak{m}_n = \{\sigma\left((12)(34)\cdots(n-1\;\; n)\right)\sigma^{-1} \mid \sigma \in S_n\}$. Note that each $m \in \mathfrak{m}_n$ is associated with one of the $(n-1)!! = (n-1)(n-3)(n-5)\cdots 1$ perfect matchings of n things, and that $\operatorname{supp}(m) = n$.

6. Strong Fourier sampling over S_n.

We consider the hidden subgroup $H = \{1, m\}$, where m is chosen uniformly from $\mathfrak{m} = \mathfrak{m}_n \subset S_n$, the conjugacy class

$$\{\pi^{-1}((1\;2)(3\;4)\cdots(n-1\;n))\pi \mid \pi \in S_n\} \;\; ;$$

we assume throughout that n is even. We start by performing weak sampling, i.e., measuring the name of an irreducible representation λ; the resulting probability distribution on $\widehat{S_n}$ is the same for all $m \in M_n$, and Hallgren, Russell, and Ta-Shma [15] established that this probability distribution on λ is exponentially close to the Plancherel distribution in total variation. We continue on to strong sampling, by allowing the algorithm designer to choose an arbitrary POVM with a frame $B = \{\mathbf{b}_j\}$ and weights $\{a_j\}$ obeying the completeness condition (3.3). We will show that with high probability (over m and λ), the conditional distribution induced on the vectors B is exponentially close to the natural distribution (3.9) on B. It will follow by the triangle inequality that it requires an exponential number of experiments of this type to distinguish two involutions from each other or, in fact, to distinguish H from the trivial subgroup.

For simplicity, and to illustrate our techniques, we first prove this for a von Neumann measurement, i.e., where B is an orthonormal basis for λ. In this case, we show that the probability distribution on B is exponentially close to the uniform distribution.

6.1. Von Neumann measurements.

Theorem 6.1. *Let B be an orthonormal basis for an irreducible representation λ. Given the hidden subgroup $H = \{1, m\}$, where m is chosen uniformly at random from $\mathfrak{m}$, let $P_m(\mathbf{b}) = P_m(\lambda, \mathbf{b})$ be the probability that we observe the vector $\mathbf{b} \in B$ conditioned on having observed the representation name λ, and let $U(\mathbf{b}) = U(\lambda, \mathbf{b})$ be the uniform distribution on B. Then there is a constant $\beta > 0$ such that for sufficiently large n, with probability at least $1 - e^{-\beta n}$ in m and λ, we have*

$$\|P_m - U\|_1 < e^{-\beta n} \;\; .$$

Proof. First, recall from (3.8) in section 3 that the conditional distribution on B is given by (since $a_j = 1$)

$$P_m(\mathbf{b}) = P_m(\lambda, \mathbf{b}) = \frac{\|\Pi_m \mathbf{b}\|^2}{\mathbf{rk}\,\Pi_m} \;\; . \tag{6.1}$$

Our strategy will be to bound $\operatorname{Var}_m \|\Pi_m \mathbf{b}\|^2$ using Lemma 4.3 and apply Chebyshev's inequality to conclude that $\|\Pi_m \mathbf{b}\|^2$ is almost certainly close to its expectation (4.1). Recall, however, that our bounds on the variance of $\|\Pi_m \mathbf{b}\|^2$ depend on the decomposition of $\lambda \otimes \lambda^*$ into irreducibles and, furthermore, on the projection of $\mathbf{b} \otimes \mathbf{b}^*$ into these irreducible subspaces. Matters are somewhat complicated by the fact that certain irreducibles μ appearing in $\lambda \otimes \lambda^*$ may contribute more to the variance than others. Specifically, while Theorem 5.5 allows us to bound the contribution of those μ

with Young diagrams whose width μ_1 and height μ_1' are much smaller than n, those which violate this condition could have large normalized characters $\chi_\mu(\mathfrak{m})/d_\mu$, and thus could conceivably contribute large terms to the sum (4.2).

Fortunately, we will see that the total fraction of the space $\lambda \otimes \lambda^*$, dimensionwise, consisting of such μ is small with overwhelming probability. Despite this, we cannot preclude the possibility that for a *specific* vector $\mathbf{b}$, the quantity $\text{Var}\,\|\Pi_m \mathbf{b}\|^2$ is large, as $\mathbf{b}$ may project solely into spaces of the type described above. On the other hand, as these troublesome spaces amount to a small fraction of $\lambda \otimes \lambda^*$, only a few $\mathbf{b}$ can have this property, and this will suffice to control the distance in total variation from the uniform distribution.

Specifically, let $0 < c < 1/4$ be a constant, and let Λ_c denote the collection of Young diagrams μ with the property that either $\mu_1 \geq (1-c)n$ or $\mu_1' \geq (1-c)n$. We have the following upper bounds on the cardinality of Λ_c and the dimension of any μ with $\mu \in \Lambda_c$.

LEMMA 6.2. *Let $p(n)$ denote the number of integer partitions of n. Then $|\Lambda_c| \leq 2cn \cdot p(cn)$, and $d_\mu < n^{cn}$ for any $\mu \in \Lambda_c$.*

Proof. For the first statement, note that removing the top row of a Young diagram μ with $\mu_1 \geq (1-c)n$ gives a Young diagram of size $n - \mu_1 \leq cn$. The number of these is at most $p(cn)$, and summing over all such μ_1 gives $cn \cdot p(cn)$. The case $\mu_1' \geq (1-c)n$ is similar, and summing the two gives $|\Lambda_c| \leq 2cn \cdot p(cn)$.

Now let $\mu \in \Lambda_c$ with $\mu_1 \geq (1-c)n$. By the hook-length formula, since the ith cell from the right in the top row has $\text{hook}(c) \geq i$, $d_\mu < n!/\mu_1! \leq n!/((1-c)n)! \leq n^{cn}$. The case $\mu_1' \geq (1-c)n$ is similar. □

To introduce a bit more notation, given a constant d, let M_d denote the set of irreducibles λ such that $d_\lambda \leq n^{dn}$. Now Lemma 5.3, part 2 shows that if λ is drawn according to the Plancherel distribution, the probability that it falls into M_d for some $d < 1/2$ is $n^{-\Omega(n)}$. The following lemma shows that this is also true for the distribution $P(\rho)$ induced on $\widehat{S_n}$ by weak Fourier sampling the coset state $|H\rangle$.

LEMMA 6.3. *Let $d < 1/2$ be a constant and let λ be drawn according to the distribution $P(\cdot)$ of (3.7). Then there is a constant $\gamma = \gamma(d) > 0$ such that for sufficiently large n we have $\Pr_\lambda[d_\lambda \in \mathrm{M}_d] \leq n^{-\gamma n}$.*

Proof. As $|H| = 2$, $|G| = n!$, and $\mathbf{rk}\,\Pi_H \leq d_\rho$, we have

$$P(\rho) = \frac{d_\rho |H|}{|G|}\,\mathbf{rk}\,\Pi_H \leq \frac{2d_\rho^2}{n!} \quad .$$

Thus $P(\cdot)$ is at most twice the Plancherel measure, and applying Lemma 5.3, part 2 completes the proof. □

Now, for a representation μ with $\mu \notin \Lambda_c$, Theorem 5.5 implies that

$$\left|\frac{\chi_\mu(\mathfrak{m})}{d_\mu}\right| \leq \bigl(\max(q, 1-c)\bigr)^{bn} \leq e^{-\alpha n} \tag{6.2}$$

for a constant $\alpha > 0$. Thus the contribution of such an irreducible to the variance estimate of Lemma 4.3 is exponentially small. In addition, note that Lemma 6.2 implies that $\Lambda_c \subset \mathrm{M}_d$ so long as $d > c$; we shall in fact assume that $c < 1/4 < d$ (and, moreover, that $2c < d$) so that conditioning on $\lambda \notin \mathrm{M}_d$, (4.4) and (6.2) imply that

$$\frac{d_\lambda}{2}\left(1 - e^{-\alpha n}\right) \leq \mathbf{rk}\,\Pi_m \leq \frac{d_\lambda}{2}\left(1 + e^{-\alpha n}\right) \quad . \tag{6.3}$$

We turn now to the problem of bounding the multiplicities with which representations $\mu \in \Lambda_c$ can appear in $\lambda \otimes \lambda^*$. While no explicit decomposition is known for $\lambda \otimes \lambda^*$, the *endomorphism representations* of S_n, we record a coarse bound below which will suffice for our purposes. Recall that a character of $\lambda \otimes \lambda^*$ is χ_λ^2 as the characters of S_n are real. The multiplicity of the representation μ in $\lambda \otimes \lambda^*$ is $\langle \chi_\mu, \chi_\lambda^2 \rangle_G$. However, this is equal to $\langle \chi_\mu \chi_\lambda, \chi_\lambda \rangle_G$, the multiplicity of λ in $\mu \otimes \lambda$. Counting dimensions, this is clearly no more than $\dim(\mu \otimes \lambda)/\dim \lambda = d_\mu$. Hence the multiplicity of μ in $\lambda \otimes \lambda^*$ is bounded by

$$\langle \chi_\mu, \chi_\lambda^2 \rangle_G \leq d_\mu \ . \tag{6.4}$$

Let $L \subset \lambda \otimes \lambda^*$ be the subspace consisting of copies of representations μ with $\mu \in \Lambda_c$, and let Π_L be the projection operator onto this subspace. By Lemma 6.2, we have

$$\dim L \leq \sum_{\mu \in \Lambda_c} d_\mu^2 \leq 2cn \cdot p(cn) \cdot n^{2cn} = n^{2cn} e^{O(\sqrt{n})} \ .$$

Moreover, as B is an orthonormal basis for λ, the vectors $\{\mathbf{b} \otimes \mathbf{b}^* \mid \mathbf{b} \in B\}$ are mutually orthogonal in $\lambda \otimes \lambda^*$. Therefore,

$$\sum_{\mathbf{b} \in B} \|\Pi_L(\mathbf{b} \otimes \mathbf{b}^*)\|^2 \leq \dim L \ . \tag{6.5}$$

Applying the general bound provided by Lemma 4.3 on the variance of $\|\Pi_m \mathbf{b}\|^2$ with the estimates (6.5) and (6.2) above, assuming pessimistically that $\chi_\mu(M)/d_\mu = 1$ for all $\mu \in \Lambda_c$, and assuming that $\lambda \notin M_d$ so that $|B| = d_\lambda > n^{dn}$ yields

(6.6)

$$\begin{aligned}\frac{1}{d_\lambda} \sum_{\mathbf{b}} \mathrm{Var}_m \|\Pi_m \mathbf{b}\|^2 &\leq \frac{1}{4d_\lambda} \Bigg[\sum_{\mathbf{b}} \sum_{\mu \in \Lambda_c} \left\|\Pi_\mu^\lambda(\mathbf{b} \otimes \mathbf{b}^*)\right\|^2 \\ &\qquad + \sum_{\mathbf{b}} \sum_{\mu \notin \Lambda_c} \frac{\chi_\mu(M)}{d_\mu} \left\|\Pi_\mu^\lambda(\mathbf{b} \otimes \mathbf{b}^*)\right\|^2 \Bigg] \\ &\leq \frac{1}{4d_\lambda} \left[n^{2cn} e^{O(\sqrt{n})} + e^{-\alpha n} d_\lambda \right] \leq \frac{1}{4} \left(n^{(2c-d)n} e^{O(\sqrt{n})} + e^{-\alpha n} \right) \\ &\leq \frac{e^{-\alpha n}}{2} \ ,\end{aligned}$$

for sufficiently large n.

We return to our goal of bounding $\|P(\lambda, \cdot) - U(\lambda, \cdot)\|_1$ for a typical λ. (We note that the following part of the proof is considerably simplified from the conference version of this paper and is similar to the argument in the multiregister case appearing in [16].) First, note that for $1/2 > d > 1/4 > c$ and sufficiently large n,

$$\begin{aligned}\mathrm{Exp}_\lambda \mathrm{Exp}_m \|P_m(\lambda, \cdot) - U(\lambda, \cdot)\|_1^2 &\leq 4 \Pr[\lambda \in \mathrm{M}_d] + \max_{\lambda \notin \mathrm{M}_d} \mathrm{Exp}_m \|P_m(\lambda, \cdot) - U(\lambda, \cdot)\|_1^2 \\ &= 4n^{-\gamma n} + \max_{\lambda \notin \mathrm{M}_d} \mathrm{Exp}_m \left(\sum_{\mathbf{b}} \left| \frac{\|\Pi_m \mathbf{b}\|^2}{\mathbf{rk}\, \Pi_m} - \frac{1}{d_\lambda} \right| \right)^2\end{aligned}$$

$$= 4n^{-\gamma n} + \max_{\lambda \notin \mathrm{M}_d} \frac{1}{(\mathbf{rk}\, \Pi_m)^2} \mathrm{Exp}_m \left(\sum_{\mathbf{b}} \left| \|\Pi_m \mathbf{b}\|^2 - \frac{\mathbf{rk}\, \Pi_m}{d_\lambda} \right| \right)^2 \tag{6.7}$$

$$\leq 4n^{-\gamma n} + \max_{\lambda \notin \mathrm{M}_d} \frac{d_\lambda}{(\mathbf{rk}\,\Pi_m)^2} \operatorname{Exp}_m \sum_{\mathbf{b}} \left(\|\Pi_m \mathbf{b}\|^2 - \frac{\mathbf{rk}\,\Pi_m}{d_\lambda} \right)^2 \tag{6.8}$$

$$\leq 4n^{-\gamma n} + \max_{\lambda \notin \mathrm{M}_d} \frac{4}{(1-e^{-\alpha n})^2} \left[\frac{1}{d_\lambda} \sum_{\mathbf{b}} \operatorname{Exp}_m \left(\|\Pi_m \mathbf{b}\|^2 - \frac{\mathbf{rk}\,\Pi_m}{d_\lambda} \right)^2 \right] , \tag{6.9}$$

where (6.8) follows from (6.7) by the Cauchy–Schwarz inequality and (6.9) follows from (6.8) by applying (6.3). Now observe that the bracketed expression is exactly that bounded by (6.6) above. Thus we have

$$\operatorname{Exp}_{\lambda,m} \|P_m(\lambda,\cdot) - U(\lambda,\cdot)\|_1^2 \leq 4n^{-\gamma n} + \frac{2e^{-\alpha n}}{(1-e^{-\alpha n})^2} \leq 3e^{-\alpha n}$$

for sufficiently large n. Finally, the assertion of the theorem follows by applying Markov's inequality and setting $\beta < \alpha/3$. □

6.2. Arbitrary POVMs. We now generalize the proof of Theorem 6.1 to the case where the algorithm designer is allowed to choose an arbitrary finite frame $B = \{\mathbf{b}\}$ of unit length vectors in λ, with a family of positive real weights $a_{\mathbf{b}}$ that satisfy the completeness condition

$$\sum_{\mathbf{b}} a_{\mathbf{b}} \left|\mathbf{b}\right\rangle \left\langle\mathbf{b}\right| = \mathbb{1} \ . \tag{6.10}$$

(Note that this is simply (3.3) where we have written $\mathbf{b}$ and $a_{\mathbf{b}}$ instead of $\mathbf{b}_j$ and a_j.)

THEOREM 6.4. *Let B be a frame with weights $\{a_{\mathbf{b}} \mid \mathbf{b} \in B\}$ satisfying the completeness condition* (6.10) *for an irreducible representation λ. Given the hidden subgroup $H = \{1, m\}$, where m is chosen uniformly at random from $\mathfrak{m}$, let $P_m(\mathbf{b}) = P_m(\lambda, \mathbf{b})$ be the probability that we observe the vector $\mathbf{b}$ conditioned on having observed the representation name λ, and let $N(\mathbf{b}) = N(\lambda, \mathbf{b})$ be the natural distribution* (3.9) *on B. Then there is a constant $\beta > 0$ such that for sufficiently large n, with probability at least $1 - e^{-\beta n}$ in m and λ, we have*

$$\|P_m - N\|_1 < e^{-\beta n} \ .$$

Proof. The proof of Theorem 6.1 goes through with a few modifications. Recall from (3.8) in section 3 that the conditional distribution on B is given by

$$P_m(\mathbf{b}) = P_m(\lambda, \mathbf{b}) = a_{\mathbf{b}} \frac{\|\Pi_m \mathbf{b}\|^2}{\mathbf{rk}\,\Pi_m} \ ,$$

and the natural distribution (3.9) is given by $N(\mathbf{b}) = a_{\mathbf{b}}/d_\lambda$.

First, let us change some semantics: given a subset $A \subseteq B$, we let $|A|$ denote the weighted size of A,

$$|A| = \sum_{\mathbf{b} \in A} a_{\mathbf{b}} \ .$$

With this definition, the total probability that falls in A under the natural distribution is $N(A) = |A|/d_\lambda$. With Λ_c and M_d defined as before, Lemmas 6.2 and 6.3 still apply. As in the development leading to (6.9), we find that $\operatorname{Exp}_\lambda \operatorname{Exp}_m \|P_m(\lambda,\cdot) - N(\lambda,\cdot)\|_1^2$ is no more than

$$4n^{-\gamma n} + \max_{\lambda \notin \mathrm{M}_d} \frac{1}{(\mathbf{rk}\,\Pi_m)^2} \operatorname{Exp}_m \left(\sum_{\mathbf{b}} a_{\mathbf{b}} \left| \|\Pi_m \mathbf{b}\|^2 - \frac{\mathbf{rk}\,\Pi_m}{d_\lambda} \right| \right)^2 \tag{6.11}$$

$$\leq 4n^{-\gamma n} + \max_{\lambda \notin \mathrm{M}_d} \frac{d_\lambda}{(\mathbf{rk}\,\Pi_m)^2} \operatorname{Exp}_m \sum_{\mathbf{b}} a_{\mathbf{b}} \left(\|\Pi_m \mathbf{b}\|^2 - \frac{\mathbf{rk}\,\Pi_m}{d_\lambda} \right)^2 \tag{6.12}$$

$$\leq 4n^{-\gamma n} + \max_{\lambda \notin \mathrm{M}_d} \frac{4}{(1-e^{-\alpha n})^2} \left[\frac{1}{d_\lambda} \sum_{\mathbf{b}} a_{\mathbf{b}} \operatorname{Exp}_m \left(\|\Pi_m \mathbf{b}\|^2 - \frac{\mathbf{rk}\,\Pi_m}{d_\lambda} \right)^2 \right] \ , \tag{6.13}$$

where (6.12) follows from (6.11) by applying the Cauchy–Schwarz inequality in the following way: for any function $f(\mathbf{b})$ we have

$$\left(\sum_{\mathbf{b}} a_{\mathbf{b}} |f(\mathbf{b})| \right)^2 \leq \left(\sum_{\mathbf{b}} a_{\mathbf{b}} \right) \left(\sum_{\mathbf{b}} a_{\mathbf{b}} \, |f(\mathbf{b})|^2 \right) = d_\lambda \sum_{\mathbf{b}} a_{\mathbf{b}} \, |f(\mathbf{b})|^2 \ .$$

As before, let $L \subset \lambda \otimes \lambda^*$ be the subspace consisting of copies of representations $\mu \in \Lambda_c$. In order to control the variance appearing in the bracketed expression of (6.13), we require an analogue of (6.5) for frames, proved below.

LEMMA 6.5. *Let L be a subspace of $\lambda \otimes \lambda^*$, and let Π_L be the projection operator onto L. Then*

$$\sum_{\mathbf{b}} a_{\mathbf{b}} \, \|\Pi_L(\mathbf{b} \otimes \mathbf{b}^*)\|^2 \leq \dim L \ . \tag{6.14}$$

Proof. First note that a vector $\mathbf{e} \in \lambda \otimes \lambda^*$ has entries $\mathbf{e}_{j,k}$ for $1 \leq j, k \leq d_\lambda$. There is a unique linear operator E on λ whose matrix entries are $E_{j,k} = \mathbf{e}_{j,k}$, and the inner product $\langle \mathbf{b} \otimes \mathbf{b}^*, \mathbf{e} \rangle$ in $\lambda \otimes \lambda^*$ can then be written as the bilinear form $\langle \mathbf{b}, E\mathbf{b} \rangle$ in λ. The Frobenius norm of E is $\|E\|^2 = \mathbf{tr}\, E^\dagger E = \|\mathbf{e}\|^2$.

Now let $\{\mathbf{e}_i\}$ be an orthonormal basis for L and let E_i be the operator corresponding to $\mathbf{e}_i$. Then

$$\begin{aligned} \sum_{\mathbf{b}} a_{\mathbf{b}} \, |\langle \mathbf{b} \otimes \mathbf{b}^*, \mathbf{e}_i \rangle|^2 &= \sum_{\mathbf{b}} a_{\mathbf{b}} \, |\langle \mathbf{b}, E_i \mathbf{b} \rangle|^2 \leq \sum_{\mathbf{b}} a_{\mathbf{b}} \, \|\mathbf{b}\|^2 \, \|E_i \mathbf{b}\|^2 = \sum_{\mathbf{b}} a_{\mathbf{b}} \, \|E_i \mathbf{b}\|^2 \\ &= \sum_{\mathbf{b}} a_{\mathbf{b}} \, \mathbf{tr} \left(E_i^\dagger \, |\mathbf{b}\rangle \, \langle \mathbf{b}| \, E_i \right) = \mathbf{tr} \left[E_i^\dagger \left(\sum_{\mathbf{b}} a_{\mathbf{b}} \, |\mathbf{b}\rangle \, \langle \mathbf{b}| \right) E_i \right] \\ &= \mathbf{tr}\, E_i^\dagger E_i = \|\mathbf{e}_i\|^2 = 1 \ , \end{aligned}$$

where we used the Cauchy–Schwarz inequality in the first line and completeness in the second line. Summing over the $\dim L$ basis vectors $\mathbf{e}_i$ then gives (6.14). □

Applying this lemma, we control $1/d_\lambda \cdot \sum_{\mathbf{b}} a_{\mathbf{b}} \operatorname{Var}_m \|\pi_m \mathbf{b}\|^2$ just as in (6.6). Substituting this bound for the bracketed expression of (6.13) and selecting $\beta < \alpha/3$ (as above) completes the proof. □

7. Structured involutions and the case of graph isomorphism. The preceding development focuses on the case where the hidden subgroup is distributed uniformly among the conjugates of the subgroup $H = \{1, m\}$. As such, this shows that the canonical reduction of GRAPH AUTOMORPHISM (the problem of determining whether a given graph has a nontrivial automorphism) to the HSP does not give rise to an efficient quantum algorithm via Fourier sampling.

However, the canonical reduction of Graph Isomorphism to the HSP induces a more structured set of involutions. As referred to in the introduction, fixing two rigid graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, each with n vertices, the automorphism group of their disjoint union $(V_1 \cup V_2,\ E_1 \cup E_2)$ is nontrivial exactly when they are isomorphic, in which case it is generated by an involution m with full support such that $m(V_1) = V_2$ and $m(V_2) = V_1$. Identifying V_1 and V_2 with the sets $\{1, \dots, n\}$ and $\{n+1, \dots, 2n\}$, respectively, and letting s denote the involution $(1\ n+1)(2\ n+2)\dots(n\ 2n)$, the standard reduction to the HSP in S_{2n} then results in a hidden subgroup $H = \{1, m\}$, where m is a conjugate involution $a^{-1}sa$. However, rather than a being drawn from all of S_{2n}, it is an element of the Young subgroup $S_{n,n}$ which fixes V_1 and V_2:

$$S_{n,n} = \big\{\pi \in S_{2n} \mid \pi(\{1, \dots, n\}) = \{1, \dots, n\}\big\} \cong S_n \times S_n \ \ .$$

In other words, rather than considering all conjugates of m in S_{2n}, it suffices just to consider conjugates in $S_{n,n}$. A priori, it seems that this smaller set of possible hidden subgroups might be easier to identify. Moreover, let K be the subgroup generated by $S_{n,n}$ and s: this is the wreath product $S_n \wr \mathbb{Z}_2$, which can also be written as a semidirect product $K = (S_n \times S_n) \rtimes \mathbb{Z}_2$. Then each such H is contained in K, and it seems that it might be more intelligent to Fourier sample over K rather than over all of S_{2n}.

However, we can show that nothing is gained by this approach. First, note that the involutions described above form the (K-)conjugacy class

$$\big\{\big((\alpha, \alpha^{-1}), 1\big) \in (S_n \times S_n) \rtimes \mathbb{Z}_2 \mid \alpha \in S_n\big\} \ \ .$$

We remark that the development of section 3 is unchanged and that the optimal measurement to find a hidden conjugate again consists of strong Fourier sampling.

Now note that Fourier sampling over S_{2n} and over K is equivalent for the following reason: suppose we are trying to distinguish a set of hidden subgroups $H_i \subset G$, all of which are contained in a subgroup $K \subset G$. Let T be a set of representatives for the cosets of K. Then a random left coset of H_i in G is the product of a random left coset of H_i in K with a random element of T. Thus the mixed state describing a uniformly random coset of H_i in G can be written as the tensor product of the corresponding coset state over K with the completely mixed state over T. Since this completely mixed state (whose density matrix is the identity) contains no information, nothing is gained, or lost, by sampling over all of G rather than over K.

To proceed, we can determine K's irreducible representations and their characters, using the machinery of *induced representations* [33] as follows. For two irreducible representations ρ and σ of S_n, let $\rho \boxtimes \sigma$ denote their tensor product as a representation of $S_{n,n} \cong S_n \times S_n$. We consider the induced representation $\tau_{\{\rho,\sigma\}} = \mathrm{Ind}_{S_{n,n}}^{K}(\rho \boxtimes \sigma)$ and denote its character $\chi_{\{\rho,\sigma\}}$. It is easy to see that

$$\chi_{\{\rho,\sigma\}}\big(((\alpha, \beta), t)\big) = \begin{cases} \chi_\rho(\alpha)\chi_\sigma(\beta) + \chi_\sigma(\alpha)\chi_\rho(\beta) & \text{if } t = 0 \ \ , \\ 0 & \text{if } t = 1 \ \ ; \end{cases}$$

as the notation suggests, this depends only on the multiset $\{\rho, \sigma\}$. An easy computation shows that $\langle \chi_{\{\rho,\sigma\}}, \chi_{\{\rho,\sigma\}} \rangle = 1 + \delta_{\rho,\sigma}$. Thus, if $\rho \ncong \sigma$, then $\tau_{\{\rho,\sigma\}}$ is irreducible of dimension $2d_\rho d_\sigma$. On the other hand, if $\rho \cong \sigma$, then it decomposes into two irreducible representations of dimension d_ρ^2,

$$\tau_{\{\rho,\rho\}} \cong \tau_{\{\rho,\rho\},\mathbb{1}} \oplus \tau_{\{\rho,\rho\},\pi} \ \ , \tag{7.1}$$

where $\mathbb{1}$ and π are the trivial and sign representations, respectively, of $\mathbb{Z}_2$. Each of these irreducible representations acts on $V_\rho \otimes V_\rho$, the vector space supporting the action of $\rho \boxtimes \rho$. Both realize the element $((\alpha, \beta), 0)$ as the linear map $\rho(\alpha) \otimes \rho(\beta)$, while $\tau_{\{\rho,\rho\},\mathbb{1}}$ and $\tau_{\{\rho,\rho\},\pi}$ realize the element $((1, 1), 1)$ as the maps which send $\mathbf{u} \otimes \mathbf{v}$ to $\mathbf{v} \otimes \mathbf{u}$ and $-\mathbf{v} \otimes \mathbf{u}$, respectively. The characters of these representations are

$$\begin{aligned} \chi_{\{\rho,\rho\},\mathbb{1}}\big(((\alpha,\beta),t)\big) &= \begin{cases} \chi_\rho(\alpha)\chi_\rho(\beta) & \text{if } t = 0 \ , \\ \chi_\rho(\alpha\beta) & \text{if } t = 1 \ , \end{cases} \\ \chi_{\{\rho,\rho\},\pi}\big(((\alpha,\beta),t)\big) &= \begin{cases} \chi_\rho(\alpha)\chi_\rho(\beta) & \text{if } t = 0 \ , \\ -\chi_\rho(\alpha\beta) & \text{if } t = 1 \ . \end{cases} \end{aligned} \tag{7.2}$$

In particular, since m is of the form $((\alpha, \alpha^{-1}), 1)$, we have the normalized characters

$$\frac{\chi_{\{\rho,\rho\},\mathbb{1}}(m)}{d_{\{\rho,\rho\},\mathbb{1}}} = \frac{1}{d_\rho}, \quad \frac{\chi_{\{\rho,\rho\},\pi}(m)}{d_{\{\rho,\rho\},\pi}} = -\frac{1}{d_\rho} \ , \tag{7.3}$$

and $\chi_{\{\rho,\sigma\}}(m) = 0$ for all $\rho \ncong \sigma$.

Given that the normalized characters (7.3) are very small (indeed, $n^{-\Omega(n)}$) for all ρ whose Young diagram is outside Λ_c, the analysis of section 6 can be undertaken mutatis mutandis and easily implies that an exponential number of strong Fourier sampling experiments would have to be performed to distinguish the isomorphic and nonisomorphic cases. We note that a similar result has been obtained by Childs and Wojcan [6], who treat Graph Isomorphism as a hidden shift problem on S_n.

We remark that the above description (7.1), (7.2) of the irreducible representations and characters of groups of the form $G \wr \mathbb{Z}_2$ works for arbitrary G. In particular, the normalized characters of the involutions that "swap" the two copies of G are either 0 or $\pm 1/d_\rho$ for some $\rho \in \widehat{G}$. It follows that strong Fourier sampling fails to find such involutions in $G \wr \mathbb{Z}_2$ whenever a sufficient fraction of G's Plancherel measure lies on sufficiently high-dimensional representations.

Acknowledgments. We are grateful to Denis Thérien, McGill University, and Bellairs Research Institute for organizing a workshop at which this work began; to Dorit Aharonov, Daniel Rockmore, and Umesh Vazirani for helpful conversations; to Chris Lomont for pointing out several typos; and to Tracy Conrad and Sally Milius for their support and tolerance. C. M. also thanks Rosemary Moore for providing a larger perspective. Finally, we thank Gorjan Alagic for his comments on the structured involutions material.

REFERENCES

[1] G. Alagic, C. Moore, and A. Russell, *Quantum algorithms for Simon's problem over general groups*, in Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2007, pp. 1217–1224.

[2] D. Bacon, A. Childs, and W. van Dam, *Optimal measurements for the dihedral hidden subgroup problem*, Chic. J. Theoret. Comput. Sci., (2006), article 2.

[3] D. Bacon, A. Childs, and W. van Dam, *From optimal measurement to efficient quantum algorithms for the hidden subgroup problem over semidirect product groups*, in Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, 2005, pp. 469–478.

[4] R. Beals, *Quantum computation of Fourier transforms over symmetric groups*, in Proceedings of the 29th Annual ACM Symposium on Theory of Computing, 1997, pp. 48–53.

[5] E. Bernstein and U. Vazirani, *Quantum complexity theory* (preliminary abstract), in Proceedings of the 25th Annual ACM Symposium on Theory of Computing, 1993, pp. 11–20.

[6] A. Childs and P. Wojcan, *On the quantum hardness of solving isomorphism problems as nonabelian hidden shift problems*, Quantum Inf. Comput., 7 (2007), pp. 504–521.

[7] W. van Dam, S. Hallgren, and L. Ip, *Quantum algorithms for some hidden shift problems*, in Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2003, pp. 489–498.

[8] M. Ettinger and P. Høyer, *On quantum algorithms for noncommutative hidden subgroups*, Adv. in Appl. Math., 25 (2000), pp. 239–251.

[9] M. Ettinger, P. Høyer, and E. Knill, *The quantum query complexity of the hidden subgroup problem is polynomial*, Inform. Process. Lett., 91 (2004), pp. 43–48.

[10] K. Friedl, G. Ivanyos, F. Magniez, M. Santha, and P. Sen, *Hidden translation and orbit coset in quantum computing*, in Proceedings of the 35th Annual ACM Symposium on Theory of Computing, 2003, pp. 1–9.

[11] W. Fulton and J. Harris, *Representation Theory: A First Course*, Grad. Texts in Math. 129, Springer-Verlag, New York, 1991.

[12] M. Grigni, L. J. Schulman, M. Vazirani, and U. Vazirani, *Quantum mechanical algorithms for the nonabelian hidden subgroup problem*, Combinatorica, 24 (2004), pp. 137–154.

[13] L. Hales and S. Hallgren, *Quantum Fourier sampling simplified*, in Proceedings of the 31st Annual ACM Symposium on Theory of Computing, 1999, pp. 330–338.

[14] L. Hales and S. Hallgren, *An improved quantum Fourier transform algorithm and applications*, in Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science, 2000, pp. 515–525.

[15] S. Hallgren, A. Russell, and A. Ta-Shma, *Normal subgroup reconstruction and quantum computation using group representations*, in Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, 2000, pp. 627–635.

[16] S. Hallgren, C. Moore, M. Rötteler, A. Russell, and P. Sen, *Limitations of quantum coset states for graph isomorphism*, in Proceedings of the 38th Annual ACM Symposium on Theory of Computing, 2006, pp. 604–617.

[17] P. Høyer, *Efficient Quantum Transforms*, preprint, 1997; available online from http://arxiv.org/abs/quant-ph/9702028.

[18] Y. Inui and F. Le Gall, *An efficient algorithm for the hidden subgroup problem over a class of semi-direct product groups*, in Proceedings of EQIS, 2004.

[19] L. Ip, *Shor's Algorithm Is Optimal*, preprint, 2004.

[20] G. Ivanyos, F. Magniez, and M. Santha, *Efficient quantum algorithms for some instances of the non-abelian hidden subgroup problem*, Internat. J. Found. Comput. Sci., 14 (2003), pp. 723–740.

[21] R. Jozsa, *Quantum factoring, discrete logarithms and the hidden subgroup problem*, IEEE MultiMedia, 3 (1996), pp. 34-43.

[22] J. Kempe and A. Shalev, *The hidden subgroup problem and permutation group theory*, in Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2005, pp. 1118–1125.

[23] S. V. Kerov, *Asymptotic Representation Theory of the Symmetric Group and Its Applications in Analysis*, Transl. Math. Monogr. 219, AMS, Providence, RI, 2003.

[24] G. Kuperberg, *A subexponential-time quantum algorithm for the dihedral hidden subgroup problem*, SIAM J. Comput., 35 (2005), pp. 170–188.

[25] C. Moore and A. Russell, *Explicit Multiregister Measurements for Hidden Subgroup Problems; or, Fourier Sampling Strikes Back*, preprint, 2005; available online from http://arxiv.org/abs/quant-ph/0504067.

[26] C. Moore and A. Russell, *For distinguishing conjugate hidden subgroups, the pretty good measurement is as good as it gets*, Quantum Inf. Commun., 7 (2007), pp. 752-765.

[27] C. Moore, D. Rockmore, and A. Russell, *Generic quantum Fourier transforms*, in Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2004, pp. 771–780.

[28] C. Moore, D. Rockmore, A. Russell, and L. J. Schulman, *The power of basis selection in Fourier sampling: Hidden subgroup problems in affine groups*, in Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2004, pp. 1106–1115.

[29] O. Regev, *Quantum computation and lattice problems*, in Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002, pp. 520–530.

[30] M. Rötteler and T. Beth, *Polynomial-Time Solution to the Hidden Subgroup Problem for a Class of Non-Abelian Groups*, preprint, 1998; available online from http://arxiv.org/abs/quant-ph/9812070.

[31] Y. Roichman, *Upper bound on the characters of the symmetric groups*, Invent. Math., 125 (1996), pp. 451–485.

[32] S. Roman, *Advanced Linear Algebra*, Grad. Texts in Math. 135, Springer-Verlag, New York, 1992.

[33] J.-P. Serre, *Linear Representations of Finite Groups*, Grad. Texts in Math. 42, Springer-Verlag, New York, 1977.

[34] P. W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM J. Comput., 26 (1997), pp. 1484–1509.

[35] D. R. Simon, *On the power of quantum computation*, SIAM J. Comput., 26 (1997), pp. 1474–1483.

[36] A. M. Vershik and S. V. Kerov, *Asymptotic behavior of the maximum and generic dimensions of irreducible representations of the symmetric group*, Funk. Anal. i Prolizhen, 19 (1985), pp. 25–36 (in Russian); Funct. Anal. Appl., 19 (1985), pp. 21–31 (in English).

SIAM J. COMPUT.
Vol. 37, No. 6, pp. 1865–1890

CRYPTOGRAPHY IN THE BOUNDED-QUANTUM-STORAGE MODEL*

IVAN B. DAMGÅRD†, SERGE FEHR‡, LOUIS SALVAIL§, AND CHRISTIAN SCHAFFNER‡

Abstract. We initiate the study of two-party cryptographic primitives with unconditional security, assuming that the adversary's *quantum* memory is of bounded size. We show that oblivious transfer and bit commitment can be implemented in this model using protocols where honest parties need no quantum memory, whereas an adversarial player needs quantum memory of size at least $n/2$ in order to break the protocol, where n is the number of qubits transmitted. This is in sharp contrast to the classical bounded-memory model, where we can only tolerate adversaries with memory of size quadratic in honest players' memory size. Our protocols are efficient and noninteractive and can be implemented using today's technology. On the technical side, a new entropic uncertainty relation involving min-entropy is established.

Key words. quantum cryptography, quantum-bounded-storage model, quantum uncertainty relation, oblivious transfer, bit commitment

AMS subject classifications. 81P68, 94A60

DOI. 10.1137/060651343

1. Introduction.

It is well known that nontrivial two-party cryptographic primitives cannot be securely implemented if only error-free communication is available and there is no limitation assumed on the computing power and memory of the players. Fundamental examples of such primitives are bit commitment (BC) and oblivious transfer (OT). In BC, a committer C commits himself to a choice of a bit b by exchanging information with a verifier V. We want that V does not learn b (we say the commitment is hiding), yet C can later choose to reveal b in a convincing way; i.e., only the value fixed at commitment time will be accepted by V (we say the commitment is binding). In (Rabin) OT, a sender S sends a bit b to a receiver R by executing some protocol in such a way that R receives b with probability $\frac{1}{2}$ and nothing with probability $\frac{1}{2}$, yet S does not learn what was received.

Informally, BC is not possible with unconditional security since hiding means that when 0 is committed, exactly the same information exchange could have happened when committing to a 1. Hence, even if 0 was actually committed to, C could always compute a complete view of the protocol consistent with having committed to 1, and pretend that this was what he had in mind originally. A similar type of argument shows that OT is also impossible in this setting.

*Received by the editors February 6, 2006; accepted for publication (in revised form) November 12, 2007; published electronically March 26, 2008. A preliminary version of this paper appeared in *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2005, pp. 449–458.

http://www.siam.org/journals/sicomp/37-6/65134.html

†Basic Research in Computer Science (BRICS) and Foundations in Cryptography and Security (FICS), Department of Computer Science, University of Århus, N-8200 Århus, The Netherlands (ivan@brics.dk).

‡Center for Mathematics and Computer Science (CWI), 1090 GB Amsterdam, The Netherlands (fehr@cwi.nl, c.schaffner@cwi.nl). The second author's research was supported by a Veni grant from the Dutch Organization for Scientific Research (NWO). The fourth author's research was supported by the European projects SECOQC and QAP and an NWO Vici grant.

§Basic Research in Computer Science (BRICS), Department of Computer Science, University of Århus, N-8200 Århus, The Netherlands (salvail@brics.dk). This author's research was supported in part by the European project PROSECCO.

One might hope that allowing the protocol to make use of quantum communication would make a difference. Here, information is stored in qubits, i.e., in the state of two-level quantum mechanical systems, such as the polarization state of a single photon. It is well known that quantum information behaves in a way that is fundamentally different from classical information, enabling, for instance, unconditionally secure key exchange between two honest players. However, in the case of two mutually distrusting parties, we are not so fortunate: even with quantum communication, unconditionally secure BC and OT remain impossible [31, 34].

There are, however, several scenarios where these impossibility results do not apply, namely:
(i) if the computing power of players is bounded,
(ii) if the communication is noisy,
(iii) if the adversary is under some physical limitation, e.g., the size of the available memory is bounded.

The first scenario is the basis of many well-known solutions based on plausible but unproven complexity assumptions, such as hardness of factoring or discrete logarithms. The second scenario has been used to construct both BC and OT protocols in various models for the noise [13, 15, 14]. The third scenario is our focus here. In this model, OT and BC can be done using classical communication, assuming, however, quite restrictive bounds on the adversary's memory size [10, 19]; namely, it can be at most quadratic in the memory size of honest players. Such an assumption is on the edge of being realistic; it would clearly be more satisfactory to have a larger separation between the memory size of honest players and that of the adversary. However, this was shown to be impossible [22].

In this paper, we study for the first time what happens if instead we consider protocols where quantum communication is used and we place a bound on the adversary's *quantum* memory size. There are two reasons why this may be a good idea: first, if we do not bound the classical memory size, we avoid the impossibility result of [22]. Second, the adversary's typical goal is to obtain a certain piece of classical information that we want to keep hidden from him. However, if he cannot store all the quantum information that is sent, he must convert some of it to classical information by measuring. This may irreversibly destroy information, and we may be able to arrange it such that the adversary cannot afford to lose information this way, while honest players can.

It turns out that this is indeed possible: we present protocols for both BC and OT in which n qubits are transmitted, where honest players need *no quantum memory*, but where the adversary must store at least $n/2$ qubits to break the protocol. We emphasize that no bound is assumed on the adversary's computing power, nor on his classical memory. This is clearly much more satisfactory than the classical case, not only from a theoretical point of view, but also in practice: while sending qubits and measuring them immediately as they arrive is well within reach of current technology, storing even a single qubit for more than a fraction of a second is a formidable technological challenge. Furthermore, we show that our protocols also work in a nonideal setting where we allow the quantum source to be imperfect and the quantum communication to be noisy.

We emphasize that what makes OT and BC possible in our model is not so much the memory bound per se, but rather the loss of information on the part of the adversary. Indeed, our results also hold if the adversary's memory device holds an arbitrary number of qubits but is imperfect in certain ways. This is discussed in more detail in section 6.2.

Our protocols are noninteractive; only one party sends information when doing OT, commitment, or opening. Furthermore, the commitment protocol has the interesting property that the only message is sent *to* the committer; i.e., it is possible to commit while only *receiving* information. Such a scheme clearly does not exist without a bound on the committer's memory, even under computational assumptions and using quantum communication: a corrupt committer could always store (possibly quantumly) all the information sent, until opening time, and only then follow the honest committer's algorithm to figure out what should be sent to convincingly open a 0 or a 1. Note that in the classical bounded-storage model, it is known how to do time-stamping that is noninteractive in our sense: a player can time-stamp a document while only receiving information [35]. However, no reasonable BC or protocol that time-stamps a bit exists in this model. It is straightforward to see that any such protocol can be broken by an adversary with classical memory of size twice that of an honest player, while our protocol requires no memory for the honest players and remains secure against any adversary unable to store more than half the size of the quantum transmission.

We also note that it has been shown earlier that BC is possible using quantum communication, assuming a different type of physical limitation, namely, a bound on the size of coherent measurement that can be implemented [39]. This limitation is incomparable to ours: it does not limit the total size of the memory; instead it limits the number of bits that can be simultaneously operated on to produce a classical result. Our adversary has a limit on the total memory size, but can measure all of it coherently. The protocol from [39] is interactive and requires a bound on the maximal measurement size that is sublinear in n.

On the technical side, we derive a new type of uncertainty relation involving the min-entropy of a quantum encoding (see Theorem 3.1 and Corollary 3.3), which might be useful in other contexts as well. The new relation is then used in combination with a proof technique by Shor and Preskill [41], where the actions of honest players are purified, and with privacy amplification against quantum adversaries as introduced by Renner and König [37, 36].

2. Preliminaries.

2.1. Notation and terminology. For a set $I = \{i_1, i_2, \ldots, i_\ell\} \subseteq \{1, \ldots, n\}$ and an n-bit string $x \in \{0,1\}^n$, we define $x|_I := x_{i_1} x_{i_2} \cdots x_{i_\ell}$, and we write $\mathrm{B}^{\delta n}(x)$ for the set of all n-bit strings at Hamming distance at most δn from x. Note that the number of elements in $\mathrm{B}^{\delta n}(x)$ is the same for all x; we denote it by $\mathrm{B}^{\delta n} := |\mathrm{B}^{\delta n}(x)|$. It is well known that $\mathrm{B}^{\delta n} \leq 2^{nh(\delta)}$, where $h(p)$ denotes the binary entropy function $h(p) := -\big(p \cdot \log p + (1-p) \cdot \log(1-p)\big)$. All logarithms in this paper are to base two. We denote by $negl(n)$ any function of n smaller than any polynomial provided that n is sufficiently large.

For a discrete probability space (Ω, P), we write $P[\mathcal{E}]$ for the probability of the event $\mathcal{E} \subset \Omega$, and we write P_X for the distribution of the random variable $X : \Omega \to \mathcal{X}$. We use similar notation for conditional probabilities and distributions. As is common practice, we do not refer to the probability space (Ω, P) but leave it implicitly defined by the joint probabilities of all considered events and random variables. For a probability distribution Q over $\mathcal{X}$, we abbreviate the (overall) probability of a set $L \subseteq \mathcal{X}$ with $Q(L) := \sum_{x \in L} Q(x)$.

The pair $\{|0\rangle, |1\rangle\}$ denotes the computational or rectilinear or "+" basis for the two-dimensional complex Hilbert space $\mathbb{C}^2$. The diagonal or "×" basis is defined as $\{|0\rangle_\times, |1\rangle_\times\}$, where $|0\rangle_\times = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|1\rangle_\times = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Measuring a qubit in the $+$-basis (resp., $\times$-basis) means applying the measurement described by projectors $|0\rangle\langle 0|$ and $|1\rangle\langle 1|$ (resp., projectors $|0\rangle_\times\langle 0|_\times$ and $|1\rangle_\times\langle 1|_\times$). When the context requires it, we write $|0\rangle_+$ and $|1\rangle_+$ instead of $|0\rangle$ and $|1\rangle$, respectively, and for any $x \in \{0,1\}^n$ and $r \in \{+,\times\}$, we write $|x\rangle_r = \bigotimes_{i=1}^n |x_i\rangle_r$. If we want to choose the $+$- or $\times$-basis according to the bit $b \in \{0,1\}$, we write $\{+,\times\}_{[b]}$.

The behavior of a quantum state in a register E is fully described by its density matrix ρ_{E}. We often consider cases where a quantum state may depend on some classical random variable X, in that it is described by the density matrix ρ_{E}^x if and only if $X = x$. For an observer who has access only to the register E but not to X, the behavior of the state is determined by the density matrix $\sum_x P_X(x)\rho_{\mathsf{E}}^x$. The joint state, consisting of the classical X and the quantum register E and therefore called a *cq-state*, is described by the density matrix $\sum_x P_X(x)|x\rangle\langle x| \otimes \rho_{\mathsf{E}}^x$. In order to have more compact expressions, we use the following notation. We write

$$\rho_{X\mathsf{E}} = \sum_x P_X(x)|x\rangle\langle x| \otimes \rho_{\mathsf{E}}^x \quad \text{and} \quad \rho_{\mathsf{E}} = \mathrm{tr}_X(\rho_{X\mathsf{E}}) = \sum_x P_X(x)\rho_{\mathsf{E}}^x\,.$$

More generally, for any event $\mathcal{E}$, we write

$$(2.1)\qquad \rho_{X\mathsf{E}|\mathcal{E}} = \sum_x P_{X|\mathcal{E}}(x)|x\rangle\langle x| \otimes \rho_{\mathsf{E}}^x \quad \text{and} \quad \rho_{\mathsf{E}|\mathcal{E}} = \mathrm{tr}_X(\rho_{X\mathsf{E}|\mathcal{E}}) = \sum_x P_{X|\mathcal{E}}(x)\rho_{\mathsf{E}}^x.$$

We also write $\rho_X = \sum_x P_X(x)|x\rangle\langle x|$ for the quantum representation of the classical random variable X (and similarly for $\rho_{X|\mathcal{E}}$). This notation extends naturally to quantum states that depend on several classical random variables (i.e., to ccq-states, etc.).

This notation extends naturally to quantum states that depend on several classical random variables (i.e., to ccq-states, etc.), defining the density matrices $\rho_{XY\mathsf{E}}$, $\rho_{XY\mathsf{E}|\mathcal{E}}$, $\rho_{Y\mathsf{E}|X=x}$, etc. We tend to slightly abuse notation and write $\rho_{Y\mathsf{E}}^x = \rho_{X\mathsf{E}|X=x}$ and $\rho_{Y\mathsf{E}|\mathcal{E}}^x = \rho_{Y\mathsf{E}|X=x,\mathcal{E}}$, as well as $\rho_{\mathsf{E}}^x = \mathrm{tr}_Y(\rho_{Y\mathsf{E}}^x)$ and $\rho_{\mathsf{E}|\mathcal{E}}^x = \mathrm{tr}_Y(\rho_{Y\mathsf{E}|\mathcal{E}}^x)$.[1] Note that writing $\rho_{X\mathsf{E}} = \mathrm{tr}_Y(\rho_{XY\mathsf{E}})$ and $\rho_{\mathsf{E}} = \mathrm{tr}_{X,Y}(\rho_{XY\mathsf{E}})$ is consistent with the above notation. We also write $\rho_{X\mathsf{E}|\mathcal{E}} = \mathrm{tr}_Y(\rho_{XY\mathsf{E}|\mathcal{E}})$ and $\rho_{\mathsf{E}|\mathcal{E}} = \mathrm{tr}_{X,Y}(\rho_{XY\mathsf{E}|\mathcal{E}})$, where one has to be aware that in contrast to (2.1), here the state E may depend on the event $\mathcal{E}$ when given x (namely, via Y), so that, e.g., $\rho_{\mathsf{E}|\mathcal{E}} = \sum_x P_{X|\mathcal{E}}(x)\rho_{\mathsf{E}|\mathcal{E}}^x$.

Given a cq-state $\rho_{X\mathsf{E}}$, by saying that there exists a random variable Y such that $\rho_{XY\mathsf{E}}$ satisfies some condition, we mean that $\rho_{X\mathsf{E}}$ can be understood as $\rho_{X\mathsf{E}} = \mathrm{tr}_Y(\rho_{XY\mathsf{E}})$ for a ccq-state $\rho_{XY\mathsf{E}}$ that satisfies the required condition.

Obviously, $\rho_{X\mathsf{E}} = \rho_X \otimes \rho_{\mathsf{E}}$ if and only if the quantum part is independent of X (in that $\rho_{\mathsf{E}}^x = \rho_{\mathsf{E}}$ for any x), where the latter in particular implies that no information on X can be learned by observing only ρ_{E}. Furthermore, if $\rho_{X\mathsf{E}}$ and $\rho_X \otimes \rho_{\mathsf{E}}$ are ε-close in terms of their trace distance $\delta(\rho,\sigma) = \frac{1}{2}\mathrm{tr}(|\rho - \sigma|)$, then the real system $\rho_{X\mathsf{E}}$ "behaves" as the ideal system $\rho_X \otimes \rho_{\mathsf{E}}$ except with probability ε [37] in that for any evolution of the system no observer can distinguish the real system from the ideal one with advantage greater than ε. Throughout the paper, $\mathbb{1}$ stands for the identity matrix (describing the fully mixed state) renormalized by the appropriate dimension.

We consider the notion of the classical *Rényi entropy* $\mathrm{H}_\alpha(X)$ of order α of a random variable X [38], as well as its generalization to the Rényi entropy $\mathrm{H}_\alpha(\rho)$ of

[1] The density matrix $\rho_{\mathsf{E}|\mathcal{E}}^x$ describes the quantum state E in the case that the event $\mathcal{E}$ occurs and X takes on the value x. The corresponding convention holds for the other density matrices considered here.

a quantum state ρ [37]. It holds that $\mathrm{H}_\alpha(\rho_X) = \mathrm{H}_\alpha(X)$ and $\mathrm{H}_\alpha(\rho_X) \leq \mathrm{H}_\beta(\rho_X)$ if $\alpha \geq \beta$. The cases that are relevant for us are the classical *min-entropy* $\mathrm{H}_\infty(X) = -\log\left(\max_x P_X(x)\right)$ as well as the quantum versions of the *max-* and *collision-entropy* $\mathrm{H}_0(\rho) = \log\left(\mathrm{rank}(\rho)\right)$, respectively, $\mathrm{H}_2(\rho) = -\log\left(\sum_i \lambda_i^2\right)$, where $\{\lambda_i\}_i$ are the eigenvalues of ρ.

2.2. Bounded quantum storage and privacy amplification. All our protocols take place in the *bounded-quantum-storage model*, which concretely means the following: the state of an adversarial player may consist of an arbitrary number of qubits, and he may perform arbitrary quantum computation. At a certain point in time, though, we say that *the memory bound applies*, which means that a measurement is applied to the system with the restriction that the resulting quantum state can be stored in at most q qubits. The classical outcome of the measurement can be of arbitrary size and (classically) stored for later use. After this point, the player is again unbounded in (quantum) memory. Throughout, the adversary may have unbounded computing power and classical memory. We note that our results also apply to some cases where the adversary's memory is not bounded but is noisy in certain ways; see section 6.2.

An important tool we use is universal hashing. A class $\mathcal{F}_n$ of hashing functions from $\{0,1\}^n$ to $\{0,1\}$ is called *two-universal* if for any pair $x, y \in \{0,1\}^n$ with $x \neq y$, and F uniformly chosen from $\mathcal{F}_n$,

$$P\big[F(x) = F(y)\big] \leq \frac{1}{2}.$$

Several two-universal classes of hashing functions are such that evaluating and picking a function uniformly and at random in $\mathcal{F}_n$ can be done efficiently [11, 42].

Theorem 2.1 (see [37]). *Let $\rho_{X\mathsf{E}}$ be a cq-state, where X is distributed over $\{0,1\}^n$ and register E contains q qubits. Let F be the random variable corresponding to the random choice (with uniform distribution and independent from X) of a member of a two-universal class of hashing functions $\mathcal{F}_n$. Then*

$$\delta\big(\rho_{F(X)F\mathsf{E}}, \mathbb{1} \otimes \rho_{F\mathsf{E}}\big) \leq \frac{1}{2}\, 2^{-\frac{1}{2}(\mathrm{H}_2(\rho_{X\mathsf{E}}) - \mathrm{H}_0(\rho_\mathsf{E}) - 1)} \tag{2.2}$$

$$\leq \frac{1}{2}\, 2^{-\frac{1}{2}(\mathrm{H}_\infty(X) - q - 1)}. \tag{2.3}$$

The first inequality (2.2) is the original theorem from [37], and (2.3) follows by observing that $\mathrm{H}_2(\rho_{X\mathsf{E}}) \geq \mathrm{H}_2(\rho_X) = \mathrm{H}_2(X) \geq \mathrm{H}_\infty(X)$. In this paper, we use only this weaker version of the theorem.

Note that if the rightmost term of (2.3) is negligible, i.e., say, smaller than $2^{-\varepsilon n}$, then this situation is $2^{-\varepsilon n}$-close to the ideal situation where $F(X)$ is perfectly uniform and independent of E and F. In particular, replacing $F(X)$ by an independent and uniformly distributed bit results in a common state which essentially cannot be distinguished from the original one.

The following lemma is a direct consequence of Theorem 2.1. In section 5, this lemma will be useful for proving the binding condition of our commitment scheme. Recall that for $X \in \{0,1\}^n$, $\mathrm{B}^{\delta n}(X)$ denotes the set of all n-bit strings at Hamming distance at most δn from X and $\mathrm{B}^{\delta n} := |\mathrm{B}^{\delta n}(X)|$ is the number of such strings.

Lemma 2.2. *Let $\rho_{X\mathsf{E}}$ be a cq-state, where X is distributed over $\{0,1\}^n$ and register E contains q qubits. Let $\hat{X}$ be a guess for X obtained by measuring E. Then,* *for all* $\delta < \frac{1}{2}$ *it holds that*

$$P\big[\hat{X} \in \mathrm{B}^{\delta n}(X)\big] \leq 2^{-\frac{1}{2}(\mathrm{H}_\infty(X)-q-1)+\log(\mathrm{B}^{\delta n})}.$$

In other words, given a quantum memory of q qubits arbitrarily correlated with a classical random variable X, the probability of finding $\hat{X}$ at Hamming distance at most δn from X, where $nh(\delta) < \frac{1}{2}(\mathrm{H}_\infty(X) - q)$, is negligible.

Proof. Here is a strategy to try to bias $F(X)$ when given $\hat{X}$ and $F \in_R \mathcal{F}_n$. Sample $X' \in_R \mathrm{B}^{\delta n}(\hat{X})$ and output $F(X')$. Note that, using p_{succ} as shorthand for the probability $P\big[\hat{X} \in \mathrm{B}^{\delta n}(X)\big]$ to be bounded,

$$\begin{aligned} P\big[F(X') = F(X)\big] &= \frac{p_{\text{succ}}}{\mathrm{B}^{\delta n}} + \left(1 - \frac{p_{\text{succ}}}{\mathrm{B}^{\delta n}}\right)\frac{1}{2} \\ &= \frac{1}{2} + \frac{p_{\text{succ}}}{2 \cdot \mathrm{B}^{\delta n}}, \end{aligned}$$

where the first equality follows from the fact that if $X' \neq X$, then, as $\mathcal{F}_n$ is two-universal, $P\left[F(X) = F(X')\right] = \frac{1}{2}$. Note that, given F and being allowed to measure E, the probability of correctly guessing a binary $F(X)$ is upper bounded by $\frac{1}{2} + \delta(\rho_{F(X)F\mathsf{E}}, \mathbb{1} \otimes \rho_{F\mathsf{E}})$ [24]. In combination with Theorem 2.1, the above results in

$$\frac{1}{2} + \frac{p_{\text{succ}}}{2 \cdot \mathrm{B}^{\delta n}} \leq \frac{1}{2} + \frac{1}{2}2^{-\frac{1}{2}(\mathrm{H}_\infty(X)-q-1)},$$

and the claim follows immediately. □

2.3. Operators and norms. For a linear operator A on the complex Hilbert space $\mathcal{H}$, we define the *operator norm*

$$\|A\| := \sup_{\langle\, x |\, x \rangle = 1} \|Ax\|$$

for the Euclidean norm $\|x\| := \sqrt{\langle x|x\rangle}$. When A is Hermitian, we have

$$\|A\| = \lambda_{\max}(A) := \max\{|\lambda_j| : \lambda_j \text{ an eigenvalue of } A\}.$$

From an equivalent definition of the norm $\|A\| = \sup_{\langle y|y\rangle = \langle x|x\rangle = 1} |\langle y|A|x\rangle|$, it is easy to see that $\|A^*\| = \|A\|$. For two Hermitian matrices A and B, we have that $\|AB\| = \|(AB)^*\| = \|B^*A^*\| = \|BA\|$. The operator norm is *unitarily invariant*; i.e., for all unitary U, V, $\|A\| = \|UAV\|$ holds. It is easy to show that

$$\left\| \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix} \right\| = \max\left\{\|A\|, \|B\|\right\}.$$

LEMMA 2.3. *Let* X, Y *be any two* $n \times n$ *matrices such that the products* XY *and* YX *are Hermitian. Then, we have*

$$\|XY\| = \|YX\|.$$

Proof. For any two $n \times n$ matrices X and Y, XY and YX have the same eigenvalues; see, e.g., [5, Exercise I.3.7]. Therefore, $\|XY\| = \lambda_{\max}(XY) = \lambda_{\max}(YX) = \|YX\|$. □

A linear operator P such that $P^2 = P$ and $P^* = P$ is called an *orthogonal projector.*

Proposition 2.4. *For two orthogonal projectors A and B, it holds that*

$$\|A+B\| \leq 1 + \|AB\|\,.$$

Proof. We adapt a technique by Kittaneh [28] to our case. We define two 2×2-block matrices X and Y as

$$X := \begin{pmatrix} A & B \\ 0 & 0 \end{pmatrix} \quad \text{and} \quad Y := \begin{pmatrix} A & 0 \\ B & 0 \end{pmatrix}$$

and using $A^2 = A$ and $B^2 = B$, we compute

$$XY := \begin{pmatrix} A+B & 0 \\ 0 & 0 \end{pmatrix} \quad \text{and} \quad YX := \begin{pmatrix} A & AB \\ BA & B \end{pmatrix} = \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix} + \begin{pmatrix} 0 & AB \\ BA & 0 \end{pmatrix}.$$

As A and B are Hermitian, so are $A+B$, AB, BA, XY, and YX. We use Lemma 2.3 and the triangle inequality to obtain

$$\left\| \begin{pmatrix} A+B & 0 \\ 0 & 0 \end{pmatrix} \right\| = \left\| \begin{pmatrix} A & AB \\ BA & B \end{pmatrix} \right\| \leq \left\| \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix} \right\| + \left\| \begin{pmatrix} 0 & AB \\ BA & 0 \end{pmatrix} \right\|.$$

Using the unitary invariance of the operator norm to permute the columns in the rightmost matrix and the facts that $\|A\| = \|B\| = 1$ as well as $\|AB\| = \|BA\|$, we conclude that $\|A+B\| \leq 1 + \|AB\|$. □

3. Uncertainty relations.

In this section, we prove a general uncertainty result and derive from that a corollary that plays the crucial role in the security proof of our protocols. The uncertainty result concerns the situation where the sender holds an arbitrary quantum register of n qubits. He may measure them in either the +- or the ×-basis. We are interested in the distribution of both these measurement results, and we claim that they cannot *both* be "very far from uniform."

3.1. History and previous work.

The history of uncertainty relations starts with Heisenberg, who showed that the outcomes of two noncommuting observables A and B applied to any state ρ are not easy to predict simultaneously. However, Heisenberg speaks only about the variance of the measurement results, and his result was shown to have several shortcomings in [25, 18]. More general forms of uncertainty relations were proposed in [6] and [18] to resolve these problems. The new relations were called *entropic uncertainty relations*, because they are expressed using Shannon entropy instead of the statistical variance. Entropic uncertainty relations have the advantage of being pure information theoretic statements. The first entropic uncertainty relation was introduced by Deutsch [18] and stated that

$$\mathrm{H}(P) + \mathrm{H}(Q) \geq -2 \log \frac{1+c}{2}, \tag{3.1}$$

where P, Q are the distributions representing the measurement results and c is the maximum inner product norm between any eigenvectors of A and B. First conjectured by Kraus [29], Deutsch's relation was improved by Maassen and Uffink [32] to the optimal

$$\mathrm{H}(P) + \mathrm{H}(Q) \geq -2 \log c. \tag{3.2}$$

Although a bound on Shannon entropy can be helpful in some cases, it is usually not good enough in cryptographic applications. The main tool to reduce the adversary's information—privacy amplification [4, 27, 3, 37, 36]—works only if a bound on the adversary's min-entropy (in fact collision entropy) is known. Unfortunately, knowing a lower bound on the Shannon entropy of a distribution does not in general allow one to lower bound its higher order Rényi entropies.

An entropic uncertainty relation involving Rényi entropy of order 2 (i.e., *collision entropy*) was introduced by Larsen [30, 40]. Larsen's relation quantifies precisely the collision entropy for the set $\{A_i\}_{i=1}^{d+1}$ of *all* maximally noncommuting observables, where d is the dimension of the Hilbert space. Its use is therefore restricted to quantum coding schemes that take advantage of *all* $d+1$ observables, i.e., to schemes that are difficult to implement in practice.

3.2. Two mutually unbiased bases. In this section, we show that two distributions obtained by measuring in two mutually unbiased bases cannot *both* be "very far from uniform." One way to express this is to say that a distribution is very nonuniform if one can identify a subset of outcomes that has much higher probability than for a uniform choice. Intuitively, the theorem below says that such sets cannot be found for both measurements. In Appendix A, we generalize the results of this section to more than two mutually unbiased bases.

THEOREM 3.1. *Let ρ be an arbitrary state of n qubits, and let $Q^+(\cdot)$ and $Q^\times(\cdot)$ be the respective distributions of the outcome when ρ is measured in the $+$-basis and the $\times$-basis, respectively. Then, for any two sets $L^+ \subset \{0,1\}^n$ and $L^\times \subset \{0,1\}^n$ it holds that*

$$Q^+(L^+) + Q^\times(L^\times) \leq 1 + 2^{-n/2}\sqrt{|L^+||L^\times|}.$$

Proof. We define the two orthogonal projectors

$$A := \sum_{x \in L^+} |x\rangle\langle x| \quad \text{and} \quad B := \sum_{y \in L^\times} H^{\otimes n}|y\rangle\langle y|H^{\otimes n}.$$

Using the spectral decomposition of $\rho = \sum_w \lambda_w |\varphi_w\rangle\langle\varphi_w|$, we have

$$\begin{aligned} Q^+(L^+) + Q^\times(L^\times) &= \mathrm{tr}\,(A\rho) + \mathrm{tr}\,(B\rho) \\ &= \sum_w \lambda_w \left(\mathrm{tr}\,(A|\varphi_w\rangle\langle\varphi_w|) + \mathrm{tr}\,(B|\varphi_w\rangle\langle\varphi_w|)\right) \\ &= \sum_w \lambda_w \left(\langle\varphi_w|A|\varphi_w\rangle + \langle\varphi_w|B|\varphi_w\rangle\right) \\ &= \sum_w \lambda_w \langle\varphi_w|(A+B)|\varphi_w\rangle \\ &\leq \|A+B\| \leq 1 + \|AB\|, \end{aligned}$$

where the last line is Proposition 2.4. In order to finish the proof, we show that $\|AB\| \leq 2^{-n/2}\sqrt{|L^+||L^\times|}$. Note that an arbitrary state $|\psi\rangle = \sum_z \lambda_z H^{\otimes n}|z\rangle$ can be expressed with coordinates λ_z in the diagonal basis. Then, with the sums over x and y understood as over $x \in L^+$ and $y \in L^\times$, respectively,

$$\begin{aligned} \big\|AB|\psi\rangle\big\| &= \left\|\sum_{x,y} |x\rangle\langle x|H^{\otimes n}|y\rangle\langle y|H^{\otimes n}|\psi\rangle\right\| = 2^{-n/2}\left\|\sum_{x,y} |x\rangle\langle y|H^{\otimes n}|\psi\rangle\right\| \\ &= 2^{-n/2}\left\|\sum_x |x\rangle\right\|\left|\sum_y \lambda_y\right| \leq 2^{-n/2}\sqrt{|L^+|}\sum_y |\lambda_y| \leq 2^{-n/2}\sqrt{|L^+||L^\times|}. \end{aligned}$$

The second equality holds since $|x\rangle$ and $H^{\otimes n}|y\rangle$ are mutually unbiased, the first inequality follows from Pythagoras and the triangle inequality, and the last inequality follows from Cauchy–Schwarz. This implies that $\|AB\| \leq 2^{-n/2}\sqrt{|L^+||L^\times|}$ and finishes the proof. □

This theorem yields a meaningful bound as long as $|L^+| \cdot |L^\times| < 2^n$, e.g., if L^+ and $L^\times$ both contain less than $2^{n/2}$ elements. The relation is tight in the sense that for the Hadamard-invariant state

$$|\varphi\rangle = \Big(|0\rangle^{\otimes n} + (H|0\rangle)^{\otimes n}\Big)/\sqrt{2(1+2^{-n/2})}$$

and $L^+ = L^\times = \{0^n\}$, it is straightforward to verify that $Q^+(L^+) = Q^\times(L^\times) = (1 + 2^{-n/2})/2$ and therefore $Q^+(L^+) + Q^\times(L^\times) = 1 + 2^{-n/2}$. Another state that achieves equality (for n even) is $|\varphi\rangle = |0\rangle^{\otimes n/2} \otimes (H|0\rangle)^{\otimes n/2}$ with $L^+ = \{0^{n/2}x \mid x \in \{0,1\}^{n/2}\}$ and $L^\times = \{x0^{n/2} \mid x \in \{0,1\}^{n/2}\}$. We get that $Q^+(L^+) = Q^\times(L^\times) = 1$ and thus $Q^+(L^+) + Q^\times(L^\times) = 2 = 1 + 2^{-n/2}\sqrt{2^n}$.

If for $r \in \{+,\times\}$, L^r contains only the n-bit string with the maximal probability of Q^r, we obtain a known tight relation (see inequality (9) in [32]).

Corollary 3.2. *Let q^+_∞ and $q^\times_\infty$ be the maximal probabilities of the distributions Q^+ and $Q^\times$ from above. It then holds that $q^+_\infty \cdot q^\times_\infty \leq \frac{1}{4}(1+c)^2$, where $c = 2^{-n/2}$.*

Equality is achieved for the same state $|\varphi\rangle = \big(|0\rangle^{\otimes n} + (H|0\rangle)^{\otimes n}\big)/\sqrt{2(1+2^{-n/2})}$ as above.

The following corollary plays a crucial role in the security proof of the OT protocol in the next section.

Corollary 3.3. *Let R be a random variable over $\{+,\times\}$, and let X be the outcome when ρ is measured in basis R, such that $P_{X|R}(x|r) = Q^r(x)$. Then, for any $\lambda < \frac{1}{2}$ there exists an event $\mathcal{E}$ such that*

$$P[\mathcal{E}|R{=}+] + P[\mathcal{E}|R{=}\times] \geq 1 - negl(n)$$

and thus $P[\mathcal{E}] \geq \frac{1}{2} - negl(n)$ in case R is uniform, and such that

$$\mathrm{H}_\infty(X|R{=}r,\mathcal{E}) \geq \lambda n$$

for $r \in \{+,\times\}$ with $P_{R|\mathcal{E}}(r) > 0$.

Proof. Choose $\varepsilon > 0$ such that $\lambda + \varepsilon < \frac{1}{2}$, define

$$S^+ := \left\{x \in \{0,1\}^n : Q^+(x) \leq 2^{-(\lambda+\varepsilon)n}\right\} \text{ and}$$
$$S^\times := \left\{z \in \{0,1\}^n : Q^\times(z) \leq 2^{-(\lambda+\varepsilon)n}\right\}$$

to be the sets of strings with small probabilities, and denote by $L^+ := \overline{S}^+$ and $L^\times := \overline{S}^\times$ their complements.[2] Note that for all $x \in L^+$, we have that $Q^+(x) > 2^{-(\lambda+\varepsilon)n}$ and therefore $|L^+| < 2^{(\lambda+\varepsilon)n}$. Analogously, we have $|L^\times| < 2^{(\lambda+\varepsilon)n}$. For ease of notation, we abbreviate the probabilities that strings with small probabilities occur with $q^+ := Q^+(S^+)$ and $q^\times := Q^\times(S^\times)$. It follows immediately from Theorem 3.1 that $q^+ + q^\times \geq 1 - negl(n)$.

We define $\mathcal{E}$ to be the event $X \in S^R$. Then $P[\mathcal{E}|R{=}+] = P[X \in S^+|R{=}+] = q^+$ and similarly $P[\mathcal{E}|R{=}\times] = q^\times$, and the first claim follows immediately. Furthermore,

[2] Here is the mnemonic: S for the strings with Small probabilities, L for Large.

if R is uniformly distributed, then $P[\mathcal{E}] = P[\mathcal{E}|R{=}+]P_R(+) + P[\mathcal{E}|R{=}\times]P_R(\times) = \frac{1}{2}(q^+ + q^\times) \geq \frac{1}{2} - \mathit{negl}(n)$. Regarding the second claim, in case $R = +$, we have

$$\begin{aligned}\mathrm{H}_\infty(X|R{=}+,\mathcal{E}) &= -\log\left(\max_{x\in S^+}\frac{Q^+(x)}{q^+}\right)\\ &\geq -\log\left(\frac{2^{-(\lambda+\varepsilon)n}}{q^+}\right) = \lambda n + \varepsilon n + \log(q^+).\end{aligned}$$

Thus, if $q^+ \geq 2^{-\varepsilon n}$, then indeed $\mathrm{H}_\infty(X|R{=}+, X \in S^+) \geq \lambda n$. The corresponding holds for the case $R = \times$.

Finally, if $q^+ < 2^{-\varepsilon n}$ (or similarly $q^\times < 2^{-\varepsilon n}$), then instead of the above, we define $\mathcal{E}$ as the *empty event* if $R = +$ and as the event $X \in S^\times$ if $R = \times$. It follows that $P[\mathcal{E}|R = +] = 0$ and $P[\mathcal{E}|R = \times] = q^\times \geq 1 - \mathit{negl}(n)$, as well as $\mathrm{H}_\infty(X|R = \times, \mathcal{E}) = \mathrm{H}_\infty(X|R{=}\times, X \in S^\times) \geq \lambda n + \varepsilon n + \log(q^\times) \geq \lambda n$ (for n large enough), both by the bound on $q^+ + q^\times$ and on q^+, whereas $P_{R|\mathcal{E}}(+) = 0$. □

4. Rabin oblivious transfer.

4.1. The definition.

A protocol for Rabin oblivious transfer (ROT) between sender Alice and receiver Bob allows for Alice to send a bit b through an erasure channel to Bob. Each transmission delivers b or an erasure with probability $\frac{1}{2}$. Intuitively, a protocol for ROT is secure if

(i) the sender Alice gets no information on whether b was received or not, no matter what she does, and

(ii) the receiver Bob gets no information about b with probability at least $\frac{1}{2}$, no matter what he does.

In this paper, we are considering quantum protocols for ROT. This means that while the inputs and outputs of the honest senders are classical, described by random variables, the protocol may contain quantum computation and quantum communication, and the view of a dishonest player is quantum and is thus described by a quantum state.

Any such (two-party) protocol is specified by a family $\{(\mathsf{S}_n, \mathsf{R}_n)\}_{n>0}$ of pairs of interactive quantum circuits (i.e., interacting through a quantum channel). Each pair is indexed by a security parameter $n > 0$, where S_n and R_n denote the circuits for sender Alice and receiver Bob, respectively. In order to simplify the notation, we often omit the index n, leaving the dependency on it implicit.

For the formal definition of the security requirements of a ROT protocol, let us fix the following notation. Let B denote the binary random variable describing S's input bit b, and let A and Y denote the binary random variables describing R's two output bits, where the meaning is that A indicates whether the bit was received or not. Furthermore, for a dishonest sender $\tilde{\mathsf{S}}$, we have the ccq-state $\rho_{AY\tilde{\mathsf{S}}}$, where (by slight abuse of notation) we also denote by $\tilde{\mathsf{S}}$ the quantum register that the sender outputs. Its state may depend on A and Y. Similarly, for a dishonest receiver $\tilde{\mathsf{R}}$, we have the cq-state $\rho_{B\tilde{\mathsf{R}}}$.

DEFINITION 4.1. *A* (statistically) secure ROT *is a two-party (quantum) protocol* (S, R) *with the following properties.*

Correctness: *For honest* S *and* R*,* $P[B = Y|A = 1] \geq 1 - \mathit{negl}(n)$*.*

Receiver-security: *For honest* R *and any dishonest* $\tilde{\mathsf{S}}$*, there exists a binary random variable* B' *such that* $P[B' = Y|A = 1] \geq 1 - \mathit{negl}(n)$ *and* $\delta\big(\rho_{AB'\tilde{\mathsf{S}}}, \mathbb{1}\otimes\rho_{B'\tilde{\mathsf{S}}}\big) \leq \mathit{negl}(n)$*.*

Sender-security: *For any* $\tilde{\mathsf{R}}$, *there exists an event* $\mathcal{E}$ *with* $P[\mathcal{E}] \geq \frac{1}{2} - negl(n)$ *such that* $\delta\big(\rho_{B\tilde{\mathsf{R}}|\mathcal{E}}, \rho_B \otimes \rho_{\tilde{\mathsf{R}}|\mathcal{E}}\big) \leq negl(n)$.

If any of the negligible terms above equals 0*, then the corresponding property is said to hold* perfectly*. If one of the properties holds only with respect to a restricted class* $\mathfrak{S}$ *of* $\tilde{\mathsf{S}}$*'s (resp.,* $\mathfrak{R}$ *of* $\tilde{\mathsf{R}}$*'s), then this property is said to hold and the protocol is said to be secure* against $\mathfrak{S}$ *(resp.,* $\mathfrak{R}$*).*

Statistical receiver-security guarantees that the joint quantum state after the execution of the protocol is, up to a negligible difference, the same as when the dishonest sender prepares the cq-state $\rho_{B'\tilde{\mathsf{S}}}$, and gives the classical bit B' to an ideal functionality which then passes it on to the receiver with probability $\frac{1}{2}$.[3] Statistical sender-security guarantees that the joint quantum state is, up to a negligible difference, the same as when the dishonest receiver gets the sender's bit B with probability at most $\frac{1}{2}$ and prepares the state $\rho_{\tilde{\mathsf{R}}|\mathcal{E}}$ in case he does not receive it, and else the state $\rho^b_{\tilde{\mathsf{R}}|\bar{\mathcal{E}}} = \rho_{\tilde{\mathsf{R}}|B=b,\bar{\mathcal{E}}}$ if $B = b$. In other words, security guarantees that the dishonest party cannot do more than when attacking an ideal functionality.

A formal treatment of the *composability* is beyond the scope of this paper. However, upcoming work of the authors implies that any quantum ROT protocol which satisfies Definition 4.1 securely replaces an ideal ROT functionality when used sequentially in a purely *classical* protocol. We also refer to [43] for recent results about the composition of quantum protocols in the bounded-quantum-storage model.

4.2. The protocol. We introduce a quantum protocol for ROT that will be shown to be perfectly receiver-secure (against any sender) and statistically sender-secure against any quantum-memory-bounded receiver. Our protocol exhibits some similarity to quantum conjugate coding introduced by Wiesner [44].

The protocol is very simple (see Figure 4.1): S picks $x \in_R \{0,1\}^n$ and sends to R n qubits in either state $|x\rangle_+$ or $|x\rangle_\times$, each chosen with probability $\frac{1}{2}$. R then measures all received qubits either in the rectilinear or in the diagonal basis. With probability $\frac{1}{2}$, R picks the right basis and gets x, while any $\tilde{\mathsf{R}}$ that is forced to measure part of the state (due to a memory bound) can have full information on x only in case the $+$-basis was used *or* in case the $\times$-basis was used (but not in both cases). Privacy amplification based on any two-universal class of hashing functions $\mathcal{F}_n$ is then used to destroy partial information. (In order to avoid aborting, we specify that if a dishonest $\tilde{\mathsf{S}}$ refuses to participate or sends data in incorrect format, then R samples both of its output bits a and y at random in $\{0,1\}$.)

We first consider receiver-security.

PROPOSITION 4.2. QOT *is perfectly receiver-secure.*

It is obvious that no information about whether R has received the bit is leaked to any sender $\tilde{\mathsf{S}}$, since R does not send anything. However, one needs to show the existence of a random variable B' as required by receiver-security.

Proof. Recall that the density matrix $\rho_{AY\tilde{\mathsf{S}}}$ is defined by the experiment where the dishonest sender $\tilde{\mathsf{S}}$ interacts with the honest memory-bounded R. Consider a modification of the experiment where we allow R to be *unbounded* in memory and where R waits to receive r and then measures all qubits in basis r. Let X' be the

[3] Note that the original definition given in [17] does not guarantee that the distribution of the input bit is determined at the end of the execution of ROT. This is a strictly weaker definition and does not fully capture what is expected from a ROT: it is easy to see that if the dishonest sender can still influence his input bit after the execution of the protocol, then known schemes based on ROT, such as bit commitments, are not secure anymore. The security definition given here is in the spirit of the security definition from [16] for 1-2 OT.

QOT(b):

1. S picks $x \in_R \{0,1\}^n$, and $r \in_R \{+,\times\}$.
2. S sends $|\psi\rangle := |x\rangle_r$ to R (i.e., the string x in basis r).
3. R picks $r' \in_R \{+,\times\}$ and measures all qubits of $|\psi\rangle$ in basis r'. Let $x' \in \{0,1\}^n$ be the result.
4. S announces r, $f \in_R \mathcal{F}_n$, and $e := b \oplus f(x)$.
5. R outputs $a := 1$ and $y := e \oplus f(x')$ if $r' = r$ and else $a := 0$ and $y := 0$.

FIG. 4.1. *Protocol for quantum Rabin OT.*

resulting string. Nevertheless, R picks $r' \in_R \{+,\times\}$ at random and outputs $(A,Y) = (0,0)$ if $r' \neq r$ and $(A,Y) = (1, e \oplus f(X'))$ if $r' = r$. Since the only difference between the two experiments is *when* R measures the qubits and *in what basis* R measures them when $r \neq r'$, in which case his final output is independent of the measurement outcome, the two experiments result in the same $\rho_{AY\tilde{\mathsf{S}}}$. However, in the modified experiment we can choose B' to be $e \oplus f(X')$ such that by construction $B' = Y$ if $A = 1$ and A is uniformly distributed, independent of anything, and thus $\rho_{AB'\tilde{\mathsf{S}}} = \mathbb{1} \otimes \rho_{B'\tilde{\mathsf{S}}}$. □

As we shall see in section 4.4, the security of the QOT protocol against receivers with bounded-size quantum memory holds as long as the bound applies before step 4 is reached. An equivalent protocol is obtained by purifying the sender's actions. Although QOT is easy to implement, the purified or EPR-based version [23] depicted in Figure 4.2 is easier to prove secure. A similar approach was taken in the Shor–Preskill proof of security for the BB84 quantum key distribution scheme [41].

EPR-QOT(b):

1. S prepares n EPR pairs each in state $|\Omega\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.
2. S sends one half of each pair to R and keeps the other halves.
3. R picks $r' \in_R \{+,\times\}$ and measures all received qubits in basis r'. Let $x' \in \{0,1\}^n$ be the result.
4. S picks $r \in_R \{+,\times\}$ and measures all kept qubits in basis r. Let $x \in \{0,1\}^n$ be the outcome. S announces r, $f \in_R \mathcal{F}_n$, and $e := b \oplus f(x)$.
5. R outputs $a := 1$ and $y := e \oplus f(x')$ if $r' = r$ and else $a := 0$ and $y := 0$.

FIG. 4.2. *Protocol for EPR-based quantum Rabin OT.*

Notice that while QOT requires no quantum memory for honest players, quantum memory for S seems to be required in EPR-QOT. The following lemma shows the strict equivalence between QOT and EPR-QOT.

LEMMA 4.3. QOT *is sender-secure if and only if* EPR-QOT *is.*

Proof. The proof follows easily after observing that S's choices of r and f, together with the measurements, all commute with $\tilde{\mathsf{R}}$'s actions. Therefore, they can be performed right after step 1 with no change for $\tilde{\mathsf{R}}$'s view. Modifying EPR-QOT that way results in QOT. □

Note that for a dishonest receiver it is not only irrelevant whether he tries to attack QOT or EPR-QOT, but in fact there is no difference in the two protocols from his point of view.

4.3. Modeling dishonest receivers. We model dishonest receivers in QOT, respectively, EPR-QOT, under the assumption that the maximum size of their quantum storage is bounded. These adversaries are required to have bounded quantum storage only when they reach step 4 in (EPR-)QOT. Before that, the adversary can store and carry out quantum computations involving any number of qubits. Apart from the restriction on the size of the quantum memory available to the adversary, no other assumption is made. In particular, the adversary is not assumed to be computationally bounded, and the size of its classical memory is not restricted.

DEFINITION 4.4. *The set $\mathfrak{R}_\gamma$ denotes all possible quantum dishonest receivers $\{\tilde{\mathsf{R}}_n\}_{n>0}$ in* QOT *or* EPR-QOT *where for each $n > 0$, $\tilde{\mathsf{R}}_n$ has quantum memory of size at most γn when step 4 is reached.*

In general, the adversary $\tilde{\mathsf{R}}$ is allowed to perform any quantum computation compressing the n qubits received from S into a quantum register M of size at most γn when step 4 is reached. More precisely, the compression function is implemented by some unitary transform C acting upon the quantum state received and an ancilla of arbitrary size. The compression is performed by a measurement that we assume in the computational basis without loss of generality. Before starting step 4, the adversary first applies a unitary transform C:

$$2^{-n/2}\sum_{x\in\{0,1\}^n}|x\rangle\otimes C|x\rangle|0\rangle \mapsto 2^{-n/2}\sum_{x\in\{0,1\}^n}|x\rangle\otimes\sum_y \alpha_{x,y}|\varphi_{x,y}\rangle^M|y\rangle^Y,$$

where for all x, $\sum_y |\alpha_{x,y}|^2 = 1$. Then, a measurement in the computational basis is applied to register Y providing classical outcome y. The result is a quantum state in register M of size γn qubits. Ignoring the value of y to ease the notation, the renormalized state of the system in its most general form when step 4 in EPR-QOT is reached is thus of the form

$$|\psi\rangle = \sum_{x\in\{0,1\}^n}\alpha_x|x\rangle\otimes|\varphi_x\rangle^M,$$

where $\sum_x |\alpha_x|^2 = 1$. We will prove security for any such state $|\psi\rangle$ and thus conditioned on any value y that may be observed. It is therefore safe to leave the dependency on y implicit.

4.4. Security against dishonest receivers. In this section, we show that EPR-QOT is secure against any dishonest receiver having access to a quantum storage device of size strictly smaller than half the number of qubits received at step 2.

THEOREM 4.5. *For all $\gamma < \frac{1}{2}$,* QOT *is statistically secure against $\mathfrak{R}_\gamma$.*

Proof. After Lemmas 4.3 and 4.2, it remains to show that EPR-QOT is sender-secure against $\mathfrak{R}_\gamma$. Since $\gamma < \frac{1}{2}$, we can find $\varepsilon > 0$ with $\gamma+\varepsilon < \frac{1}{2}$. Consider a dishonest receiver $\tilde{\mathsf{R}}$ in EPR-QOT with quantum memory of size γn. Let R and X denote the random variables describing the basis r and the outcome x of S's measurement (in basis r) in step 4 of EPR-QOT, respectively. We implicitly understand the distribution of X given R to be conditioned on the classical outcome y of the measurement $\tilde{\mathsf{R}}$ performed when the memory bound applies, as described in section 4.3; the following analysis works no matter what y is. Corollary 3.3 with $\lambda = \gamma+\varepsilon$ implies the existence of an event $\mathcal{E}$ such that $P[\mathcal{E}] \geq \frac{1}{2} - negl(n)$ and such that $\mathrm{H}_\infty(X|R{=}r,\mathcal{E}) \geq \gamma n + \varepsilon n$ for any relevant r. Note that by construction, the random variables X and R, and thus also the event $\mathcal{E}$, are independent of the sender's input bit B, and hence $\rho_{B|\mathcal{E}} = \rho_B$. It remains to show that $\delta(\rho_{B\tilde{\mathsf{R}}|\mathcal{E}}, \rho_{B|\mathcal{E}}\otimes\rho_{\tilde{\mathsf{R}}|\mathcal{E}}) \leq negl(n)$. As the bit B is masked by the output of the hash function $F(X)$ in step 4 of EPR-QOT (where the random variable F represents the random choice for f), it suffices to show that $F(X)$ is close to uniform and essentially independent from $\tilde{\mathsf{R}}$'s view, conditioned on $\mathcal{E}$. But this is guaranteed by the above bound on $\mathrm{H}_\infty(X|R{=}r,\mathcal{E})$ and by Theorem 2.1. $\square$

4.5. On the necessity of privacy amplification. In this section, we show that randomized privacy amplification is needed for protocol QOT to be secure. For instance, it is tempting to believe that the sender could use the XOR $\bigoplus_i x_i$ in order to mask the bit b, rather than $f(x)$ for a randomly sampled $f \in \mathcal{F}_n$. This would reduce the communication complexity as well as the number of random coins needed. However, we argue in this section that this is not secure (against an adversary as we model it). Indeed, somewhat surprisingly, this variant can be broken by a dishonest receiver that has *no quantum memory at all* (but that can do coherent measurements on pairs of qubits) in the case n is even. For odd n, the dishonest receiver needs to store *a single qubit*.

Clearly, a dishonest receiver can break the modified scheme QOT and learn the bit b with probability 1 if he can compute $\bigoplus_i x_i$ with probability 1. Note that, using the equivalence between QOT and EPR-QOT, x_i can be understood as the outcome of the measurement in either the +- or the ×-basis, performed by the sender on one part of an EPR pair while the other has been handed over to the receiver. The following proposition shows that indeed the receiver can learn $\bigoplus_i x_i$ by a suitable measurement of his parts of the EPR pairs. Concretely, he measures the qubits he receives pairwise by a suitable measurement which allows him to learn the XOR of the two corresponding x_i's, no matter what the basis is (and he needs to store one single qubit in case n is odd). This obviously allows him to learn the XOR of all x_i's in all cases.

PROPOSITION 4.6. *Consider two EPR pairs, i.e.,* $|\psi\rangle = \frac{1}{2}\sum_x |x\rangle^S |x\rangle^R$, *where* x *ranges over* $\{0,1\}^2$. *Let* $r \in \{+,\times\}$, *and let* x_1 *and* x_2 *be the result when measuring the two qubits in register* S *in basis* r. *There exists a fixed measurement for register* R *so that the outcome together with* r *uniquely determines* $x_1 \oplus x_2$.

Proof. The measurement that does the job is the *Bell measurement*, i.e., the measurement in the Bell basis $\{|\Phi^+\rangle, |\Psi^+\rangle, |\Phi^-\rangle, |\Psi^-\rangle\}$. Recall that

$$\begin{aligned}
\big|\Phi^+\big\rangle &= \frac{1}{\sqrt{2}}\big(|00\rangle_+ + |11\rangle_+\big) = \frac{1}{\sqrt{2}}\big(|00\rangle_\times + |11\rangle_\times\big),\\
\big|\Psi^+\big\rangle &= \frac{1}{\sqrt{2}}\big(|01\rangle_+ + |10\rangle_+\big) = \frac{1}{\sqrt{2}}\big(|00\rangle_\times - |11\rangle_\times\big),\\
\big|\Phi^-\big\rangle &= \frac{1}{\sqrt{2}}\big(|00\rangle_+ - |11\rangle_+\big) = \frac{1}{\sqrt{2}}\big(|01\rangle_\times + |10\rangle_\times\big),\\
\big|\Psi^-\big\rangle &= \frac{1}{\sqrt{2}}\big(|01\rangle_+ - |10\rangle_+\big) = \frac{1}{\sqrt{2}}\big(|10\rangle_\times - |01\rangle_\times\big).
\end{aligned}$$

Due to the special form of the Bell basis, when register R is measured and, as a consequence, one of the four Bell states is observed, the state in register S collapses to that *same* Bell state. Indeed, when doing the basis transformation, all cross-products cancel each other out. It now follows by inspection that knowledge of the Bell state and the basis r allows one to predict the XOR of the two bits observed when measuring the Bell state in basis r. For instance, for the Bell state $|\Psi^+\rangle$, the XOR is 1 if $r = +$ and 0 if $r = \times$. $\square$

Note that from the above proof one can see that the receiver's attack (resp., his measurement on each pair of qubits) can be understood as teleporting one of the two entangled qubits from the receiver to the sender using the other as EPR pair. However, the receiver does not send the outcome of his measurement to the sender, but keeps it in order to predict the XOR.

Clearly, the same strategy also works against any fixed linear function. Therefore, the only hope for doing deterministic privacy amplification is by using a nonlinear function. However, it has been shown recently in [1] that this approach is also doomed to fail in our scenario, because the outcome of *any Boolean function* can be perfectly predicted by a dishonest receiver who can store a single qubit and later learns the correct basis $r \in \{+, \times\}$.

4.6. Weakening the assumptions. Observe that QOT requires error-free quantum communication, in that a transmitted bit b that is encoded by the sender and measured by the receiver using the same basis is always received as b. In addition, it also requires a perfect quantum source which on request produces *one and only one* qubit in the right state, e.g., *one* photon with the right polarization. Indeed, in case of noisy quantum communication, an honest receiver in QOT is likely to receive an incorrect bit, and the sender-security of QOT is vulnerable to imperfect sources that once in a while transmit more than one qubit in the same state: a malicious receiver $\tilde{\mathsf{R}}$ can easily determine the basis $r \in \{+, \times\}$ and measure all the following qubits in the right basis. However, current technology only allows one to approximate the behavior of single-photon sources and noise-free quantum communication. It would be preferable to find a variant of QOT that allows one to weaken the technological requirements put upon the honest parties.

In this section, we present such a protocol based on BB84 states [2], BB84-QOT (see Figure 4.3). The security proof follows essentially by adapting the security analysis of QOT in a rather straightforward way, as will be discussed later.

Let us consider a quantum channel with an error probability $\phi < \frac{1}{2}$; i.e., ϕ denotes the probability that a transmitted bit b that is encoded by the sender and measured by the receiver using the same basis is received as $1 - b$. For the sake of simplicity we assume that the error rate is the same for qubits encoded in the $+$- and $\times$-basis. It is straightforward to adapt the analysis below to basis-dependent error rates. In order to not have the security rely on any level of noise, we assume the error probability to be zero when considering a *dishonest* receiver. Also, let us consider a quantum source which produces two or more qubits (in the same state), rather than just one, with probability $\eta < 1 - \phi$. We assume that the parameters ϕ and η which describe the precision of the physical apparatus being used are known to the players.

We call this the (ϕ, η)-weak quantum model. By adjusting the parameters, this model can also cope with dark counts and empty pulses; see section 6.1.

In order to deal with noisy quantum communication, we need to do error-correction without giving the adversary too much information. Techniques for solving this problem are known as *information reconciliation* (e.g., [9]) or as *secure sketches* [20]. Let $x \in \{0,1\}^\ell$ be an arbitrary string, and let $x' \in \{0,1\}^\ell$ be the result of flipping every bit in x (independently) with probability ϕ. It is well known that learning the syndrome $S(x)$ of x, with respect to a suitable efficiently decodable linear error-correcting code C of length ℓ, allows us to recover x from x', except with negligible probability in ℓ (e.g., [33, 12, 20]). Furthermore, it is known from coding theory that, for large enough ℓ, such a code can be chosen with rate R arbitrarily close to but smaller than $1 - h(\phi)$, i.e., such that the syndrome length s is bounded by $s < (h(\phi) + \varepsilon)\ell$, where $\varepsilon > 0$ (see, e.g., [12] or the full version of [20] and the references therein).

Regarding the loss of information, we can analyze privacy amplification in a similar way as before, just by appending a register for the syndrome $S(x)$ to the quantum register E. Using that $S_0(\rho_{S(X)\mathsf{E}}) \leq q + s$, Theorem 2.1 then reads

$$\delta\big(\rho_{F(X)FS(X)\mathsf{E}}, \mathbb{1} \otimes \rho_{FS(X)\mathsf{E}}\big) \leq \frac{1}{2} 2^{-\frac{1}{2}(\mathrm{H}_\infty(X)-q-s-1)}. \tag{4.1}$$

Consider the protocol BB84-QOT in the (ϕ, η)-weak quantum model shown in Figure 4.3. The protocol uses an efficiently decodable linear code C_ℓ, parametrized in $\ell \in \mathbb{N}$, with codeword length ℓ, rate $R = 1 - h(\phi) - \varepsilon$ for some small $\varepsilon > 0$, and the ability to correct errors occurring with probability ϕ (except with negligible probability). Let S_ℓ be the corresponding syndrome function. As before, the memory bound in BB84-QOT applies before step 4.

```
BB84-QOT($b$):
    1. S picks $x \in_R \{0,1\}^n$ and $\theta \in_R \{+,\times\}^n$.
    2. S sends $x_i$ in the corresponding bases $|x_1\rangle_{\theta_1}, \dots, |x_n\rangle_{\theta_n}$ to R.
    3. R picks $r' \in_R \{+,\times\}$ and measures all qubits in basis $r'$. Let $x' \in \{0,1\}^n$
       be the result.
    4. S picks $r \in_R \{+,\times\}$, sets $I := \{i : \theta_i = \{+,\times\}_{[r]}\}$ and $\ell := |I|$, and
       announces $r$, $I$, $syn := S_\ell(x|_I)$, $f \in_R \mathcal{F}_\ell$, and $e := b \oplus f(x|_I)$.
    5. R recovers $x|_I$ from $x'|_I$ and $syn$, and outputs $a := 1$ and $b' := e \oplus f(x|_I)$
       if $r' = r$ and else $a := 0$ and $b' := 0$.
```

FIG. 4.3. *Protocol for the BB84 version of quantum Rabin OT.*

By the abovementioned properties of the code C_ℓ, it is obvious that R receives the correct bit b if $r' = r$, except with negligible probability. (The error probability is negligible in ℓ, but by Bernstein's law of large numbers, ℓ is linear in n except with negligible probability.) Also, since there is no communication from R to S, a dishonest sender $\tilde{\mathsf{S}}$ cannot learn whether R received the bit. In fact, BB84-QOT can be shown to be perfectly receiver-secure in the same way as in Proposition 4.2. In a manner similar to that for protocol QOT, in order to argue about sender-security we compare BB84-QOT with a purified version shown in Figure 4.4. BB84-EPR-QOT runs in the $(\phi, 0)$-weak quantum model, and the imperfectness of the quantum source assumed in BB84-QOT is simulated by S in BB84-EPR-QOT so that there is no difference from R's point of view.

The security equivalence between BB84-QOT (in the (ϕ, η)-weak quantum model) and BB84-EPR-QOT (in the $(\phi, 0)$-weak quantum model) is omitted here as it follows essentially along the same lines as in section 4.2.

THEOREM 4.7. *In the (ϕ, η)-weak quantum model,* BB84-QOT *is statistically secure against $\mathfrak{R}_\gamma$ for any $\gamma < \frac{1-\eta}{4} - \frac{h(\phi)}{2}$ (if parameter ε is chosen small enough).*

Proof sketch. It remains to show that BB84-EPR-QOT is statistically sender-secure against $\mathfrak{R}_\gamma$ (in the $(\phi, 0)$-weak quantum model). The reasoning goes exactly along the lines of the proof of Theorem 4.5, except that we restrict our attention to those i's which are in J. By Bernstein's law of large numbers, ℓ lies within $(1 \pm \varepsilon)n/2$ and $|J|$ within $(1 - \eta \pm \varepsilon)n/2$ except with negligible probability. In order to make the proof easier to read, we assume that $\ell = n/2$ and $|J| = (1 - \eta)n/2$ and also treat the ε occurring in the rate of the code C_ℓ as zero. For the full proof, we simply need to carry the ε's along and then choose them small enough at the end of the proof.

BB84-EPR-QOT(b):

1. S prepares n EPR pairs each in state $|\Omega\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. Additionally, S initializes $I'_{+} := \emptyset$ and $I'_{\times} := \emptyset$.
2. For every $i \in \{1, \dots, n\}$, S does the following. With probability $1 - \eta$, S sends one half of the ith pair to R and keeps the other half. While with probability η, S picks $\theta_i \in_R \{+, \times\}$, replaces I'_{θ_i} by $I'_{\theta_i} \cup \{i\}$, and sends two or more qubits in the same state $|x_i\rangle_{\theta_i}$ to R, where $x_i \in_R \{0, 1\}$.
3. R picks $r' \in_R \{+, \times\}$ and measures all received qubits in basis r'. Let $x' \in \{0, 1\}^n$ be the result.
4. S picks a random index set $J \subset_R \{1, \dots, n\} \setminus (I'_{+} \cup I'_{\times})$. Then, it picks $r \in_R \{+, \times\}$, sets $I := J \cup I'_r$ and $\ell := |I|$, and for each $i \in J$ measures the corresponding qubit in basis r. Let x_i be the corresponding outcome, and let $x|_I$ be the collection of all x_i's with $i \in I$. S announces r, I, $syn = S_\ell(x|_I)$, $f \in_R \mathcal{F}_\ell$, and $e = b \oplus f(x|_I)$.
5. R recovers $x|_I$ from $x'|_I$ and syn, and outputs $a := 1$ and $b' := e \oplus f(x|_I)$, if $r' = r$ and else $a := 0$ and $b' := 0$.

FIG. 4.4. *Protocol for EPR-based quantum Rabin OT, BB84 version.*

Write $n' = |J| = (1 - \eta)n/2$, and let γ' be such that $\gamma n = \gamma' n'$, i.e., $\gamma' = 2\gamma/(1 - \eta)$. Assume $\kappa > 0$ such that $\gamma' + \kappa < \frac{1}{2}$, where we make sure later that such κ exists. It then follows from Corollary 3.3 that there exists an event $\mathcal{E}$ such that $P[\mathcal{E}] \geq \frac{1}{2} - negl(n') = \frac{1}{2} - negl(n)$ and

$$\mathrm{H}_\infty\big(X|_J \big| R{=}r, \mathcal{E}\big) \geq (\gamma' + \kappa)n' = \gamma n + \kappa(1 - \eta)n/2 \, .$$

By (4.1), it remains to argue that this is larger than $q + s = \gamma n + h(\phi)n/2$; i.e.,

$$\kappa(1 - \eta) > h(\phi) \, ,$$

where κ has to satisfy

$$\kappa < \frac{1}{2} - \gamma' = \frac{1}{2} - \frac{2}{\gamma/(1 - \eta)} .$$

This can obviously be achieved (by choosing κ appropriately) if and only if the claimed bound on γ holds. □

5. Quantum commitment scheme. In this section, we present a BC scheme from a committer C with bounded quantum memory to an unbounded receiver V. The scheme is peculiar since in order to commit to a bit, the committer does not send anything. During the committing stage information goes only from V to C. The security analysis of the scheme uses similar techniques as the analysis of EPR-QOT.

5.1. The protocol. The objective of this section is to present a bounded-quantum-memory BC scheme COMM (see Figure 5.1). Intuitively, a commitment to a bit b is made by measuring random BB84-states in basis $\{+, \times\}_{[b]}$.

It is clear that EPR-COMM is hiding, i.e., that the commit phase reveals no information on the committed bit, since no information is transmitted to V at all. Hence we have the following lemma.

COMM(b):

1. V picks $x \in_R \{0,1\}^n$ and $r \in_R \{+,\times\}^n$.
2. V sends x_i in the corresponding bases $|x_1\rangle_{r_1}, |x_2\rangle_{r_2}, \ldots, |x_n\rangle_{r_n}$ to C.
3. C commits to the bit b by measuring all qubits in basis $\{+,\times\}_{[b]}$. Let $x' \in \{0,1\}^n$ be the result.
4. To open the commitment, C sends b and x' to V.
5. V verifies that $x_i = x'_i$ for those i where $r_i = \{+,\times\}_{[b]}$. V accepts if and only if this is the case.

FIG. 5.1. *Protocol for quantum commitment.*

LEMMA 5.1. EPR-COMM *is perfectly hiding.*

As for the OT-protocol of section 4.2, we present an equivalent EPR-version of the protocol that is easier to analyze (see Figure 5.2).

EPR-COMM(b):

1. V prepares n EPR pairs each in state $|\Omega\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.
2. V sends one half of each pair to C and keeps the other halves.
3. C commits to the bit b by measuring all received qubits in basis $\{+,\times\}_{[b]}$. Let $x' \in \{0,1\}^n$ be the result.
4. To open the commitment, C sends b and x' to V.
5. V measures all his qubits in basis $\{+,\times\}_{[b]}$ and obtains $x \in \{0,1\}^n$. He chooses a random subset $I \subseteq \{1,\ldots,n\}$. V verifies that $x_i = x'_i$ for all $i \in I$ and accepts if and only if this is the case.

FIG. 5.2. *Protocol for EPR-based quantum commitment.*

LEMMA 5.2. COMM *is secure against dishonest* $\tilde{\mathsf{C}}$ *if and only if* EPR-COMM *is.*

Proof. The proof uses similar reasoning as the proof of Lemma 4.3. First, it clearly makes no difference if we change step 5 to the following:

5′. V chooses the subset I, measures all qubits with index in I in basis $\{+,\times\}_{[b]}$ and all qubits not in I in basis $\{+,\times\}_{[1-b]}$. V verifies that $x_i = x'_i$ for all $i \in I$ and accepts if and only if this is the case.

Finally, we can observe that the view of $\tilde{\mathsf{C}}$ does not change if V would have done his choice of I and his measurement already in step 1. Doing the measurements at this point means that the qubits to be sent to $\tilde{\mathsf{C}}$ collapse to a state that is distributed identically to the state prepared in the original scheme. The EPR-version is therefore equivalent to the original commitment scheme from $\tilde{\mathsf{C}}$'s point of view. □

5.2. Modeling dishonest committers. A dishonest committer $\tilde{\mathsf{C}}$ with bounded memory of at most γn qubits in EPR-COMM can be modeled very similarly to the dishonest OT-receiver $\tilde{\mathsf{R}}$ from section 4.3: $\tilde{\mathsf{C}}$ consists first of a circuit acting on all n qubits received, then of a measurement of all but at most γn qubits, and finally of a circuit that takes the following input: a bit b that $\tilde{\mathsf{C}}$ will attempt to open, the γn qubits in memory, and some ancilla in a fixed state. The output is a string $x' \in \{0,1\}^n$ to be sent to V at the opening stage.

Definition 5.3. *We define $\mathfrak{C}_\gamma$ to be the class of all committers $\{\tilde{\mathsf{C}}_n\}_{n>0}$ in* comm *or* epr-comm *that, at the start of the opening phase (i.e., at step 4), have a quantum memory of size at most γn qubits.*

We adopt the binding condition for quantum BC from [21].

Definition 5.4. *A (quantum) BC scheme is* (statistically) binding *against $\mathfrak{C}$ if for all $\{\tilde{\mathsf{C}}_n\}_{n>0} \in \mathfrak{C}$, the probability $p_b(n)$ that $\tilde{\mathsf{C}}_n$ opens $b \in \{0,1\}$ with success satisfies*

$$p_0(n) + p_1(n) \leq 1 + negl(n).$$

In the next section, we show that epr-comm is binding against $\mathfrak{C}_\gamma$ for any $\gamma < \frac{1}{2}$.

Note that the binding condition given here in Definition 5.4 is weaker than the classical one, where one would require that a bit b exists such that $p_b(n)$ is negligible. For a general quantum adversary who can always commit to 0 and 1 in superposition, however, this is too strong a requirement; thus, it is typically argued that Definition 5.4 is the best one can hope for. In upcoming work [16], though, we show that one *can* ask for a stronger binding property, and in fact protocol comm proposed here does satisfy a stronger binding property (but for a smaller bound on the committer's quantum memory). While the weaker condition is sufficient for many applications, the stronger one seems to be necessary in some cases. For instance, intuitively, comm can easily be transformed into a *string* commitment scheme simply by committing bitwise, but in order to prove this string commitment secure, it is necessary that comm is secure with respect to the stronger security definition. However, proving comm secure with respect to the stronger binding condition requires quite different techniques, and therefore we settle here for the weaker version and refer the interested reader to [16].

5.3. Security proof of the commitment scheme.

Note that the first three steps of epr-qot and epr-comm (i.e., before the memory bound applies) are exactly the same! This allows us to reuse Corollary 3.3 and the analysis of section 4.4 to prove the binding property of epr-comm.

Theorem 5.5. *For any $\gamma < \frac{1}{2}$,* comm *is perfectly hiding and statistically binding against $\mathfrak{C}_\gamma$.*

Proof. It remains to show that epr-comm is binding against $\mathfrak{C}_\gamma$. Let $\varepsilon, \delta > 0$ be such that $\gamma + 2h(\delta) + 2\varepsilon < 1/2$, where h is the binary entropy function. Recall that $\mathrm{B}^{\delta n} \leq 2^{h(\delta)n}$. Let R be the basis, determined by the bit that $\tilde{\mathsf{C}}$ claims in step 4, in which V measures the quantum state in step 5, and let X be the outcome. Corollary 3.3 implies the existence of an event $\mathcal{E}$ such that $P[\mathcal{E}|R{=}+] + P[\mathcal{E}|R{=}\times] \geq 1 - negl(n)$ and $\mathrm{H}_\infty(X|R{=}r, \mathcal{E}) \geq (\gamma + 2h(\delta) + 2\varepsilon)n$. Applying Lemma 2.2, it follows that any guess $\hat{X}$ for X satisfies

$$P\big[\hat{X} \in \mathrm{B}^{\delta n}(X) \,|\, R{=}r, \mathcal{E}\big] \leq 2^{-\frac{1}{2}(\mathrm{H}_\infty(X|X \in S^+) - \gamma n - 1) + \log(\mathrm{B}^{\delta n})} \leq 2^{-\varepsilon n + \frac{1}{2}}.$$

However, if $\hat{X} \notin \mathrm{B}^{\delta n}(X)$, then sampling a random subset of the positions will detect an error except with probability at most $2^{-\delta n}$. Hence, writing $q^+ := P[\mathcal{E}|R{=}+]$ and $q^\times := P[\mathcal{E}|R{=}\times]$,

$$p_0(n) \leq (1 - q^+) + q^+ \cdot (2^{-\varepsilon n + \frac{1}{2}} + 2^{-\delta n}) \leq 1 - q^+ + negl(n),$$

and analogously $p_1(n) \leq 1 - q^\times + negl(n)$. We conclude that

$$p_0(n) + p_1(n) \leq 2 - q^+ - q^\times + negl(n) \leq 1 + negl(n)\,. \qquad \square$$

5.4. Weakening the assumptions. As argued earlier, assuming that a party can produce single qubits (with probability 1) is not reasonable given current technology. Also the assumption that there is no noise on the quantum channel is impractical. It can be shown that a straightforward modification of COMM remains secure in the (ϕ, η)-weak quantum model as introduced in section 4.6 (see also section 6.1), with $\phi < \frac{1}{2}$ and $\eta < 1 - \phi$.

Let COMM$'$ be the modification of COMM where in step 5 V accepts if and only if $x_i = x'_i$ for all *but about a ϕ-fraction* of the i, where $r_i = \{+, \times\}_{[b]}$. More precisely, for all but a $(\phi + \varepsilon)$-fraction, where $\varepsilon > 0$ is sufficiently small.

THEOREM 5.6. *In the (ϕ, η)-weak quantum model,* COMM$'$ *is perfectly hiding and is binding against $\mathfrak{C}_\gamma$ for any γ satisfying $\gamma < \frac{1}{2}(1 - \eta) - 2h(\phi)$.*

Proof sketch. Using Bernstein's law of large numbers, one can argue that for *honest* C and V, the opening of a commitment is accepted except with negligible probability. The hiding property holds using the same reasoning as in Lemma 5.1. And the binding property can be argued essentially along the lines of Theorem 5.5, with the following modifications. Let J denote the set of indices i where V succeeds in sending a single qubit. We restrict the analysis to those i's which are in J. By Bernstein's law of large numbers, the cardinality of J is about $(1 - \eta)n$ (meaning within $(1 - \eta \pm \varepsilon)n$), except with negligible probability. Thus, restricting to these i's has the same effect as replacing γ by $\gamma/(1 - \eta)$ (neglecting the $\pm\varepsilon$ to simplify notation). Assuming that $\tilde{\mathsf{C}}$ knows every x_i for $i \notin J$, for all x_i's with $i \in J$ he has to be able to guess all but about a $\phi/(1 - \eta)$-fraction correctly, in order to be successful in the opening. However, $\tilde{\mathsf{C}}$ succeeds with only negligible probability if

$$\phi/(1 - \eta) < \delta \,.$$

Additionally, δ must be such that

$$\frac{\gamma}{1 - \eta} + 2h(\delta) < \frac{1}{2} \,.$$

δ can be chosen that way if

$$2\, h\!\left(\frac{\phi}{1 - \eta}\right) + \frac{\gamma}{1 - \eta} < \frac{1}{2} \,.$$

Using the fact that $h(\nu p) \leq \nu h(p)$ for any $\nu \geq 1$ and $0 \leq p \leq \frac{1}{2}$ such that $\nu p \leq 1$, this is clearly satisfied if $2h(\phi) + \gamma < \frac{1}{2}(1 - \eta)$. □

6. Towards practice.

In the following two sections, we elaborate on the question of how close to practice our systems are. First, we argue that imperfections occurring in practice like dark counts and empty pulses are covered by our (ϕ, η)-weak quantum model used in sections 4.6 and 5.6. Second, we sketch how our techniques can be extended to the more realistic setting of *noisy quantum memory.*

6.1. More imperfections.

In practice, quantum transmissions are subject to other imperfections: dark counts and empty pulses. Dark counts occur due to thermal fluctuation in the detector hardware which results in detection even though no qubit was received. Dark counts contribute to the error rate (i.e., each dark count accounts for a bit error with probability $\frac{1}{2}$) of the channel. This imperfection can therefore be included in the (ϕ, η)-weak quantum model by an appropriate choice of parameter ϕ without the need for any further modification.

Empty pulses occur in two cases: when the quantum channel lets a transmitted qubit escape (or when it is absorbed) and when the source does not produce any qubit for a given time slot. The latter is unavoidable for sources using weak coherent pulses as is the case in most experimental settings. Weak coherent pulses approximate a single-qubit source by producing on average only a small fraction of one qubit per pulse. It means that although most of the pulses are empty, the probability for a multiqubit pulse is very small. In this case, the receiver must report to the sender the positions of all pulses detected. Assuming the honest sender knows a tight upper bound on the rate at which the source produces empty pulses, the adversary can only take advantage of empty pulses caused by absorption in the fiber. The best the adversary can do is to substitute the fiber for one that preserves all qubits sent and to report empty pulses when a single pulse has been received. The effect is to increase the rate at which multiqubit pulses occur. This attack is known as the *photon number splitting attack* [7, 8, 26] in quantum key distribution applications. It follows that empty pulses can also be included in the (ϕ, η)-weak quantum model by an appropriate adjustment of parameter η.

Assume that a practical implementation of BB84-QOT or COMM takes place in a setting where ϕ_{X} is the probability for a bit error caused by the channel, ϕ_{DC} is the probability for a dark count, η_{MQ} is the probability for a multiqubit transmission, and η_{AB} is the probability for an empty pulse caused by absorption. These parameters are defined under the condition that the source is sending out a signal. It follows that if BB84-QOT and COMM are secure in the $(\phi_{\mathrm{X}} + \frac{\phi_{\mathrm{DC}}}{2}, \frac{\eta_{\mathrm{MQ}}}{1-\eta_{\mathrm{AB}}})$-weak quantum model, then their implementation is also secure provided that it is accurately modeled by these four parameters.

A variety of imperfections specific to particular implementations can be adapted to the weak quantum model in a similar way.

6.2. Generalizing the memory model. The bounded-quantum-storage model limits the number of physical qubits the adversary's memory can contain. A more realistic model would rather address the noise process which the adversary's memory undergoes. For instance, it is not hard to build a very large but unreliable memory device containing a large number of qubits. It is reasonable to expect that our protocols remain secure also in a scenario where the adversary's memory is of arbitrary size, but where some quantum operation (modeling noise) is applied to it. Inequality (2.2) of the privacy amplification theorem, Theorem 2.1, allows us to apply our constructions to slightly more general memory models. In particular, all our protocols that are secure against adversaries with memory of no more than γn qubits are also secure against any noise model that reduces the rank of the mixed state ρ_{E}, held by the adversary, to at most $2^{\gamma n}$.

An example of a noise process resulting in a reduction of $\mathrm{H}_0(\rho_{\mathsf{E}})$ is an erasure channel. Assuming the n initial qubits are each erased with probability larger than $1-\gamma$ when the memory bound applies, it holds except with negligible probability in n that $\mathrm{H}_0(\rho_{\mathsf{E}}) < \gamma n$. The same applies if the noise process is modeled by a depolarizing channel with error probability $p = 1 - \gamma$. Such a depolarizing channel replaces each qubit by a random one with probability p and does nothing with probability $1 - p$.

The technique we have developed does not allow us to deal with depolarizing channels with $p < 1-\gamma$ although one would expect that some $0 < p < 1-\gamma$ should be sufficient to ensure security against such adversaries. The reason for this is that not knowing the positions where the errors occurred should make it more difficult for the adversary than when the noise process is modeled by an erasure channel. However, it seems that our uncertainty relations (i.e., Theorems 3.1 and A.3) are not strong enough to address this case. Generalizing the bounded-quantum-storage model to more realistic noisy-memory models is an interesting open question.

7. Conclusion, further research, and open problems.

We have shown how to construct ROT and BC securely in the bounded-quantum-storage model. Our protocols require no quantum memory for honest players and remain secure provided the adversary has access to only quantum memory of size bounded by a large fraction of all qubits transmitted. Such a gap between the amount of storage required for honest players and adversaries is not achievable by classical means. All our protocols are noninteractive and can be implemented using current technology.

In this paper, we considered ROT of only one bit per invocation. Our technique can easily be extended to deal with string ROT, essentially by using a class of two-universal functions with range $\{0,1\}^{\ell n}$ rather than $\{0,1\}$, for some ℓ with $\gamma + \ell < \frac{1}{2}$ (resp., $< \frac{1-\eta}{4} - \frac{h(\phi)}{2}$ for BB84-QOT).

Although other flavors of OTs can be constructed from ROT using standard reductions, a more direct approach would give a better ratio between storage bound and communication complexity. Recent extensions have shown that a 1-2 OT protocol built along the lines of BB84-QOT is secure against adversaries with bounded quantum memory [16]. Interestingly, the techniques used are quite different from the ones of this paper (which appear to fail in the case of 1-2 OT), and they additionally allow us to analyze and prove secure the BC scheme COMM with respect to the stronger security definition, as briefly discussed in section 5.2.

A main open problem is the optimality of the bound on the adversary's quantum memory. The protocol QOT for instance, appears to be secure against any adversary with memory less than n qubits, but our analysis requires the memory to be smaller than $n/2$. Also, finding protocols secure against adversaries in more general noisy-memory models, briefly discussed in section 6.2, would certainly be a natural and interesting extension of this work to more practical settings.

Appendix A. Uncertainty relation for more mutually unbiased bases.

In this appendix, we generalize the uncertainty relations derived in section 3 to more than two mutually unbiased bases. Such uncertainty relations over more but not all mutually unbiased bases in terms of min-entropy may be of independent interest; see the discussion at the end of section 3.1.

First, we generalize Proposition 2.4 to more projectors.

PROPOSITION A.1. *For orthogonal projectors $A_0, A_1, A_2, \ldots, A_M$, it holds that*

$$\left\| \sum_{i=0}^{M} A_i \right\| \leq 1 + M \cdot \max_{0 \leq i < j \leq M} \|A_i A_j\|. \tag{A.1}$$

Proof. Defining

$$X := \begin{pmatrix} A_0 & A_1 & \cdots & A_M \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \quad \text{and} \quad Y := \begin{pmatrix} A_0 & 0 & \cdots & 0 \\ A_1 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ A_M & 0 & \cdots & 0 \end{pmatrix}$$

yields

$$XY = \begin{pmatrix} A_0 + A_1 + \ldots + A_M & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \quad \text{and}$$

$$YX = \begin{pmatrix} A_0 & A_0A_1 & \cdots & A_0A_M \\ A_1A_0 & A_1 & \cdots & A_1A_M \\ \vdots & \vdots & \ddots & \vdots \\ A_MA_0 & A_MA_1 & \cdots & A_M \end{pmatrix}.$$

The matrix YX can be additively decomposed into $M+1$ matrices according to the following pattern:

$$YX = \begin{pmatrix} * &&&& \\ & * &&& \\ && \ddots && \\ &&& * & \\ &&&& * \end{pmatrix} + \begin{pmatrix} 0 & * &&& \\ & 0 &&& \\ && \ddots & \ddots & \\ &&& 0 & * \\ * &&&& 0 \end{pmatrix} + \cdots + \begin{pmatrix} 0 &&&& * \\ * & 0 &&& \\ & \ddots & \ddots && \\ &&& 0 & \\ &&& * & 0 \end{pmatrix},$$

where the $*$'s stand for entries of YX and for $i = 0, \ldots, M$ the ith star pattern after the diagonal pattern is obtained by i cyclic shifts of the columns of the diagonal pattern.

As in the proof of Proposition 2.4, XY and YX are Hermitian, and we use Lemma 2.3, the triangle inequality, the unitary invariance of the operator norm, and the fact that for all $i \neq j : \|A_i\| = 1$ and $\|A_iA_j\| = \|A_jA_i\|$ to obtain the desired statement (A.1). □

Definition A.2. *Sets $\mathcal{B}^0, \mathcal{B}^1, \ldots, \mathcal{B}^M$ of bases of the complex Hilbert space $\mathbb{C}^{2^n}$ are called* mutually unbiased *if for all $i \neq j \in \{0, \ldots, M\}$, it holds that*

$$\forall |\varphi\rangle \in \mathcal{B}^i, \quad \forall |\psi\rangle \in \mathcal{B}^j : |\langle\varphi|\psi\rangle|^2 = 2^{-n}.$$

Theorem A.3. *Let the density matrix ρ describe the state of n qubits, and let $\mathcal{B}^0, \mathcal{B}^1, \ldots, \mathcal{B}^M$ be mutually unbiased bases of $\mathbb{C}^{2^n}$. Let $Q^0(\cdot), Q^1(\cdot), \ldots, Q^M(\cdot)$ be the distributions of the outcome when ρ is measured in bases $\mathcal{B}^0, \mathcal{B}^1, \ldots, \mathcal{B}^M$, respectively. Then, for any sets $L^0, L^1, \ldots, L^M \subset \{0,1\}^n$, it holds that*

$$\sum_{i=0}^{M} Q^i(L^i) \leq\ 1 + M \cdot 2^{-n/2} \max_{0 \leq i < j \leq M} \sqrt{|L^i||L^j|}.$$

Proof. The proof is analogous to that of Theorem 3.1. □

Analogous to Corollary 3.2, we derive an uncertainty relation about the sum of the min-entropies of up to $2^{n/2}$ distributions.

Corollary A.4. *For an $\varepsilon > 0$, let $0 < M < 2^{\frac{n}{2} - \varepsilon n}$. For $i = 0, \ldots, M$, let H_∞^i be the min-entropies of the distributions Q^i from the theorem above. Then,*

$$\sum_{i=0}^{M} \mathrm{H}_\infty^i \geq (M+1)\big(\log(M+1) - \mathit{negl}(n)\big).$$

Proof. For $i = 0, \ldots, M$, we denote by q^i_∞ the maximal probability of Q^i and let L^i be the set containing only the n-bit string x with this maximal probability q^i_∞. Theorem A.3 together with the assumption about M assures $\sum_{i=0}^M q^i_\infty \leq 1 + \mathit{negl}(n)$. By the inequality of the geometric and arithmetic mean it follows that

$$\begin{aligned}\sum_{i=0}^M \mathrm{H}^i_\infty &= -\log \prod_{i=0}^M q^i_\infty \geq -\log\left(\frac{1+\mathit{negl}(n)}{M+1}\right)^{M+1} \\ &= (M+1)\big(\log(M+1) - \mathit{negl}(n)\big). \qquad \square\end{aligned}$$

Acknowledgments. We would like to thank the anonymous referees for suggesting a different proof technique for our uncertainty relation and for making many useful comments. The authors are also grateful to Renato Renner for enlightening discussions and Charles H. Bennett for comments on earlier drafts.

REFERENCES

[1] M. A. Ballester, S. Wehner, and A. Winter, *State Discrimination with Post-Measurement Information*, http://arxiv.org/abs/quant-ph/0608014 (2006).

[2] C. H. Bennett and G. Brassard, *Quantum cryptography: Public key distribution and coin tossing*, in Proceedings of the IEEE International Conference on Computers, Systems, and Signal Processing, 1984, pp. 175–179.

[3] C. H. Bennett, G. Brassard, C. Crépeau, and U. M. Maurer, *Generalized privacy amplification*, IEEE Trans. Inform. Theory, 41 (1995), pp. 1915–1923.

[4] C. H. Bennett, G. Brassard, and J.-M. Robert, *Privacy amplification by public discussion*, SIAM J. Comput., 17 (1988), pp. 210–229.

[5] R. Bhatia, *Matrix Analysis*, Grad. Texts in Math. 169, Springer, Berlin, 1997.

[6] I. Bialynicki-Birula and J. Mycielski, *Uncertainty relations for information entropy in wave mechanics*, Comm. Math. Phys., 44 (1975), pp. 129–132.

[7] G. Brassard, N. Lütkenhaus, T. Mor, and B. C. Sanders, *Limitations on practical quantum cryptography*, Phys. Rev. Lett., 85 (2000), pp. 1330–1333.

[8] G. Brassard, N. Lütkenhaus, T. Mor, and B. C. Sanders, *Security aspects of practical quantum cryptography*, in Advances in Cryptology—EUROCRYPT '00, Lecture Notes in Comput. Sci. 1807, Springer, Berlin, 2000, pp. 289–299.

[9] G. Brassard and L. Salvail, *Secret-key reconciliation by public discussion*, in Advances in Cryptology—EUROCRYPT '93, Lecture Notes in Comput. Sci. 765, Springer, Berlin, 1993, pp. 410–423.

[10] C. Cachin, C. Crépeau, and J. Marcil, *Oblivious transfer with a memory-bounded receiver*, in Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 1998, pp. 493–502.

[11] J. L. Carter and M. N. Wegman, *Universal classes of hash functions*, in Proceedings of the 9th Annual ACM Symposium on Theory of Computing (STOC), 1977, pp. 106–112.

[12] C. Crépeau, *Efficient cryptographic protocols based on noisy channels*, in Advances in Cryptology—EUROCRYPT '97, Lecture Notes in Comput. Sci. 1233, Springer, Berlin, 1997, pp. 306–317.

[13] C. Crépeau and J. Kilian, *Achieving oblivious transfer using weakened security assumptions*, in Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 1988, pp. 42–53.

[14] I. Damgård, J. Kilian, and L. Salvail, *On the (im)possibility of basing oblivious transfer and bit commitment on weakened security assumptions*, in Advances in Cryptology—EUROCRYPT '99, Lecture Notes in Comput. Sci. 1592, Springer, Berlin, 1999, pp. 56–73.

[15] I. B. Damgård, S. Fehr, K. Morozov, and L. Salvail, *Unfair noisy channels and oblivious transfer*, in Theory of Cryptography Conference (TCC), Lecture Notes in Comput. Sci. 2951, Springer, Berlin, 2004, pp. 355–373.

[16] I. B. Damgård, S. Fehr, R. Renner, L. Salvail, and C. Schaffner, *A tight high-order entropic quantum uncertainty relation with applications*, in Advances in Cryptology—CRYPTO '07, Lecture Notes in Comput. Sci. 4622, Springer, Berlin, 2007, pp. 360–378.

[17] I. B. Damgård, S. Fehr, L. Salvail, and C. Schaffner, *Cryptography in the bounded quantum-storage model*, in Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2005, pp. 449–458.

[18] D. Deutsch, *Uncertainty in quantum measurements*, Phys. Rev. Lett., 50 (1983), pp. 631–633.

[19] Y. Z. Ding, D. Harnik, A. Rosen, and R. Shaltiel, *Constant-round oblivious transfer in the bounded storage model*, in Theory of Cryptography Conference (TCC), Lecture Notes in Comput. Sci. 2951, Springer, Berlin, 2004, pp. 446–472.

[20] Y. Dodis, L. Reyzin, and A. Smith, *Fuzzy extractors: How to generate strong keys from biometrics and other noisy data*, in Advances in Cryptology—EUROCRYPT '04, Lecture Notes in Comput. Sci. 3027, Springer, Berlin, 2004, pp. 523–540.

[21] P. Dumais, D. Mayers, and L. Salvail, *Perfectly concealing quantum bit commitment from any quantum one-way permutation*, in Advances in Cryptology—EUROCRYPT '00, Lecture Notes in Comput. Sci. 1807, Springer, Berlin, 2000, pp. 300–315.

[22] S. Dziembowski and U. M. Maurer, *On generating the initial key in the bounded-storage model*, in Advances in Cryptology—EUROCRYPT '04, Lecture Notes in Comput. Sci. 3027, Springer, Berlin, 2004, pp. 126–137.

[23] A. K. Ekert, *Quantum cryptography based on Bell's theorem*, Phys. Rev. Lett., 67 (1991), pp. 661–663.

[24] C. A. Fuchs and J. van de Graaf, *Cryptographic distinguishability measures for quantum-mechanical states*, IEEE Trans. Inform. Theory, 45 (1999), pp. 1216–1227.

[25] J. Hilgevood and J. B. M. Uffink, *The mathematical expression of the uncertainty principle*, in Microphysical Reality and Quantum Description, Kluwer Academic Publishers, Norwell, MA, 1988, pp. 91–114.

[26] B. Huttner, N. Imoto, N. Gisin, and T. Mor, *Quantum cryptography with coherent states*, Phys. Rev. A, 51 (1995), pp. 1863–1869.

[27] R. Impagliazzo, L. A. Levin, and M. Luby, *Pseudorandom generation from one-way functions*, in Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC), 1989, pp. 12–24.

[28] F. Kittaneh, *Norm inequalities for certain operator sums*, J. Funct. Anal., 143 (1997), pp. 337–348.

[29] K. Kraus, *Complementary observables and uncertainty relations*, Phys. Rev. D, 35 (1987), pp. 3070–3075.

[30] U. Larsen, *Superspace geometry: The exact uncertainty relationship between complementary aspects*, J. Phys. A, 23 (1990), pp. 1041–1061.

[31] H.-K. Lo and H. F. Chau, *Is quantum bit commitment really possible?*, Phys. Rev. Lett., 78 (1997), pp. 3410–3413.

[32] H. Maassen and J. B. M. Uffink, *Generalized entropic uncertainty relations*, Phys. Rev. Lett., 60 (1988), pp. 1103–1106.

[33] U. M. Maurer, *Perfect cryptographic security from partially independent channels*, in Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC), 1991, pp. 561–572.

[34] D. Mayers, *Unconditionally secure quantum bit commitment is impossible*, Phys. Rev. Lett., 78 (1997), pp. 3414–3417.

[35] T. Moran, R. Shaltiel, and A. Ta-Shma, *Non-interactive timestamping in the bounded storage model*, in Advances in Cryptology—CRYPTO '04, Lecture Notes in Comput. Sci. 3152, Springer, Berlin, 2004, pp. 460–476.

[36] R. Renner, *Security of Quantum Key Distribution*, Ph.D. thesis, ETH Zürich, Zürich, Switzerland, 2005; available online from http://arxiv.org/abs/quant-ph/0512258.

[37] R. Renner and R. König, *Universally composable privacy amplification against quantum adversaries*, in Theory of Cryptography Conference (TCC), Lecture Notes in Comput. Sci. 3378, Springer, Berlin, 2005, pp. 407–425.

[38] A. Rényi, *On measures of entropy and information*, in Proceedings of the 4th Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1, University of California Press, Berkeley, CA, 1961, pp. 547–561.

[39] L. Salvail, *Quantum bit commitment from a physical assumption*, in Advances in Cryptology—CRYPTO '98, Lecture Notes in Comput. Sci. 1462, Springer, Berlin, 1998, pp. 338–353.

[40] J. Sánchez-Ruiz, *Improved bounds in the entropic uncertainty and certainty relations for complementary observables*, Phys. Lett. A, 201 (1995), pp. 125–131.

[41] P. W. Shor and J. Preskill, *Simple proof of security of the BB84 quantum key distribution protocol*, Phys. Rev. Lett., 85 (2000), pp. 441–444.

[42] M. N. Wegman and J. L. Carter, *New classes and applications of hash functions*, in Proceedings of the 20th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 1979, pp. 175–182.

[43] S. Wehner and J. Wullschleger, *Composable Security in the Bounded-Quantum-Storage Model*, http://arxiv.org/abs/0709.0492 (2007).

[44] S. Wiesner, *Conjugate coding*, SIGACT News, 15 (1983), pp. 78–88.

SIAM J. COMPUT.
Vol. 37, No. 6, pp. 1891–1925

CONCURRENT NONMALLEABLE COMMITMENTS[*]

RAFAEL PASS[†] AND ALON ROSEN[‡]

Abstract. We present a nonmalleable commitment scheme that retains its security properties even when concurrently executed a polynomial number of times. That is, a man-in-the-middle adversary who is simultaneously participating in multiple concurrent *commitment phases* of our scheme, both as a sender and as a receiver, cannot make the values to which he commits depend on the values to which he receives commitments. Our result is achieved without assuming an a priori bound on the number of executions and without relying on any setup assumptions. Our construction relies on the existence of *standard* claw-free permutations and requires only a constant number of communication rounds.

Key words. cryptography, commitments, nonmalleability, concurrency, non-black-box simulation

AMS subject classification. 94A60

DOI. 10.1137/060661880

1. Introduction.

The notion of commitment is central in cryptographic protocol design. Often described as the "digital" analogue of sealed envelopes, commitment schemes enable a party, known as the *sender*, to commit itself to a value while keeping it secret from the *receiver*. This property is called *hiding*. Furthermore, the commitment is *binding*, and thus, in a later stage when the commitment is opened, it is guaranteed that the "opening" can yield only a single value determined in the committing stage.

For some applications, the above security guarantees are not sufficient, and additional properties are required. For instance, the definition of commitments does not rule out the possibility that an adversary, upon seeing a commitment to a specific value v, is able to commit to a related value (say, $v-1$), even though it does not know the actual value of v. This kind of attack might have devastating consequences if the underlying application relies on the *independence* of committed values (e.g., consider a case in which the commitment scheme is used for securely implementing a contract-bidding mechanism). The state of affairs is even worsened by the fact that many of the known commitment schemes are actually susceptible to this kind of attack.

1.1. Nonmalleable commitments.

In order to address the above concerns, Dolev, Dwork, and Naor (DDN) introduced the concept of *nonmalleable commitments* [16]. Loosely speaking, a commitment scheme is said to be nonmalleable if no adversary can succeed in the attack described above. That is, it is infeasible for the adversary to maul a commitment to a value v into a commitment to a "related" value $\tilde{v}$.

The first nonmalleable commitment protocol was constructed by Dolev, Dwork, and Naor [16]. The security of their protocol relies on the existence of one-way functions and requires $O(\log n)$ rounds of interaction, where $n \in N$ is a security parameter.

[*] Received by the editors June 7, 2006; accepted for publication (in revised form) December 12, 2007; published electronically March 26, 2008. A preliminary version of this paper appeared in *Proceedings of the* 46*th Annual IEEE Symposium on Foundations of Computer Science*, Pittsburgh, PA, 2005, pp. 563–572. Part of this work was done while the authors were at CSAIL, MIT.
http://www.siam.org/journals/sicomp/37-6/66188.html

[†] Department of Computer Science, Cornell University, Ithaca, NY 14853 (rafael@cs.cornell.edu).
[‡] DEAS, Harvard University, Cambridge, MA 02138 (alon@eecs.harvard.edu).

A more recent result by Barak presents a constant-round protocol for nonmalleable commitment, whose security relies on the existence of trapdoor permutations and hash functions that are collision-resistant against subexponential-sized circuits [2]. Even more recently, Pass and Rosen present a constant-round protocol for the same task, assuming only a collision-resistant hash function secure against polynomial-sized circuits [42].

1.2. Concurrent nonmalleable commitments. The basic definition of nonmalleable commitments considers only a scenario in which two executions take place at the same time. A natural extension of this scenario (already suggested in [16]) is one in which more than two invocations of the commitment protocol take place concurrently. In the concurrent scenario, the adversary is receiving commitments to multiple values $v_1, \ldots, v_m$ while attempting to commit to related values $\tilde{v}_1, \ldots, \tilde{v}_m$. As argued in [16], nonmalleability with respect to two executions can be shown to guarantee *individual* independence of any $\tilde{v}_i$ from any v_j. However, it does not rule out the possibility of an adversary to create *joint* dependencies between more than a single individual pair (see [16, section 3.4.1], for an example in the context of nonmalleable encryption). Resolving this issue has been stated as a major open problem in [16].

Partially addressing this issue, Pass demonstrates the existence of commitment schemes that remain nonmalleable under *bounded concurrent* composition [40]. That is, for any (predetermined) polynomial $p(\cdot)$, there exists a nonmalleable commitment that remains secure as long as it is not executed more than $p(n)$ times, where $n \in N$ is a security parameter.

One evident disadvantage of the above solution is that it requires that the number of executions be fixed *before* the protocol is specified; otherwise, no security guarantee is provided. Less evidently, the length of the messages in the protocols has to grow linearly with the number of executions. Thus, from both a theoretical and a practical point of view, the solution is still not satisfactory. What we would like to have is a *single* protocol that preserves its nonmalleability even when it is executed concurrently for *any* (not predetermined) polynomial number of times.

1.3. Our results. We present a new protocol for *concurrent nonmalleable* commitments. Our protocol remains nonmalleable even when concurrently executed an (unbounded) polynomial number of times. We do not rely on any kind of setup assumption (such as the existence of a common reference string).

The resulting commitment is *statistically binding* and satisfies nonmalleability *with respect to commitment*. The former condition implies that, except with negligible probability, a transcript of a commitment corresponds to a unique value, whereas the latter implies that, upon concurrently participating in polynomially many commitments, both as a receiver and as a sender, the adversary is not able to *commit* to a sequence of related values.[1] Here we assume that the adversary does not get to see the decommitment to any of the values to which he is receiving a commitment until he is done with committing to all of his values.

THEOREM 1.1 (concurrent nonmalleable commitment). *Suppose that there exists a family of pairs of claw-free permutations. Then there exists a constant-round statistically binding commitment scheme that is concurrently nonmalleable with respect*

[1] In a different variant, called nonmalleable commitment *with respect to opening* [19], the adversary is considered to have succeeded only if it manages to *decommit* to a related value. This paper considers only the notion of nonmalleability with respect to commitments.

to commitment.[2]

To the best of our knowledge, this result yields the first instance of a nontrivial protocol that simultaneously satisfies nonmalleability and unbounded concurrency without relying on set-up assumptions.

Additional contributions. Our proof also yields the first commitment scheme that satisfies nonmalleability by using a *strict* polynomial-time simulator (also known as strict nonmalleability) with respect to commitment.[3] By this we mean that the simulation used to prove nonmalleability runs in strict (as opposed to expected) polynomial time. This was the security notion originally defined (but not achieved in) the DDN paper [16].

Our definitions of nonmalleable commitments are somewhat different (stronger) than the ones appearing in the DDN paper [16]. Specifically, we formalize the notion of two values being unrelated through the concept of computational indistinguishability (rather than by using polynomial-time computable relations). The main reason for strengthening the definition is that it yields a notion that is more intuitive and easier to work with (especially in the concurrent setting). We stress that any protocol satisfying our definition also satisfies the original one.

Techniques and ideas. Our construction follows the paradigm introduced by Pass and Rosen, of using a protocol for nonmalleable zero knowledge in order to obtain (single execution) nonmalleable commitments [42] and relies on the "message-length" technique of Pass [40]. While our construction relies on the same high-level structure, the analysis of the protocol is significantly different. The central observation that enables the analysis is that concurrent simulation of the underlying (nonmalleable) zero-knowledge protocol is not actually necessary for proving the concurrent nonmalleability of our commitments. Indeed, for our analysis to go through, it will be sufficient to simulate only a *single* execution of the underlying zero-knowledge protocol. This will be performed while concurrently extracting *multiple* witnesses for the statements proved by the adversary. We call the above property *one-many simulation extractability.* We prove that this property is indeed satisfied by a variant of the nonmalleable zero-knowledge protocols of [40, 42]. To show this, we rely on a *non-black-box* simulation argument, which is delicately combined with a *black-box* extraction technique. (Here we use the fact that concurrent extraction is significantly easier than concurrent simulation (cf. [33]).)

1.4. Related work. A large body of previous work deals with the construction of nonmalleable protocols by assuming various kinds of trusted setup. Known constructions include nonmalleable commitment schemes assuming the existence of a common reference string [19, 11] as well as nonmalleable commitment schemes and noninteractive nonmalleable $\mathcal{ZK}$ protocols assuming the existence of a common random string [15, 14, 13].

Several of the above works explicitly address the issue of multiple executions of nonmalleable schemes [13, 11, 9] (also called *reusability* in the terminology of [11]). Perhaps most notable among the works addressing concurrency is the one on universally composable commitments [9]. Universal composability implies concurrent

[2] The existence of claw-free permutations follows from the assumption that factoring Blum integers is hard (or from the hardness of finding discrete logarithms modulo a prime). They are required for obtaining *perfectly* hiding commitments, as well as collision-resistant hashing.

[3] This should not be confused with a previous result showing the existence of commitment schemes that are strictly nonmalleable with respect to *opening* [42].

nonmalleability. However, it is impossible to construct universally composable commitments without making setup assumptions [9].

Other related works involve the task of session-key generation in a setting where the honest parties share a password that is taken from a relatively small dictionary [22, 39, 3]. These protocols are designed having a man-in-the-middle adversary in mind and require only the usage of a "mild" setup assumption (namely, the existence of a "short" password). Some of these works explicitly address the issue of multiple protocol execution (cf. [22]), but their treatment is limited to the case of sequential composition. A treatment of the full concurrent case appears in [31] (see also [10, 3]), but it relies on the existence of a common reference string.

2. Preliminaries.

2.1. Basic notation.

We let N denote the set of all integers. For any integer $m \in N$, denote by $[m]$ the set $\{1, 2, \ldots, m\}$. For any $x \in \{0,1\}^*$, we let $|x|$ denote the size of x (i.e., the number of bits used in order to write it). For two machines M, A, we let $M^A(x)$ denote the output of machine M on input x and given oracle access to A. The term *negligible* is used for denoting functions that are (asymptotically) smaller than one over any polynomial. More precisely, a function $\nu(\cdot)$ from nonnegative integers to reals is called negligible if, for every constant $c > 0$ and all sufficiently large n, it holds that $\nu(n) < n^{-c}$.

2.2. Witness relations.

We recall the definition of a witness relation for an $\mathcal{NP}$ language [20].

Definition 2.1 (witness relation). *A* witness relation *for a language $L \in \mathcal{NP}$ is a binary relation R_L that is polynomially bounded and polynomial-time recognizable and characterizes L by*

$$L = \{x : \exists y \, s.t. \, (x, y) \in R_L\}.$$

We say that y is a witness for the membership $x \in L$ if $(x, y) \in R_L$. We will also let $R_L(x)$ denote the set of witnesses for the membership $x \in L$, i.e.,

$$R_L(x) = \{y : (x, y) \in L\}.$$

In the following, we assume a fixed witness relation R_L for each language $L \in \mathcal{NP}$.

2.3. Computational indistinguishability and statistical closeness.

The following definition of (computational) indistinguishability originates in the seminal paper of Goldwasser and Micali [26].

Let X be a countable set of strings. A *probability ensemble indexed by X* is a sequence of random variables indexed by X. Namely, any $A = \{A_x\}_{x \in X}$ is a random variable indexed by X.

Definition 2.2 ((computational) indistinguishability). *Let X and Y be countable sets. Two ensembles $\{A_{x,y}\}_{x \in X, y \in Y}$ and $\{B_{x,y}\}_{x \in X, y \in Y}$ are said to be* computationally indistinguishable over X *if, for every probabilistic "distinguishing" machine D whose running time is polynomial in its first input, there exists a negligible function $\nu(\cdot)$ so that for every $x \in X$, $y \in Y$:*

$$|\Pr[D(x, y, A_{x,y}) = 1] - \Pr[D(x, y, B_{x,y}) = 1]| < \nu(|x|).$$

$\{A_{x,y}\}_{x \in X, y \in Y}$ and $\{B_{x,y}\}_{x \in X, y \in Y}$ are said to be statistically close *over X if the above condition holds for all (possibly unbounded) machines D.*

2.4. Interactive proofs, zero knowledge, and witness indistinguishability. We use the standard definitions of interactive proofs (and interactive Turing machines) [27, 20] and arguments [4]. Given a pair of interactive Turing machines P and V, we denote by $\langle P, V\rangle(x)$ the random variable representing the (local) output of V when interacting with machine P on common input x, when the random input to each machine is uniformly and independently chosen.

Definition 2.3 (interactive proof system). *A pair of interactive machines $\langle P, V\rangle$ is called an* interactive proof system *for a language L if machine V is polynomial-time and the following two conditions hold with respect to some negligible function $\nu(\cdot)$:*

- Completeness: *For every $x \in L$,*

$$\Pr[\langle P, V\rangle(x) = 1] \geq 1 - \nu(|x|).$$

- Soundness: *For every $x \notin L$, and every interactive machine B,*

$$\Pr[\langle B, V\rangle(x) = 1] \leq \nu(|x|).$$

In the case that the soundness condition is required to hold only with respect to a computationally bounded prover, the pair $\langle P, V\rangle$ is called an interactive argument *system.*

Zero knowledge. An interactive proof is said to be *zero knowledge* ($\mathcal{ZK}$) if it yields nothing beyond the validity of the assertion being proved. This is formalized by requiring that the view of every probabilistic polynomial-time adversary V^* interacting with the honest prover P can be simulated by a probabilistic polynomial-time machine S (the *simulator*). The idea behind this definition is that, whatever V^* might have learned from interacting with P, he could have actually learned by himself (by running the simulator S).

The notion of $\mathcal{ZK}$ was introduced by Goldwasser, Micali, and Rackoff [27]. To make $\mathcal{ZK}$ robust in the context of protocol composition, Goldreich and Oren [25] suggested to augment the definition so that the above requirement holds also with respect to all $z \in \{0,1\}^*$, where both V^* and S are allowed to obtain z as auxiliary input. The verifier's *view* of an interaction consists of the common input x, followed by its random tape and the sequence of prover messages the verifier receives during the interaction. We denote by $\text{view}_{V^*}^{P}(x, z)$ a random variable describing $V^*(z)$'s view of the interaction with P on common input x.

Definition 2.4 (zero knowledge). *Let $\langle P, V\rangle$ be an interactive proof system. We say that $\langle P, V\rangle$ is* zero knowledge *if, for every probabilistic polynomial-time interactive machine V^*, there exists a probabilistic polynomial-time algorithm S such that the ensembles $\{\text{view}_{V^*}^{P}(x, z)\}_{x\in L, z\in\{0,1\}^*}$ and $\{S(x, z)\}_{x\in L, z\in\{0,1\}^*}$ are computationally indistinguishable over L.*

A stronger variant of zero knowledge is one in which the output of the simulator is statistically close to the verifier's view of real interactions. We focus on *argument* systems, in which the soundness property is guaranteed to hold only with respect to polynomial-time provers.

Definition 2.5 (statistical zero knowledge). *Let $\langle P, V\rangle$ be an interactive argument system. We say that $\langle P, V\rangle$ is* statistical zero knowledge *if, for every probabilistic polynomial-time V^*, there exists a probabilistic polynomial-time S such that the ensembles $\{\text{view}_{V^*}^{P}(x, z)\}_{x\in L, z\in\{0,1\}^*}$ and $\{S(x, z)\}_{x\in L, z\in\{0,1\}^*}$ are statistically close over L.*

In the case that the ensembles $\{\text{view}_{V^*}^{P}(x, z)\}_{x\in L, z\in\{0,1\}^*}$ and $\{S(x, z)\}_{x\in L, z\in\{0,1\}^*}$ are identically distributed, the protocol $\langle P, V\rangle$ is said to be *perfect* zero knowledge.

Witness indistinguishability. An interactive proof is said to be *witness indistinguishable* ($\mathcal{WI}$) if the verifier's view is "computationally independent" of the witness used by the prover for proving the statement. In this context, we focus on languages $L \in \mathcal{NP}$ with a corresponding witness relation R_L. Namely, we consider interactions in which on common input x the prover is given a witness in $R_L(x)$. By saying that the view is computationally independent of the witness, we mean that, for any two possible $\mathcal{NP}$-witnesses that could be used by the prover to prove the statement $x \in L$, the corresponding views are computationally indistinguishable.

Let V^* be a probabilistic polynomial-time adversary interacting with the prover, and let $\text{view}^P_{V^*}(x, w, z)$ denote V^*'s view of an interaction in which the witness used by the prover is w (where the common input is x and V^*'s auxiliary input is z).

Definition 2.6 (witness indistinguishability). *Let $\langle P, V\rangle$ be an interactive proof system for a language $L \in \mathcal{NP}$. We say that $\langle P, V\rangle$ is* witness indistinguishable *for R_L if, for every probabilistic polynomial-time interactive machine V^* and for every two sequences $\{w^1_x\}_{x\in L}$ and $\{w^2_x\}_{x\in L}$, such that $w^1_x, w^2_x \in R_L(x)$ for every $x \in L$, the probability ensembles $\{\text{view}^P_{V^*}(x, w^1_x, z)\}_{x\in L,z\in\{0,1\}^*}$ and $\{\text{view}^P_{V^*}(x, w^2_x, z)\}_{x\in L,z\in\{0,1\}^*}$ are computationally indistinguishable over L.*

In the case that the ensembles $\{\text{view}^P_{V^*}(x, w^1_x, z)\}_{x\in L,z\in\{0,1\}^*}$ and $\{\text{view}^P_{V^*}(x, w^2_x, z)\}$ $_{x\in L,z\in\{0,1\}^*}$ are identically distributed, the proof system $\langle P, V\rangle$ is said to be witness *independent*.

2.5. Universal arguments. Universal arguments (introduced in [5] and closely related to the notion of computationally sound proofs [34]) are used in order to provide "efficient" proofs to statements of the form $y = (M, x, t)$, where y is considered to be a true statement if M is a nondeterministic machine that accepts x within t steps. The corresponding language and witness relation are denoted $L_{\mathcal{U}}$ and $R_{\mathcal{U}}$, respectively, where the pair $((M, x, t), w)$ is in $R_{\mathcal{U}}$ if M (viewed here as a two-input deterministic machine) accepts the pair (x, w) within t steps. Notice that every language in $\mathcal{NP}$ is linear-time reducible to $L_{\mathcal{U}}$. Thus, a proof system for $L_{\mathcal{U}}$ allows us to handle all $\mathcal{NP}$-statements. In fact, a proof system for $L_{\mathcal{U}}$ enables us to handle languages that are presumably "beyond" $\mathcal{NP}$, as the language $L_{\mathcal{U}}$ is $\mathcal{NE}$-complete (hence the name universal arguments).[4]

Definition 2.7 (universal argument). *A pair of interactive Turing machines (P, V) is called a* universal argument *system if it satisfies the following properties:*

- Efficient verification: *There exists a polynomial p such that, for any $y = (M, x, t)$, the total time spent by the (probabilistic) verifier strategy V, on common input y, is at most $p(|y|)$. In particular, all messages exchanged in the protocol have length smaller than $p(|y|)$.*
- Completeness by a relatively efficient prover: *For every $((M, x, t); w)$ in $R_{\mathcal{U}}$,*

$$\Pr[(P(w), V)(M, x, t) = 1] = 1.$$

Furthermore, there exists a polynomial q such that the total time spent by $P(w)$, on common input (M, x, t), is at most $q(T_M(x, w)) \le q(t)$, where $T_M(x, w)$ denotes the running time of M on input (x, w).
- Computational soundness: *For every polynomial-size circuit family $\{P^*_n\}_{n\in N}$, and every triplet $(M, x, t) \in \{0, 1\}^n \setminus L_{\mathcal{U}}$,*

$$\Pr[(P^*_n, V)(M; x; t) = 1] < \nu(n),$$

[4] Furthermore, every language in $\mathcal{NEXP}$ is polynomial-time (but not linear-time) reducible to $L_{\mathcal{U}}$.

where $\nu(\cdot)$ is a negligible function.

- Weak proof of knowledge: *For every positive polynomial p there exists a positive polynomial p' and a probabilistic polynomial-time oracle machine E such that the following holds: For every polynomial-size circuit family $\{P_n^*\}_{n\in N}$, and every sufficiently long $y = (M; x; t) \in \{0,1\}^*$, if $\Pr[(P_n^*; V)(y) = 1] > 1/p(|y|)$, then*

  $$\Pr[\exists w = w_1, \ldots w_t \in R_{\mathcal{U}}(y) \ \ s.t. \ \ \forall i \in [t], \ E_r^{P_n^*}(y; i) = w_i] > \frac{1}{p'(|y|)},$$

  where $R_{\mathcal{U}}(y) \overset{\text{def}}{=} \{w : (y, w) \in R_{\mathcal{U}}\}$ and $E_r^{P_n^*}(\cdot, \cdot)$ denotes the function defined by fixing the random tape of E to equal r and providing the resulting E_r with oracle access to P_n^*.

2.6. Commitment schemes. Commitment schemes are used to enable a party, known as the *sender*, to commit itself to a value while keeping it secret from the *receiver* (this property is called *hiding*). Furthermore, the commitment is *binding*, and thus, in a later stage when the commitment is opened, it is guaranteed that the "opening" can yield only a single value determined in the committing phase. Commitment schemes come in two different flavors: *statistically binding* and *statistically hiding*. We sketch the properties of each one of these flavors. Full definitions can be found in [20].

- *Statistically binding.* In statistically binding commitments, the binding property holds against unbounded adversaries, while the hiding property holds only against computationally bounded (nonuniform) adversaries. Loosely speaking, the statistical-binding property asserts that, with overwhelming probability over the coin tosses of the receiver, the transcript of the interaction fully determines the value committed to by the sender. The computational-hiding property guarantees that the commitments to any two different values are computationally indistinguishable.
- *Statistically hiding.* In statistically hiding commitments, the hiding property holds against unbounded adversaries, while the binding property holds only against computationally bounded (nonuniform) adversaries. Loosely speaking, the statistical-hiding property asserts that commitments to any two different values are statistically close (i.e., have negligible statistical distance). In the case where the statistical distance is 0, the commitments are said to be *perfectly hiding.* The computational-binding property guarantees that no polynomial-time machine is able to open a given commitment in two different ways.

Noninteractive perfectly binding commitment schemes can be constructed by using any 1–1 one-way function (see section 4.4.1 of [20]). By allowing some minimal interaction (in which the receiver first sends a single random initialization message), statistically binding commitment schemes can be obtained from any one-way function [35, 30]. We will think of such commitments as a *family* of noninteractive commitments, where the description of members in the family will be the initialization message. Statistically hiding commitment schemes can be constructed from any one-way function [36, 28], but constant-round schemes are only known to exist under stronger assumptions specifically, by assuming collision-resistant hash functions [12, 29]. Perfectly hiding commitment schemes, on the other hand, can be constructed from one-way permutations [37], but constant-round schemes are known only under stronger assumptions, such as the existence of a collection of certified claw-free permutations [21].

2.7. Proofs of knowledge. Informally an interactive proof is a proof of knowledge if the prover convinces the verifier not only of the validity of a statement but also that it possesses a witness for the statement. This notion is formalized by the introduction of a machine E, called a *knowledge extractor*. As the name suggests, the extractor E is supposed to extract a witness from any malicious prover P^* that succeeds in convincing an honest verifier. More formally, we have the following.

Definition 2.8 (proof of knowledge). *Let (P, V) be an interactive proof system for the language L with witness relation R_L. We say that (P, V) is a proof of knowledge if there exists a polynomial q and a probabilistic oracle machine E such that, for every probabilistic polynomial-time interactive machine P^*, there exists some negligible function $\mu(\cdot)$ such that for every $x \in L$ and every $y, r \in \{0,1\}^*$ such that $\Pr[\langle P^*_{x,y,r}, V(x)\rangle = 1] > 0$, where $P^*_{x,y,r}$ denotes the machine P^* with the common input fixed to x, the auxiliary input fixed to y, and the random tape fixed to r, the following hold:*

1. *The expected number of steps taken by $E^{P^*_{x,y,r}}$ is bounded by*

$$\frac{q(|x|)}{\Pr[\langle P^*_{x,y,r}, V(x)\rangle = 1]},$$

where $E^{P^*_{x,y,r}}$ denotes the machine E with oracle access to $P^*_{x,y,r}$.

2. *Furthermore,*

$$\Pr[\langle P^*_{x,y,r}, V(x)\rangle = 1 \land E^{P^*_{x,y,r}} \notin R_L(x)] \leq \mu(|x|).$$

The machine E is called a (knowledge) extractor.

We remark that, as our definition considers only computationally bounded provers, we get only a "computationally convincing" notion of a proof of knowledge (*arguments of knowledge*) [4]. We additionally point out that our definition is slightly different from the definition of [6] in that we require that the expected running time of the extractor is always bounded by poly$(|x|)/p$, where p denotes the success probability of P^*, whereas [6] allows for some additional slackness in the running time. On the other hand, whereas [6] requires the extractor to always output a valid witness, we instead allow the extractor to fail with some negligible probability. We will rely on the following theorem.

Theorem 2.9 (see [7, 4]). *Assume the existence of claw-free permutations. Then there exists a constant-round public-coin witness-independent argument of knowledge for $\mathcal{NP}$.*

Indeed, standard techniques can be used to show that the parallelized version of the protocol of [7], using perfectly hiding commitments, is an argument of knowledge (as defined above). As usual, the knowledge extractor E proceeds by feeding new "challenges" to the prover P^* until it gets two accepting transcripts. If the two accepting challenges contain the same challenge, or if the prover manages to open up a commitment in two different ways, the extractor outputs *fail*; otherwise, it can extract a witness.

3. Concurrent nonmalleable commitments.

Nonmalleable commitments were introduced by DDN [16]. Our definitions of nonmalleability are somewhat stronger than the ones proposed by DDN [16]. Specifically, we formalize the notion of two values being unrelated through the concept of computational indistinguishability (rather than by using polynomial-time computable relations).

3.1. The general setting. Let $\langle C, R\rangle$ be a commitment scheme, and consider a *man-in-the-middle* adversary A that is simultaneously participating in multiple concurrent executions of $\langle C, R\rangle$. Executions in which A is playing the role of the receiver are said to belong to the *left* interaction, whereas executions in which A is playing the role of the sender are said to belong to the *right* interaction. We assume for simplicity, and without loss of generality, that the number of commitment schemes taking place in the left and right interactions is identical. The total number of the interactions in which the adversary is involved (either as a sender or as a receiver) is not a priori bounded by any polynomial (though it is assumed to be polynomial in the security parameter). We assume that the adversary does not get to see the decommitment to any of the values to which he is receiving a commitment until he is done with committing to all of his values.

Besides controlling the messages that it sends in the left and right interactions, A has control over their scheduling. In particular, it may delay the transmission of a message in one interaction until it receives a message (or even multiple messages) in the other interaction. It can also arbitrarily interleave messages that belong to different executions within an interaction.

The adversary A is trying to take advantage of his participation in the commitments taking place in the left interaction in order to commit to a related value in the right interaction. The honest sender and receiver are not necessarily aware of the existence of the adversary and might be under the impression that they are interacting one with the other. We let $v_1, \ldots, v_m$ denote the values committed to in the left interaction and $\tilde{v}_1, \ldots, \tilde{v}_m$ denote the values committed to in the right interaction.[5] The above scenario is depicted in Figure 1 (with no explicit demonstration of possible interleavings of messages between different executions).

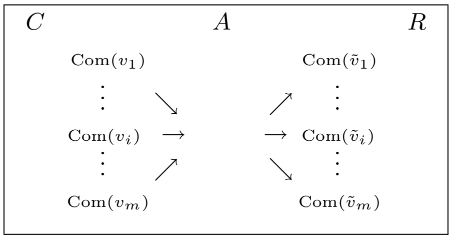

C
A
R
$\mathrm{Com}(v_1)$
$\mathrm{Com}(v_i)$
$\mathrm{Com}(v_m)$
$\mathrm{Com}(\tilde{v}_1)$
$\mathrm{Com}(\tilde{v}_i)$
$\mathrm{Com}(\tilde{v}_m)$

FIG. 1. *A concurrent man-in-the-middle adversary.*

The traditional definition of nonmalleable commitments [16] considers the case when $m = 1$. Loosely speaking, it requires that the left interaction does not "help" the adversary A in committing to a value $\tilde{v}_1$ that is somehow correlated with the value v_1. In this work we focus on nonmalleability with respect to commitment [16], where the adversary is said to succeed if it manages to *commit* to a related value (even without being able to later decommit to this value). Note that this notion makes sense only in the case of statistically binding commitments.

3.2. Nonmalleability via indistinguishability. Following the simulation paradigm [26, 27, 23, 24], the notion of nonmalleability is formalized by comparing between a *man-in-the-middle* and a *simulated* execution. In the man-in-the-middle execution the adversary is simultaneously acting as a receiver in one interaction and

[5] The adversary does not necessarily "know" the values $\tilde{v}_1, \ldots, \tilde{v}_m$. However, since the commitment is statistically binding, this value is (almost always) well defined.

as a committer in another interaction. In the simulated execution the adversary is engaged in a single interaction where it is acting as a committer.

The original definition of nonmalleability required that, for any polynomial-time computable (nonreflexive) relation $\mathcal{R}$, the value $\tilde{v}$ committed to by the adversary in the simulated execution is no (significantly) less likely to satisfy $\mathcal{R}(v, \tilde{v}) = 1$ than the value committed to by the adversary in the man-in-the-middle execution [16].

To facilitate the formalization for $m > 1$, we choose to adopt a slightly different definitional approach and will actually require an even stronger condition (which we are still able to satisfy with our protocol). Specifically, we require that, for any adversary in a man-in-the-middle execution, there exists an adversary that commits to essentially the same value in the simulated execution. By essentially the same value, we mean that the value committed to by the simulator is computationally indistinguishable from the value committed to by the adversary in the man-in-the-middle execution.

Since copying cannot be ruled out, we will be interested only in the case where copying is not considered success. We therefore impose the condition that, whenever the adversary has *fully* copied a transcript of an interaction in which it acts as a receiver, the value $\tilde{v}$ that he has committed to in the corresponding execution is set to be a special "failure" symbol, denoted by $\perp$.

3.3. The actual definition. Let $\langle C, R\rangle$ be a commitment scheme, and let $n \in N$ be a security parameter. Consider man-in-the-middle adversaries that are participating in left and right interactions in which $m = \text{poly}(n)$ commitments take place. We compare between a *man-in-the-middle* and a *simulated* execution.

The man-in-the-middle execution. In the man-in-the-middle execution, the adversary A is simultaneously participating in m left and right interactions. In the left interactions the man-in-the-middle adversary A interacts with C by receiving commitments to values $v_1, \dots, v_m$. In the right interaction A interacts with R by attempting to commit to a sequence of related values $\tilde{v}_1, \dots, \tilde{v}_m$. Prior to the interaction, the values $v_1, \dots, v_m$ are given to C as local input. A receives an auxiliary input z, which may contain a priori information about $v_1, \dots, v_m$. Let $\mathsf{mim}^A_{\mathsf{com}}(v_1, \dots, v_m, z)$ denote a random variable that describes the values $\tilde{v}_1, \dots, \tilde{v}_m$ to which the adversary has committed in the right interaction. (Since we are dealing with statistically binding commitments, $\tilde{v}_1, \dots, \tilde{v}_m$ are (almost always) well defined. Whenever the value of the commitment is not uniquely defined (which can happen with some negligible probability in the case of statistically binding commitments), the value of the commitment is defined to be $\perp$.) If the transcript of the ith right commitment is identical to the transcript of *any* of the left interactions (which means that the adversary has fully copied a specific commitment that has taken place on the left), the value $\tilde{v}_i$ is set to be $\perp$.[6]

The simulated execution. In the simulated execution, a simulator S directly interacts with R. As in the man-in-the-middle execution, the values $v_1, \dots, v_m$ are chosen prior to the interaction, and S receives some a priori information about $v_1, \dots, v_m$ as part of its auxiliary input z. We let $\mathsf{sim}^S_{\mathsf{com}}(z)$ denote a random variable that describes the values committed to in the output of S (which consists of a sequence of values $\tilde{v}_1, \dots, \tilde{v}_m$).

[6] This approach allows $\tilde{v}_i = v_j$ for some $i, j \in \{1, \dots, m\}$, as long as the man-in-the-middle adversary does not fully copy the messages from one of the left executions. This is in contrast to the original definition which does not handle the case of $\tilde{v}_i = v_j$ (as $\mathcal{R}$ is nonreflexive). This means that the new approach takes into consideration a potentially larger class of attacks.

Definition 3.1. *A commitment scheme $\langle C, R\rangle$ is said to be* concurrent nonmalleable with respect to commitment *if, for every polynomial $p(\cdot)$, and every probabilistic polynomial-time man-in-the-middle adversary A that participates in at most $m = p(n)$ concurrent executions, there exists a probabilistic polynomial-time simulator S such that the following ensembles are computationally indistinguishable over $\{0,1\}^*$:*

- $\left\{\mathsf{mim}^A_{\mathsf{com}}(v_1,\ldots,v_m,z)\right\}_{v_1,\ldots,v_m\in\{0,1\}^n,n\in N,z\in\{0,1\}^*}$,
- $\left\{\mathsf{sim}^S_{\mathsf{com}}(z)\right\}_{v_1,\ldots,v_m\in\{0,1\}^n,n\in N,z\in\{0,1\}^*}$.

It can be seen that for $m = 1$ any protocol that satisfies Definition 3.1 also satisfies the original (relation-based) definition of nonmalleability of [16]. Loosely speaking, this is because the existence of a polynomial-time computable relation $\mathcal{R}$ that violates the original definition of nonmalleability could be used to distinguish between the values of $\mathsf{mim}^A_{\mathsf{com}}(v, z)$ and $\mathsf{sim}^S_{\mathsf{com}}(z)$. We additionally point out that, even when considering only the case when $m = 1$, Definition 3.1 is stronger than the relation-based definition in that it prevents an adversary from producing a different commitment to the same value to which it receives a commitment; the original definition did not consider this as a successful attack.[7]

3.4. One-many concurrent nonmalleable commitments. A seemingly more relaxed (and thus potentially easier to satisfy) notion of concurrent nonmalleable commitments is one in which the man-in-the-middle adversary A engages in only a *single* commitment protocol in the left interaction (but still polynomially many in the right interaction). Such a notion is a special case of Definition 3.1 in which the adversary A participates in only one commitment session on the left-hand side (instead of m sessions).

A commitment protocol that satisfies the relaxed definition is said to be *one-many concurrent nonmalleable*. As we argue below, the relaxed notion "in essence" implies full-fledged nonmalleability. In particular, in order to construct concurrent nonmalleable commitments, it will be sufficient to come up with a protocol that is one-many concurrent nonmalleable. To formalize this composability property we need to restrict our attention to certain "natural" commitment schemes.[8] We say that a commitment scheme is *natural* if any commitment where the committer aborts (sending $\perp$) before the end of the protocols makes the commitment invalid (i.e., a commitment to $\perp$).

Proposition 3.2. *Let $\langle C, R\rangle$ be a natural one-many concurrent nonmalleable commitment. Then $\langle C, R\rangle$ is also a (full-fledged) concurrent nonmalleable commitment.*

Proof. Let A be a man-in-the-middle adversary that participates in at most $m = p(n)$ concurrent executions. We show the existence of a simulator S such that the following ensembles are computationally indistinguishable over $\{0,1\}^*$:

- $\left\{\mathsf{mim}^A_{\mathsf{com}}(v_1,\ldots,v_m,z)\right\}_{v_1,\ldots,v_m\in\{0,1\}^n,n\in N,z\in\{0,1\}^*}$,
- $\left\{\mathsf{sim}^S_{\mathsf{com}}(z)\right\}_{v_1,\ldots,v_m\in\{0,1\}^n,n\in N,z\in\{0,1\}^*}$.

The simulator S proceeds as follows on input z. S incorporates $A(z)$ and internally emulates all of the left interactions for A by simply *honestly* committing to the string 0^n (i.e., in order to emulate the ith left interaction, S executes the algorithm C on input 0^n). Messages from the right interactions are instead forwarded externally,

[7] In contrast, the definition of nonmalleable *encryption* of [16] indeed also prevents this type of attack. In a sense, our definition is thus more in line with their definition of nonmalleability for encryptions schemes.

[8] Similar technical restrictions are needed to show composability of nonmalleable encryption [43].

with the following exception: Whenever A wishes to send the last message q_i in the ith right session, S "holds on" to it without (yet) forwarding it externally. Finally, when S has completed the emulation of all left interactions for A, it checks whether A fully copied any of the left executions. For each execution i where A fully copied one of the left executions, S externally sends $\perp$ as its last message in the ith right execution (to invalidate that commitment); for all other executions j, A instead sends the final message q_j.

We show that the values to which S commits are indistinguishable from the values to which A commits. Suppose, for contradiction, that this is not the case. That is, there exists a polynomial-time distinguisher D and a polynomial $p(n)$ such that, for infinitely many n, there exist strings $v_1, \ldots, v_m \in \{0,1\}^n, z \in \{0,1\}^*$ such that

$$\Pr\Big[D(\mathsf{mim}^A_{\mathsf{com}}(v_1, \ldots, v_m, z)) = 1\Big] - \Pr\Big[D(\mathsf{sim}^S_{\mathsf{com}}(z)) = 1\Big] \geq \frac{1}{p(n)}.$$

Fix a generic n for which this happens. We provide a hybrid argument that will contradict the one-many nonmalleability of $\langle C, R\rangle$. The "hybrid" random variable $\mathsf{hyb}_k(v_1, \ldots, v_m, z)$ involves an execution where $A(z)$ is participating in m left and m right interactions and is defined in the following way:

- For $j \leq k$, the jth session in the left interaction consists of a commitment to 0^n.
- For $j > k$, the jth session in the left interaction consists of a commitment to v_j.
- Output the values $\tilde{v}_1, \ldots, \tilde{v}_m$ committed to by A in the right interactions with an honest R. As in the definition of $\mathsf{mim}_{\mathsf{com}}$ (see Definition 3.1), the value $\tilde{v}_i$ of a commitment c_i is set to $\perp$ if A fully copies one of the left executions.

Note that the values $\tilde{v}_1, \ldots, \tilde{v}_m$ are not efficiently computable but are well defined nevertheless. Just as in Definition 3.1, if a commitment can be opened to two (or more) different values, we set its value to $\perp$. By construction, it directly follows that

$$\mathsf{hyb}_0(v_1, \ldots, v_m, z) = \mathsf{mim}^A_{\mathsf{com}}(v_1, \ldots, v_m, z).$$

By additionally relying on the naturality property of $\langle C, R\rangle$, it holds that[9]

$$\mathsf{hyb}_m(v_1, \ldots, v_m, z) = \mathsf{sim}^S_{\mathsf{com}}(z).$$

It follows by a standard hybrid argument that there exists an $i \in [m]$ such that

$$\Pr\Big[D(\mathsf{hyb}_{i-1}(v_1, \ldots, v_m, z)) = 1\Big] - \Pr\Big[D(\mathsf{hyb}_i(v_1, \ldots, v_m, z)) = 1\Big] \geq \frac{1}{p(n)m}.$$

Note that the only difference between the experiments $\mathsf{hyb}_{i-1}(v_1, \ldots, v_m, z)$ and $\mathsf{hyb}_i(v_1, \ldots, v_m, z)$ is that in the former A receives a commitment to v_i in session i, whereas in the latter it receives a commitment to 0^n. Now consider the one-many adversary $\tilde{A}$ that when receiving $\tilde{z} = (i, v_1, \ldots, v_m, z)$ as auxiliary input proceeds as follows. $\tilde{A}$ internally incorporates $A(z)$ and emulates the left and right interactions for A.

1. $\tilde{A}$ forwards messages in its jth right execution directly to and from A (as part of its jth right execution) with the following exception: Whenever A wishes to send the last message q_i in the ith right execution, $\tilde{A}$ holds on to it without forwarding it externally.

[9] The naturality property together with the construction of S is used to make sure that S produces a commitment to $\perp$ whenever A copies the left interaction.

2. $\tilde{A}$ forwards messages from its left session directly to and from A (as part of its ith session).
3. $\tilde{A}$ emulates all left sessions $j \neq i$, by committing to v_j if $j > i$ and committing to 0^n otherwise.
4. Whenever $\tilde{A}$ has completed the emulation of all left executions, it checks whether A fully copied any of the left executions; for each such copied execution i, $\tilde{A}$ sends $\perp$ as its last message in the ith right execution and otherwise sends q_i.

It follows directly from the construction and the natural property of $\langle C, R\rangle$ that

$$\mathsf{mim}^{\tilde{A}}_{\mathsf{com}}(v_i, \tilde{z}) = \mathsf{hyb}_{i-1}(v_1, \ldots, v_m, z),$$
$$\mathsf{mim}^{\tilde{A}}_{\mathsf{com}}(0, \tilde{z}) = \mathsf{hyb}_{i}(v_1, \ldots, v_m, z).$$

This contradicts the fact that there exists a simulator $\tilde{S}$ for $\tilde{A}$ such that both

1. $\mathsf{mim}^{\tilde{A}}_{\mathsf{com}}(v_i, \tilde{z})$ and $\mathsf{sim}^{\tilde{S}}_{\mathsf{com}}(\tilde{z})$ are indistinguishable and
2. $\mathsf{mim}^{\tilde{A}}_{\mathsf{com}}(0, \tilde{z})$ and $\mathsf{sim}^{\tilde{S}}_{\mathsf{com}}(\tilde{z})$ are indistinguishable.

We conclude that $\langle C, R\rangle$ is not one-many concurrent nonmalleable. □

4. The protocol.

Our construction of concurrent nonmalleable commitments follows the paradigm introduced by Pass and Rosen for obtaining (single execution) nonmalleable commitments [42]. The commit phase of the Pass–Rosen protocol consists of having the sender engage in a (standard) statistically binding commitment with the receiver and thereafter also provide a *nonmalleable* $\mathcal{ZK}$ proof of knowledge of the value committed to. The reveal phase consists of sending the decommitment information of the statistically binding commitment used in the commit phase.

The basic scenario in which nonmalleable $\mathcal{ZK}$ protocols take place involves a man-in-the-middle adversary A (see Figure 2) that is simultaneously participating in two executions of the protocol. These executions are called the *left* and the *right* interactions.

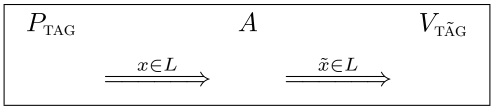


FIG. 2. *The man-in-the-middle adversary.*

The left interaction is tagged by an identity string TAG $\in \{0,1\}^n$, and the right interaction is tagged by an identity $\tilde{\text{TAG}} \in \{0,1\}^n$. The instructions of the protocol executed in each of the interactions depend on the corresponding identities TAG and $\tilde{\text{TAG}}$. (The way in which the identity strings are determined and used will become clear at a later stage.) We let $\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle$ denote a protocol execution in which the identity TAG is used as a tag.

In the left interaction, the adversary A is verifying the validity of a statement x by interacting with an honest prover P_{TAG} using a protocol $\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle$. In the right interaction, A proves the validity of a statement $\tilde{x}$ to the honest verifier $V_{\tilde{\text{TAG}}}$. The statement $\tilde{x}$ proved in the right interaction is chosen by A, possibly depending on the messages it receives in the left interaction. As in the case of concurrent nonmalleable commitments, A has control over the scheduling of the messages.

Loosely speaking, a protocol is nonmalleable if, for any such man-in-the-middle adversary, there exists a "stand-alone" prover that convinces the verifier in the right interaction with essentially the same success probability as the adversary does in a man-in-the-middle execution. See [42] and section 4.1 for further details.

4.1. Nonmalleability and simulation extractability. In [42] it is shown that a commitment is nonmalleable provided that the underlying $\mathcal{ZK}$ protocols satisfy a *simulation extractability* property (a strengthening of nonmalleability). Loosely speaking, simulation extractability requires that, for any man-in-the-middle adversary A, there exists a simulator extractor that can simulate both the left and the right interactions for A while outputting a witness for the statement proved by the adversary in the right interaction.

For the purpose of the current work we will need to show that the $\mathcal{ZK}$ protocols used in the compilation satisfy an even stronger property, which we call *one-many simulation extractability*. This is a strengthening of the simulation extractability property in that it guarantees simulation and extraction (of all witnesses on the right) even if there is an *unbounded* number of concurrent *right* interactions (but still with only one left interaction).

As we will show later, a (noninteractive) commitment scheme that is compiled with one-many simulation extractable $\mathcal{ZK}$ will result in a one-many concurrent nonmalleable commitment protocol $\langle C, R\rangle$. By Proposition 3.2, this implies that $\langle C, R\rangle$ is also concurrent nonmalleable.

Let A be a man-in-the-middle adversary that is simultaneously participating in one left interaction of $\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle$ while acting as verifier and an (unbounded) polynomial number of right interactions of $\langle P_{\tilde{\text{TAG}}_i}, V_{\tilde{\text{TAG}}_i}\rangle_{i=1}^m$ while acting as prover.

Let $\mathsf{view}_A(x, z, \text{TAG})$ denote the *joint* view of $A(x, z)$ and the honest verifier $V_{\tilde{\text{TAG}}}$ when A is verifying a left proof of the statement x, by using identity TAG, and proving in the right interactions statements of its choice by using tags of its choice. (The view consists of the messages sent and received by A in both left and right interactions, the random coins of A, and $V_{\tilde{\text{TAG}}}$).[10] Given a function $t = t(n)$ we use the notation $\{\cdot\}_{x,z,\text{TAG}}$ as shorthand for $\{\cdot\}_{x\in L, z\in\{0,1\}^*, \text{TAG}\in\{0,1\}^{t(|x|)}}$.

DEFINITION 4.1 (one-many simulation extractability). *A family* $\{\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle\}_{\text{TAG}\in\{0,1\}^*}$ *of interactive proofs for L is said to be* simulation extractable *with tags of length $t = t(n)$ if, for any polynomial $p(\cdot)$ and any man-in-the-middle adversary A that participates in one left interaction and at most $m = p(n)$ right interactions, there exists a probabilistic expected poly-time machine S such that:*

1. *The probability ensembles* $\{S_1(x, z, \text{TAG})\}_{x,z,\text{TAG}}$ *and* $\{\mathsf{view}_A(x, z, \text{TAG})\}_{x,z,\text{TAG}}$ *are statistically close over L, where $S_1(x, z, \text{TAG})$ denotes the first output of $S(x, z, \text{TAG})$.*
2. *Let $x \in L$, $z \in \{0,1\}^*$, and* $\text{TAG} \in \{0,1\}^{t(|x|)}$, *and let* $(\mathsf{view}, \bar{w})$ *denote the output of $S(x, z, \text{TAG})$ (on input of some random tape). Let $\tilde{x}_1 \ldots, \tilde{x}_m$ be the right-execution statements appearing in* view, *and let* $\tilde{\text{TAG}}_1 \ldots \tilde{\text{TAG}}_m$ *denote the correspoding right-execution tags. Then, for any $i \in [m]$ such that the ith right execution in* view *is accepting* and $\text{TAG} \neq \tilde{\text{TAG}}_i$, *$\bar{w}$ contains a witness w_i so that $R_L(\tilde{x}_i, w_i) = 1$.*

4.2. A simulation extractable protocol. We now describe our construction of simulation extractable protocols. At a high level, the construction proceeds in two steps:

[10] Since the messages sent by A are fully determined given the code of A and the messages it receives, including them as part of the view is somewhat redundant. The reason we have chosen to do so is for convenience of presentation.

1. For any $n \in N$, construct a family $\{\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle\}_{\mathsf{tag}\in[2n]}$ of simulation extractable arguments with tags of length $t(n) = \log n + 1$.
2. For any $n \in N$, use the family $\{\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle\}_{\mathsf{tag}\in[2n]}$ to construct a family $\{\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle\}_{\text{TAG}\in\{0,1\}^n}$ of simulation extractable arguments with tags of length $t(n) = n$.

The construction of the family $\{\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle\}_{\mathsf{tag}\in[2n]}$ relies on Barak's non-black-box techniques for obtaining constant-round public-coin $\mathcal{ZK}$ for $\mathcal{NP}$ [1] and are very similar in structure to the $\mathcal{ZK}$ protocols used by Pass in [40]. Overall, the construction of $\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle$ is essentially identical to the construction of the simulation extractable protocols in [42]; the only difference is that in the current paper we will replace statistically hiding commitments with *perfectly* hiding commitments and statistically witness indistinguishable arguments with witness-*independent* arguments.

Let $n \in N$, and let $T : N \to N$ be a function that satisfies $T(n) = n^{\omega(1)}$. Our construction relies on a "special" $\mathbf{NTIME}(T(n))$ relation, which we denote by $\mathbf{R}_{\mathsf{sim}}$. It also makes use of a "special-purpose" universal argument ($UARG$) [18, 17, 32, 34, 5, 42]. Let $\{\mathcal{H}_n\}_n$ be a family of hash functions where a function $h \in \mathcal{H}_n$ maps $\{0,1\}^*$ to $\{0,1\}^n$, and let $\mathbf{Com}$ be a perfectly hiding commitment scheme for strings of length n, where, for any $\alpha \in \{0,1\}^n$, the length of $\mathbf{Com}(\alpha)$ is upper bounded by $2n$. The relation $\mathbf{R}_{\mathsf{sim}}$ is described in Figure 3.

Instance: A triplet $\langle h, c, r\rangle \in \mathcal{H}_n \times \{0,1\}^n \times \{0,1\}^{\text{poly}(n)}$.

Witness: A program $\Pi \in \{0,1\}^*$, a string $y \in \{0,1\}^*$, and a string $s \in \{0,1\}^{\text{poly}(n)}$.

Relation: $\mathbf{R}_{\mathsf{sim}}(\langle h, c, r\rangle, \langle \Pi, y, s\rangle) = 1$ if and only if:

1. $|y| \le |r| - n$.
2. $c = \mathbf{Com}(h(\Pi); s)$.
3. $\Pi(y) = r$ within $T(n)$ steps.

FIG. 3. *The relation* $\mathbf{R}_{\mathsf{sim}}$.

Remark 1 (simplifying assumption). For simplicity of exposition, we view $\mathbf{Com}$ as a one-message perfectly hiding commitment scheme (even though such commitments cannot exist). In reality, $\mathbf{Com}$ would be taken to be a two-message commitment scheme (which may be based on collections of claw-free permutations [21]).

The construction of our protocol employs a universal argument that is specially tailored for our purposes (a variant of which has already appeared in [42]). The main distinguishing features of this universal argument, which we call the *special-purpose* argument, are (1) it is witness *independent* and (2) it will enable us to prove that our protocols satisfy the proof of knowledge property of Definition 2.8.[11] Let $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}}\rangle$ be a witness-independent argument of knowledge, and let $\langle P_{\mathsf{UA}}, V_{\mathsf{UA}}\rangle$ be a four-message, public-coin universal argument where the length of the messages is upper bounded by n.[12] The special-purpose $UARG$, which we denote by $\langle P_{\mathsf{sUA}}, V_{\mathsf{sUA}}\rangle$, handles statements of the form $(x, \langle h, c_1, c_2, r_1, r_2\rangle)$, where the triplets $\langle h, c_1, r_1\rangle$ and

[11] The "weak" proof of knowledge property of a universal argument (as defined in [5]) is not sufficient for our purposes. Specifically, while in a weak proof of knowledge it is required that the extractor succeeds with a probability that is polynomially related to the success probability of the prover, in our proof of security we will make use of an extractor that succeeds with a probability negligibly close to the success probability of the prover.

[12] Both witness-independent arguments of knowledge and four-message, public-coin, universal arguments can be constructed by assuming a family $\mathcal{H}_n$ of claw-free permutations (cf. [20, 32, 34, 5]).

$\langle h, c_2, r_2 \rangle$ correspond to instances for $\mathbf{R}_{\mathsf{sim}}$. The protocol $\langle P_{\mathsf{sUA}}, V_{\mathsf{sUA}} \rangle$ is described in Figure 4.

Parameters: Security parameter 1^n.

Common input: $x \in \{0,1\}^n$, $\langle h, c_1, c_2, r_1, r_2 \rangle$, where, for $i \in \{1,2\}$, $\langle h, c_i, r_i \rangle$ is an instance for $\mathbf{R}_{\mathsf{sim}}$.

Stage 1 (encrypted UARG):

$V \rightarrow P$: Send $\alpha \xleftarrow{\text{R}} \{0,1\}^n$.
$P \rightarrow V$: Send $\widehat{\beta} = \mathbf{Com}(0^n)$.
$V \rightarrow P$: Send $\gamma \xleftarrow{\text{R}} \{0,1\}^n$.
$P \rightarrow V$: Send $\widehat{\delta} = \mathbf{Com}(0^n)$.

Stage 2 (body of the proof):

$P \leftrightarrow V$: A witness-independent argument of knowledge $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}} \rangle$ proving the OR of the following statements:

1. $\exists\, w \in \{0,1\}^{\mathrm{poly}(|x|)}$ so that $R_L(x, w) = 1$.
2. $\exists\, \langle \beta, \delta, s_1, s_2 \rangle$ so that:
   - $\widehat{\beta} = \mathbf{Com}(\beta; s_1)$.
   - $\widehat{\delta} = \mathbf{Com}(\delta; s_2)$.
   - $(\alpha, \beta, \gamma, \delta)$ is an accepting transcript for $\langle P_{\mathsf{UA}}, V_{\mathsf{UA}} \rangle$ proving the statement:
     – $\exists\, \langle i, \Pi, y, s \rangle$ so that $\mathbf{R}_{\mathsf{sim}}(\langle h, c_i, r_i \rangle, \langle \Pi, y, s \rangle) = 1$.

FIG. 4. *A special-purpose universal argument* $\langle P_{\mathsf{sUA}}, V_{\mathsf{sUA}} \rangle$.

4.2.1. A family of $2n$ protocols. We next present a family of protocols $\{\langle P_{\mathsf{tag}}, V_{\mathsf{tag}} \rangle\}_{\mathsf{tag} \in [2n]}$ (with tags of length $t(n) = \log n + 1$).[13] These protocols are a "two-slot" version of Barak's $\mathcal{ZK}$ arguments [1]. (The idea of using a multiple-slot version of Barak's protocol already appeared in [41, 40], and the "message-length" technique appeared in [40].) There are two aspects in which the protocols presented here differ from the protocol of [40]: The new protocols satisfy (1) a *perfect* secrecy property (note that this is also different from [42], where secrecy is statistical) and (2) a *proof of knowledge* property.

Let $\mathbf{Com}$ be a perfectly hiding commitment scheme for strings of length n, where, for any $\alpha \in \{0,1\}^n$, the length of $\mathbf{Com}(\alpha)$ is upper bounded by $2n$. Let $\mathbf{R}_{\mathsf{sim}}$ be the perfect variant of the relation R_{sim}, and let $\langle P_{\mathsf{sUA}}, V_{\mathsf{sUA}} \rangle$ be the special-purpose universal argument. Protocol $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}} \rangle$ is described in Figure 5.

Note that the only difference between two protocols $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}} \rangle$ and $\langle P_{\tilde{\mathsf{tag}}}, V_{\tilde{\mathsf{tag}}} \rangle$ is the length of the verifier's "next messages": In fact, the length of those messages in $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}} \rangle$ is a parameter that depends on tag (as well as on the length parameter $\ell(n)$). This property will be crucial for the analysis of these protocols in the man-in-the-middle setting.

4.2.2. A family of 2^n protocols. By relying on the protocol family $\{\langle P_{\mathsf{tag}}, V_{\mathsf{tag}} \rangle\}_{\mathsf{tag} \in [2n]}$, we now show how to construct a family $\{\langle P_{\text{TAG}}, V_{\text{TAG}} \rangle\}_{\text{TAG} \in \{0,1\}^n}$ with tags of length $t(n) = n$. The protocols are *constant-round* and involve n parallel executions of $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}} \rangle$, with appropriately chosen tags. This new family of protocols

[13] A closer look at the construction will reveal that it will in fact work for any $t(n) = O(\log n)$. The choice of $t(n) = \log n + 1$ is simply made for the sake of concreteness (as in our constructions it is the case that $\mathsf{tag} \in [2n]$).

Common input: An instance $x \in \{0,1\}^n$.

Parameters: Security parameter 1^n, length parameter $\ell(n)$.

Tag string: $\mathsf{tag} \in [2n]$.

Stage 0 (setup):
$V \to P$: Send $h \overset{\mathrm{R}}{\leftarrow} \mathcal{H}_n$.

Stage 1 (slot 1):
$P \to V$: Send $c_1 = \mathbf{Com}(0^n)$.
$V \to P$: Send $r_1 \overset{\mathrm{R}}{\leftarrow} \{0,1\}^{\mathsf{tag}\cdot\ell(n)}$.

Stage 1 (slot 2):
$P \to V$: Send $c_2 = \mathbf{Com}(0^n)$.
$V \to P$: Send $r_2 \overset{\mathrm{R}}{\leftarrow} \{0,1\}^{(2n+1-\mathsf{tag})\cdot\ell(n)}$.

Stage 2 (body of the proof):
$P \Leftrightarrow V$: A special-purpose *UARG* $\langle P_{\mathsf{sUA}}, V_{\mathsf{sUA}}\rangle$ proving the OR of the following statements:
1. $\exists\, w \in \{0,1\}^{\mathrm{poly}(|x|)}$ s.t. $R_L(x,w) = 1$.
2. $\exists\, \langle \Pi, y, s\rangle$ s.t. $\mathbf{R}_{\mathsf{Sim}}(\langle h, c_1, r_1\rangle, \langle \Pi, y, s\rangle) = 1$.
3. $\exists\, \langle \Pi, y, s\rangle$ s.t. $\mathbf{R}_{\mathsf{Sim}}(\langle h, c_2, r_2\rangle, \langle \Pi, y, s\rangle) = 1$.

FIG. 5. *Protocol* $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle$.

Common input: An instance $x \in \{0,1\}^n$.

Parameters: Security parameter 1^n, length parameter $\ell(n)$.

Tag string: $\text{TAG} \in \{0,1\}^n$. Let $\text{TAG} = \text{TAG}_1, \dots, \text{TAG}_n$.

The protocol:
$P \leftrightarrow V$: For all $i \in \{1, \dots, n\}$ (in parallel):
1. Set $\mathsf{tag}_i = (i, \text{TAG}_i)$.
2. Run $\langle P_{\mathsf{tag}_i}, V_{\mathsf{tag}_i}\rangle$ with common input x and length parameter $\ell(n)$.

V: Accept if and only if all runs are accepting.

FIG. 6. *Protocol* $\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle$.

is denoted $\{\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle\}_{\text{TAG}\in\{0,1\}^n}$ and is described in Figure 6.

Notice that $\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle$ has a constant number of rounds (since each $\langle P_{\mathsf{tag}_i}, V_{\mathsf{tag}_i}\rangle$ is constant-round). Also notice that for $i \in [n]$ the length of $\mathsf{tag}_i = (i, \text{TAG}_i)$ is

$$|i| + |\text{TAG}_i| = \log n + 1 = \log 2n.$$

By viewing (i, TAG_i) as elements in $[2n]$ we infer that the length of verifier messages in $\langle P_{\mathsf{tag}_i}, V_{\mathsf{tag}_i}\rangle$ is upper bounded by $2n\ell(n)$. Hence, as long as $\ell(n) = \mathrm{poly}(n)$, the length of verifier messages in $\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle$ is $2n^2\ell(n) = \mathrm{poly}(n)$.

As shown in [42], for any $\text{TAG} \in 2^n$, the protocol $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle$ is an interactive argument. In fact, what they show is a stronger statement, namely, that the protocols $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle$ are arguments of knowledge (as in Definition 2.8). What [42] actually shows is how to prove the above by assuming a family of hash functions that is collision-resistant against $T(n) = n^{\omega(1)}$-sized circuits. As mentioned in [42], by slightly modifying $\mathbf{R}_{\mathsf{sim}}$, one can prove the same statement under the more standard assumption of collision resistance against polynomial-sized circuits (cf. [5]).

PROPOSITION 4.2 (argument of knowledge [42]). *Let $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}}\rangle$ and $\langle P_{\mathsf{UA}}, V_{\mathsf{UA}}\rangle$ be the protocols used in the construction of $\langle P_{\mathsf{sUA}}, V_{\mathsf{sUA}}\rangle$. Suppose that $\{\mathcal{H}_n\}_n$ is collision-resistant for $T(n)$-sized circuits, that* **Com** *is perfectly hiding, that $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}}\rangle$ is a witness-independent argument of knowledge, and that $\langle P_{\mathsf{UA}}, V_{\mathsf{UA}}\rangle$ is a universal argument. Then, for any* TAG $\in \{0,1\}^n$, *$\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle$ is a perfect zero-knowledge argument of knowledge.*

The main technical contribution of the current paper consists of proving that the protocol $\{\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle\}_{\text{TAG}\in\{0,1\}^n}$ is one-many simulation extractable.

LEMMA 4.3 (main technical lemma). *Let $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}}\rangle$ and $\langle P_{\mathsf{UA}}, V_{\mathsf{UA}}\rangle$ be the protocols used in the construction of $\langle P_{\mathsf{sUA}}, V_{\mathsf{sUA}}\rangle$. Suppose that $\{\mathcal{H}_n\}_n$ is collision-resistant for $T(n)$-sized circuits, that* **Com** *is perfectly hiding, that $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}}\rangle$ is a witness-independent argument of knowledge, that $\langle P_{\mathsf{UA}}, V_{\mathsf{UA}}\rangle$ is a universal argument, and that $\ell(n) \geq 2n^3 + n$. Then $\{\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle\}_{\text{TAG}\in\{0,1\}^n}$ is one-many simulation extractable with tags of length $t(n) = n$.*

Before we go on and prove Lemma 4.3, we turn to describe our commitment protocol and to show how one-many simulation extractability is used in order to establish its one-many concurrent nonmalleability. The full proof of Lemma 4.3 can be found in section 5.

4.3. The commitment protocol.

By using protocols from $\{\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle\}_{\text{TAG}\in\{0,1\}^n}$ as a subroutine, we present the construction of concurrent nonmalleable commitments. Let $\{\text{Com}_r\}_{r\in\{0,1\}^*}$ be a family of *noninteractive* statistically binding commitment schemes (e.g., Naor's commitment [35]). Let $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ be a one-time signature scheme secure against a chosen-message attack. Consider the protocol in Figure 7 (which is a variant of the nonmalleable commitment of Pass and Rosen [42]).[14]

Security parameter: 1^n.

String to be committed to: $v \in \{0,1\}^n$.

Commit phase:

$R \to C$: Send a uniformly chosen $r \in \{0,1\}^n$.
$C \to R$: Let $vk, sk \leftarrow \mathsf{Gen}(1^n)$. Pick uniformly $s \in \{0,1\}^n$.
Set TAG $= vk$ and send $c = \text{Com}_r(v; s)$, TAG.
$C \leftrightarrow R$: Prove using $\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle$ that there exist $v, s \in \{0,1\}^n$ so that $c = \text{Com}_r(v; s)$.
$C \to R$: Let T denote the transcript of the above interaction.
Compute $\sigma = \mathsf{Sign}(sk, T)$ and send σ.
R: Verify that $\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle$ is accepting and that $\mathsf{Verify}(vk, T, \sigma) = 1$.

Reveal phase:

$C \to R$: Send v and s.
R: Verify that $c = \text{Com}_r(v; s)$.

FIG. 7. *Concurrent nonmalleable commitment $\langle C, R\rangle$.*

As argued in [42], the statistical binding property of $\langle C, R\rangle$ follows directly from the statistical binding of Com. The computational hiding property follows from the computational hiding of Com, as well as from the (stand-alone) $\mathcal{ZK}$ property of $\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle$. Hence, we have the following.

[14] The difference between this protocol and the protocol of [42] is that here we also employ a signature scheme. We note that the important difference, nevertheless, lies in the analysis of the protocol.

PROPOSITION 4.4 (see [42]). *Suppose that* $\{\mathrm{Com}_r\}_{r\in\{0,1\}^*}$ *is a family of* noninteractive *statistically binding commitment schemes and that all members in the family* $\{\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle\}_{\text{TAG}\in\{0,1\}^n}$ *are (stand-alone) zero knowledge. Then* $\langle C, R\rangle$ *is a statistically binding commitment protocol.*

4.4. Concurrent nonmalleability. By relying on the one-many simulation extractability of $\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle$, we next argue that $\langle C, R\rangle$ is one-many nonmalleable.

THEOREM 4.5. *Suppose that* $\{\mathrm{Com}_r\}_{r\in\{0,1\}^*}$ *is a family of noninteractive statistically binding commitment schemes and that* $\{\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle\}_{\text{TAG}\in\{0,1\}^n}$ *is one-many simulation extractable and natural. Then* $\langle C, R\rangle$ *is a natural one-many concurrent nonmalleable commitment.*

Proof. We start by noting that it follows directly from the fact that the last message of $\langle C, R\rangle$ is supposed to be a signature on the transcript (in case the commitment is valid) and that any commitment where the committer aborts before the last round is invalid—in other words, $\langle C, R\rangle$ is natural.

Next, consider a man-in-the-middle adversary A that participates in one left execution and $m = m(n)$ right executions. We assume without loss of generality that A is deterministic (since A can obtain its "best" random tape as auxiliary input). Consider the simulator S that proceeds as follows on input z. S incorporates $A(z)$ and internally emulates the left interactions for A by simply *honestly* committing to the string 0^n (i.e., S executes the algorithm C on input 0^n). Messages from the right interactions are instead forwarded externally, with the following exception: Whenever A wishes to send the last message q_i in the ith right session, S "holds on" to it without (yet) forwarding it externally. Finally, when S has completed the emulation of the left interactions for A, it checks whether A fully copied any of the left executions. For each execution i where A fully copied any of the left executions, S externally sends $\perp$ as its last message in the ith right execution (to invalidate that commitment); for all other executions j, $\tilde{A}$ instead sends the final message q_j. We show that the following distributions are indistinguishable over $\{0,1\}^*$:

- $\left\{\mathsf{mim}^{A}_{\mathsf{com}}(v, z)\right\}_{v\in\{0,1\}^n, n\in N, z\in\{0,1\}^*}$,
- $\left\{\mathsf{sim}^{S}_{\mathsf{com}}(z)\right\}_{v\in\{0,1\}^n, n\in N, z\in\{0,1\}^*}$.

Suppose, for contradiction, that this is not the case. That is, there exists a polynomial-time distinguisher D and a polynomial $p(n)$ such that, for infinitely many n, there exist strings $v \in \{0,1\}^n$, $z \in \{0,1\}^*$ such that

$$\Pr\left[D(\mathsf{mim}^{A}_{\mathsf{com}}(v, z)) = 1\right] - \Pr\left[D(\mathsf{sim}^{S}_{\mathsf{com}}(z)) = 1\right] \geq \frac{1}{p(n)}.$$

Fix a generic n for which this happens. We show how this contradicts the simulation extractability property of $\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle$. We start by providing an (oversimplified) sketch. On a high level, the proof consists of the following steps:

1. We first note that, since the commit phase of (C, R) "essentially" consists only of a statement (c, r) (i.e., the commitment) and a proof of the validity of (c, r), A can be interpreted as a one-many simulation extractability adversary A' for $\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle$.
2. It follows from the simulation extractability property of $\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle$ that there exists a combined simulator-extractor S' for A' that outputs a view that is statistically close to that of A' while at the same time outputting witnesses to all accepting right proofs.

3. Since the view output by the simulator-extractor S' is *statistically* close to the view of A' in the real interaction, it follows that also the *values* committed to in that view are statistically close to the values committed to by A'. (Note that computational indistinguishability would not have been enough to argue the indistinguishability of these values, since they are not efficiently computable from the view.)
4. It also follows that, except with negligible probability, the simulator-extractor S' will output also the witnesses to all accepting right executions.[15] We conclude that S' additionally outputs the values *committed to* in the right executions.
5. We finally note that if D can distinguish between the values committed to by A and by S, then D can also distinguish the second output (which consists of the committed values) of S' when run on input a commitment (using Com) to v and the second output of S' when run on input a commitment to 0. This contradicts the hiding property of Com.

We proceed to a formal proof. One particular complication that arises with the above proof sketch is that in the construction of $\langle C, R\rangle$ we are relying on the use of a family of commitment schemes $\{\text{Com}_r\}_{r\in\{0,1\}^*}$ and not a single noninteractive commitment scheme. To address this issue we make use of nonuniformity to show the existence of a particular "prefix" of right interactions such that A always chooses the instance Com_r in its left interaction, yet A commits to different values when receiving a commitment to v (as in mim) and 0 (as in sim).

More precisely, since in both experiments mim and sim are identical up until the point where A sends its first message in the left interaction, there must exist some *fixed* prefix transcript τ of A's right interactions (see Figure 8) such that

1. A sends its first message r_τ in its *left* interaction directly after receiving the messages in τ (as part of its right executions);
2. D distinguishes between $\mathsf{mim}^A_{\mathsf{com}}(v,z)$ and $\mathsf{sim}^S_{\mathsf{com}}(z)$ with probability $1/p(n)$, conditioned on the event that the right executions are consistent with τ.

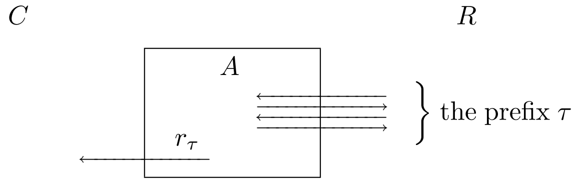


FIG. 8. *The prefix τ before r_τ is sent.*

Note that, since the receiver R sends the first message in the protocol, τ contains all first messages r sent by the honest R in all m right executions. Let $r_\tau^{(1)},..,r_\tau^{(m)}$ denote these first messages. Let $\texttt{committed}_\tau \subseteq [m]$ denote the set of executions i such that $A(z)$ has sent its first message in the ith execution in τ. (Recall that the first message sent by A, playing the "role" of C in execution i, consists of a commitment using $\text{Com}_{r_\tau^{(i)}}$.) For each $i \in \texttt{committed}_\tau$, let $\texttt{value}_\tau(i)$ denote the value committed to in the first message sent by $A(z)$ in execution i in τ. (If this value is not uniquely defined, set $\texttt{value}_\tau(i) = \perp$.) Additionally, for each $i \in \texttt{committed}_\tau$, let $\texttt{coins}_\tau(i)$ be

[15] More precisely, the simulator-extractor outputs only witnesses to all right executions that use a different tag than the left interaction. We rely on the use of the digital signature to handle the case when A copies the tag of the left interaction.

a random tape for R such that D distinguishes between $\mathsf{mim}^A_{\mathsf{com}}(v, z)$ and $\mathsf{sim}^S_{\mathsf{com}}(z)$ with probability $\frac{1}{p(n)}$, conditioned on the event that the right executions are consistent with τ and that, in each right execution $i \in \mathtt{committed}_\tau$, R uses the random tape $\mathtt{coins}_\tau(i)$. (It follows by an averaging argument that such random tapes must exist.)

Given the partial transcript τ we next proceed in the following three steps:

1. We first define a simulation extractability adversary A'.
2. We next show that A' can be used to violate the nonmalleability property of Com.
3. In the final step, we show how to use the simulator-extractor S' for A' to violate the hiding property of Com.

Step 1: *Defining a simulation extractability adversary* A'. We define a one-many simulation extractability adversary A' for $\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle$. On a high level A' internally incorporates A on input the transcript τ and externally forwards all messages that are part of $\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle$ while internally handling all other messages. Additionally, A' internally emulates (without externally forwarding) all messages that are part of executions $i \in \mathtt{committed}_\tau$ (note that messages in these executions cannot be externally forwarded as the execution of protocol $\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle$ could potentially have already begun).

We proceed to a formal description of A'. To simplify notation we assume that the machine A' has the values $\tau, \mathtt{committed}_\tau, \mathtt{coins}_\tau$ hard-coded in its description (as it can always receive them as part of its auxiliary input z'). On input x', TAG', z' (i.e., $A'(z')$ expects to receive a proof of the statement x' using tag TAG'), where $x' = (c, r_\tau)$, and $z' = (z, sk)$ such that sk is the corresponding secret key for the signature verification key TAG', A' then internally incorporates $A(z)$ and emulates the left and right interactions for A in the following manner:

1. It starts by feeding A all messages in τ as part of its right executions.
2. For each $i \in \mathtt{committed}_\tau$, A' internally emulates the (rest of the) ith right execution for A by honestly following the strategy of R using the random tape $\mathtt{coins}_\tau(i)$.
3. For each $i \in [m]$ such that $i \notin \mathtt{committed}_\tau$, A' externally forwards messages in the ith right interaction, as follows. Whenever A sends its first message c_i, vk_i, in the ith execution, A externally forwards $(c_i, r^{(i)}_\tau)$ as its statement and vk_i as its tag. Thereafter, it externally forwards all of the messages from $\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle$. (Note that, since $i \notin \mathtt{committed}_\tau$, A has not yet sent any messages in execution i; thus, A is expected to produce a proof of the statement $(c', r^{(i)}_\tau)$ using TAG' as a tag.)
4. Messages in A's left interaction are instead forwarded externally as part of A''s left interaction. Once A has sent its first message r_τ, A' starts by feeding it c, TAG' and next externally forwards all remaining messages (i.e., all of the messages that are part of $\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle$). Once the execution of $\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle$ has concluded, A' signs the transcript of the left interaction by using sk (received as auxiliary input) and feeds the signature to A. (Recall that whereas A' receives only a proof as part of its left interaction, A expects to see a commitment using $\langle C, R\rangle$. It is therefore essential that A' adds a signature to the proof.)

Step 2: *Show that* A' *violates nonmalleability of* Com. Define the following experiment $\mathsf{hyb}_1(v')$:

1. Pick $vk, sk \leftarrow \mathsf{Gen}(1^n)$, and pick uniformly $s \in \{0, 1\}^n$.
2. Let $c = \text{Com}_{r_\tau}(v', s)$.

3. Let $x' = (c, r_\tau)$, $\text{TAG}' = vk$, $z' = (z, sk)$.
4. Emulate an execution for $A'(x', \text{TAG}', z')$ by honestly providing a proof of x' (using tag TAG' and the witness (v', s)) as part of its left interaction and honestly verifying all right interactions.
5. Finally, given the view of A' in the above emulation, reconstruct the view of A in the emulation by A'. Output the pair $(\mathsf{view}, \bar{v})$, where view denotes the reconstructed view of A and $\bar{v}$ denotes the values committed to in the view of A. (As in Definition 3.1, if a commitment is undefined or invalid, or if the transcript of the commitment is identical to the transcript of the commitment received by A on the left, its value is set to $\perp$.) Note that, although the values committed to are not necessarily efficiently computable from the view of A, they are determined.

Note that $\mathsf{hyb}_1(\cdot)$ is not efficiently samplable, since the last step in the description of hyb_1 is not efficient. However, except for that last step, every other operation in hyb_1 is indeed efficient. (This will be useful to us at a later stage.)

Claim 4.6.

$$\Pr\Big[D(\mathsf{hyb}_1(v)) = 1\Big] - \Pr\Big[D(\mathsf{hyb}_1(0^n)) = 1\Big] \geq \frac{1}{p(n)}.$$

Proof. By construction of A' and hyb_1, it directly follows that the view of A in $\mathsf{hyb}_1(v)$ is identically distributed to the view of A in $\mathsf{mim}^A_{\mathsf{com}}(v, z)$, conditioned on the event that the right executions (in the view of A) are consistent with τ and $\mathtt{coins}_\tau$. Additionally, by relying on the natural property of $\langle C, R\rangle$ (which guarantees that S outputs a commitment to $\perp$ whenever A fully copies any of the left executions), it holds that the view of A in $\mathsf{hyb}_1(0^n)$ is identically distributed to the view of A in $\mathsf{sim}^S_{\mathsf{com}}(z)$, conditioned on the event that the right executions (in the view of A) are consistent with τ and $\mathtt{coins}_\tau$. Since the outputs of hyb_1, mim, and sim are computed by applying the same function to the view of A, in the corresponding experiments, it follows that

1. the output of $\mathsf{hyb}_1(v)$ is identically distributed to the output of $\mathsf{mim}^A_{\mathsf{com}}(v, z)$, conditioned on the event that the right executions (in the view of A) are consistent with τ and $\mathtt{coins}_\tau$;
2. the output of $\mathsf{hyb}_1(0^n)$ is identically distributed to the output of $\mathsf{sim}^S_{\mathsf{com}}(v, z)$, conditioned on the event that the right executions (in the view of A) are consistent with τ and $\mathtt{coins}_\tau$.

The claim now follows from the fact that D distinguishes $\mathsf{mim}^A_{\mathsf{com}}(v, z)$ and $\mathsf{sim}^S_{\mathsf{com}}(z)$ with probability $\frac{1}{p(n)}$, conditioned on the right executions being consistent with τ and $\mathtt{coins}_\tau$. □

Step 3: *Show that the simulator for A' violates the hiding property of* Com. We next use the simulator-extractor S' for A' to construct a (nonuniformly) *efficiently computable* experiment that is statistically close to hyb_1. Towards this goal, we first define an additional hybrid experiment $\mathsf{hyb}_2(\cdot)$. $\mathsf{hyb}_2(\cdot)$ proceeds just as hyb_1, except that, instead of emulating the left and right interactions for A', hyb_2 runs the combined simulator-extractor S' for A' to generate the view of A'. (The second output of hyb_2 is, however, still computed as in hyb_1—i.e., hyb_2 ignores the second output of S'.)

Claim 4.7. *The ensembles $\{\mathsf{hyb}_1(v')\}_{v' \in \{0,1\}^*}$ and $\{\mathsf{hyb}_2(v')\}_{v' \in \{0,1\}^*}$ are statistically close over $\{0, 1\}^*$.*

Proof. It directly follows from the statistical indistinguishability property of S' that the first output of $\mathsf{hyb}_1(\cdot)$ is statistically close to the first output of $\mathsf{hyb}_2(\cdot)$.

The claim is concluded by observing that the second output in both experiments is determined in the same way (as a function of the first output). □

Remark 2. Note that the proof of Claim 4.7 inherently relies on the *statistical* indistinguishability property of S'. Indeed, if the simulation had only been computationally indistinguishable, we would not have been able to argue indistinguishability of the second output of $\mathsf{hyb}_1(\cdot)$ and $\mathsf{hyb}_2(\cdot)$. This follows from the fact that the second output (which consists of the *actual* committed values) is not efficiently computable from the first output (i.e., the view).

We next define the final experiment $\mathsf{hyb}_3(\cdot)$. $\mathsf{hyb}_3(\cdot)$ proceeds just as hyb_2 with the exception that, instead of setting its second output $\bar{v}$ to the actual values committed to in the view of A, hyb_3 efficiently computes values $\bar{v} = v_1, .., v_m$ as follows. Recall that the combined-simulator extractor S' outputs both a view and witnesses to all accepting right interactions. For each accepting right interaction i in the reconstructed view of A, hyb_3 lets $v_i = \mathtt{value}_\tau(i)$ if $i \in \mathtt{committed}_\tau$ and otherwise sets v_i to be consistent with the witness output for execution i by the combined simulator-extractor S'. For all right executions j for which the reconstructed view of A is rejecting, instead set $v_j = \perp$.[16] Note that, in contrast to $\mathsf{hyb}_2(\cdot)$, $\mathsf{hyb}_3(\cdot)$ is efficiently computable.

CLAIM 4.8. *The ensembles* $\{\mathsf{hyb}_2(v')\}_{v' \in \{0,1\}^*}$ *and* $\{\mathsf{hyb}_3(v')\}_{v' \in \{0,1\}^*}$ *are statistically close over* $\{0,1\}^*$.

Proof. Recall that the only difference between the experiments $\mathsf{hyb}_2(\cdot)$ and $\mathsf{hyb}_3(\cdot)$ is the way the second output is computed; in the former it is computed as the actual values committed to in the view of A, whereas in the latter it is computed by relying on $\mathtt{value}_\tau$ and the witnesses output by the simulator-extractor S'. We show that except with negligible probability these values are identical, which concludes the claim.

First, note that in any given view of A it "trivially" holds that hyb_3 outputs the correct value for all rejecting right executions and all accepting right executions i such that $i \in \mathtt{committed}_\tau$. It remains only to consider accepting right executions i such that $i \notin \mathtt{committed}_\tau$. For these executions the value v_i computed by hyb_3 is obtained from the witnesses output by the simulator-extractor S'.

Assume, first, that A *never* is able to violate the security of $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ or is able to construct a commitment using Com that can be opened up in two different ways (i.e., violates the statistical binding property of Com). Under these assumptions, we show that the values computed by hyb_3 are identical to the actual values committed to in the view of A.

By the definition of the simulator-extractor, it holds that the witnesses output by S', for all accepting right interactions which use a different tag than the one used in the left interaction, are valid. In other words, values for all (nonrejected) right commitments that use a verification key vk for the signature scheme that is different from the one used in the left commitment are extracted. Furthermore, by our assumption that A is not able to break the statistical binding property of Com, it follows that the extracted values are identical to the actual values committed to in the view of A. Additionally, note that values for right commitments that have *exactly* the same transcript as the left commitment are "trivially" extracted (as they are just $\perp$). It remains only to analyze what happens to (nonrejecting) right commitments that use the same verification key as the left commitment but a different transcript. In this case, A must have been able to produce a signature on a new message, given only a randomly generated verification key and a single signed message of its choice

[16] Note that an interaction that is accepting in the view of A' can still be rejecting in the view of A, since in the latter we additionally require a valid signature.

(namely, the transcript on the left). This contradicts our assumption that A does not forge signatures.

We conclude that, conditioned on the event that A is not able to forge a signature or is able to break the statistical binding property of Com, $\mathsf{hyb}_2(\cdot)$ and $\mathsf{hyb}_3(\cdot)$ are identical. The claim now follows by the security of $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ and Com. □

By combining the above two claims we thus get the following.

Claim 4.9. *The ensembles* $\{\mathsf{hyb}_1(v')\}_{v' \in \{0,1\}^*}$ *and* $\{\mathsf{hyb}_3(v')\}_{v' \in \{0,1\}^*}$ *are statistically close over* $\{0,1\}^*$. *Furthermore,* $\mathsf{hyb}_3(\cdot)$ *is (nonuniformly) efficiently computable.*

Finally, by combining Claim 4.9 with Claim 4.6 we get that D distinguishes the second outputs of $\mathsf{hyb}_3(v)$ and $\mathsf{hyb}_3(0^n)$ with inverse polynomial probability. However, since hyb_3 is (nonuniformly) efficiently samplable, this contradicts the (nonuniform) hiding property of $\mathrm{Com}_{\mathrm{r}_\tau}$. More formally, define the distinguisher D' that proceeds as follows on input a commitment c' using Com_{r_τ}:

1. D' performs the same operations as hyb_3, except that, instead of generating the commitment c, it simply sets $c = c'$. Let $(\mathsf{view}, \bar{v})$ denote the output when executing $\tilde{H}$ in this manner.
2. Finally, D' outputs $D(\mathsf{view}, \bar{v})$.

It directly follows from the construction that $D'(c')$ is identically distributed to $D(\mathsf{hyb}_3(0^n))$ when c' is a random commitment to 0^n and identically distributed to $D(\mathsf{hyb}_3(v))$ when c' is a random commitment to v. We conclude that D'—which is efficient—distinguishes commitments (using Com_{r_τ}) to 0^n and v. □

5. Simulation extractability. We now turn to prove Lemma 4.3. We start by proving an analogous lemma for the "small" family $\{\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle\}_{\mathsf{tag} \in [2n]}$ of $2n$ protocols. Then we show how to extend the analysis to the family $\{\langle P_{\mathrm{TAG}}, V_{\mathrm{TAG}}\rangle\}_{\mathrm{TAG} \in \{0,1\}^n}$.

Lemma 5.1. *Suppose that* $\{\mathcal{H}_n\}_n$ *is collision-resistant for* $T(n)$*-sized circuits, that* **Com** *is perfectly hiding, that* $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}}\rangle$ *is a witness-independent argument of knowledge, that* $\langle P_{\mathsf{UA}}, V_{\mathsf{UA}}\rangle$ *is a universal argument, and that* $\ell(n) \geq 2n^2 + n$. *Then* $\{\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle\}_{\mathsf{tag} \in [2n]}$ *is one-many simulation extractable with tags of length* $t(n) = \log n + 1$.

Proof. Recall that one-many simulation extractability (Definition 4.1) means that there exists a combined simulator-extractor $S = (\mathsf{SIM}, \mathsf{EXT})$ that is able to simulate both the left and the right interactions for a man-in-the-middle adversary A, while simultaneously extracting witnesses to the m statements proved in the right interaction. The construction of S is fairly complex. To keep things simple, we decompose the description of the simulator into three simulation procedures, where each procedure relies on the previous (simpler) ones:

- *Basic simulator.* This consists of the simulator that is used to establish the traditional (stand-alone) zero-knowledge property of $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle$. The simulator is similar to the one used in Barak's original protocol [1].
- *Alternative simulator.* This consists of the simulator that is used for establishing the "simulation soundness" (cf. [44]) of $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle$. The simulator is designed to work in the presence of a man-in-the-middle adversary that is conducting a single left interaction of $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle$ concurrently with a single right interaction of $\langle P_{\tilde{\mathsf{tag}}}, V_{\tilde{\mathsf{tag}}}\rangle$. It guarantees that an adversary whose left view consists of a simulated execution of $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle$ cannot break the soundness of $\langle P_{\tilde{\mathsf{tag}}}, V_{\tilde{\mathsf{tag}}}\rangle$. The simulator is essentially identical to the one used by Pass [40].

- *Simulator-extractor.* The description of the simulation extraction procedure $S = (\mathsf{SIM}, \mathsf{EXT})$ relies on the previous two simulators. The simulator SIM relies on the basic simulator, whereas the extractor EXT (which also employs a simulation of the left interaction) makes use of the alternative simulator.

We turn to provide a description of the above simulation procedures. (We provide only a brief sketch of the basic and alternative simulators and assume familiarity with the protocols of [1] and [40]. For completeness, the description of the simulator-extractor is nevertheless self-contained.)

5.1. Basic simulator. Given the program V^*_{tag} of an adversary verifier, the basic simulator acts as follows. In stage 1 of the protocol (i.e., in slots 1 and 2), the simulator proceeds by committing to the program $\Pi \stackrel{\text{def}}{=} V^*_{\mathsf{tag}}$. Let s_1, s_2 be the randomness used for the commitments.

In stage 2 of the protocol, the simulator proves that it committed to the program of the verifier in slot 1. More concretely, the simulator uses the tuple $\langle \Pi, c_1, s_1 \rangle$ as a witness for $\langle h, c_1, r_1 \rangle \in L_{\mathsf{sim}}$ (where L_{sim} is the language that corresponds to R_{sim}). This is a valid witness, since (1) by the definition of Π it holds that $\Pi(c_1) = r_1$, and (2) as long as $\ell(n) \geq 3n$, for every $\mathsf{tag} \in [n]$, $|r_i| - |c_i| = \ell(n) - |c_i| \geq n$.

5.2. Alternative simulator. The alternative simulator is constructed with a man-in-the-middle adversary A in mind. Consider an A that manages to violate the soundness of protocol $\langle P_{\tilde{\mathsf{tag}}}, V_{\tilde{\mathsf{tag}}} \rangle$ while verifying a simulated proof of $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}} \rangle$. We show how to construct a cheating prover P^* for a single instance of $\langle P_{\tilde{\mathsf{tag}}}, V_{\tilde{\mathsf{tag}}} \rangle$ by forwarding A's messages in $\langle P_{\tilde{\mathsf{tag}}}, V_{\tilde{\mathsf{tag}}} \rangle$ to an external honest verifier V and internally simulating the messages of $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}} \rangle$ for A. The problem that arises in the attempt to simulate is that the code of the external verifier V is not available to the simulator. This means that a stand-alone simulation of the protocol $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}} \rangle$ cannot be completed as it is, since it explicitly requires possession of a "short" program Π that would have generated the corresponding verifier messages.

On a high level, the prover P^* simulates the left interaction in the following way.[17] In slot 1 of the protocol, the simulator proceeds by committing to the program $\Pi_1 \stackrel{\text{def}}{=} A$. So far its instructions are just like the basic simulator. In slot 2, however, the simulator commits to a program Π_2 which consists of both the code of A and all messages A has received from $V_{\tilde{\mathsf{tag}}}$ in the right interaction. In stage 2 of the protocol, the simulator attempts to prove that it committed to the program of the verifier in either slot 1 or slot 2. The simulator will succeed in this task provided that there exists a "short" message y (the actual required length of y is determined by the tag tag and the slot number) such that $\Pi_1(y) = r_1$ or $\Pi_2(y) = r_2$, where r_1 and r_2 denote the challenges receive in slots 1 and 2, respectively (of $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}} \rangle$).

Note that, except for the "long" challenges $\tilde{r_1}, \tilde{r_2}$ sent by the verifier of $\langle P_{\tilde{\mathsf{tag}}}, V_{\tilde{\mathsf{tag}}} \rangle$, we do have a description of all messages sent to the adversary A that is shorter than $\ell(n) - n$ (since $\ell(n) = \ell'(n) + n$, where $\ell'(n)$ upper bounds the total length of both prover and verifier messages, except for the challenges r_1, r_2). In order to show that we can still perform a simulation, even in the presence of these messages (for which we do not have a short description), we use the fact that it is sufficient to have a short description of the messages sent in *one* of the slots of $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}} \rangle$. As in [40], we separate between two different schedulings:

[17] We provide a detailed description of the actual simulation procedure when we later apply it in the construction of the simulator-extractor.

- *There exists one "free" slot j in $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle$ in which neither of $\tilde{r_1}$, $\tilde{r_2}$ are contained.* In this case the "free" slot j in $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle$ can be used to perform a basic simulation (since in this case the simulator did indeed produce a commitment c_j to the code of a machine that on input c_j outputs the challenge r_j in slot j).
- *The messages r_1 and r_2 in $\langle P_{\tilde{\mathsf{tag}}}, V_{\tilde{\mathsf{tag}}}\rangle$ occur in slots* 1 *and* 2*, respectively, in $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle$.* By the construction of the protocols it follows that the length of either the first or the second challenge in $\langle P_{\tilde{\mathsf{tag}}}, V_{\tilde{\mathsf{tag}}}\rangle$ is at least $\ell(n)$ bits longer than the corresponding challenge in $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle$. Thus there exists a slot j in $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle$ such that, even if we include the verifier's challenge $\tilde{r}_j$ from the protocol $\langle P_{\tilde{\mathsf{tag}}}, V_{\tilde{\mathsf{tag}}}\rangle$ in the description y, we still have $\ell(n) - n$ bits to describe all other messages.

5.3. Simulator-extractor. Consider a man-in-the-middle adversary A. We assume without loss of generality that A is deterministic and has the auxiliary input z hardwired in. Let k denote the number of rounds in $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle$, and let m be an upper bound on the number of right interactions in which A participates. We describe a combined simulator-extractor $S = (\mathsf{SIM}, \mathsf{EXT})$ that proceeds as follows on input x, z, and TAG.

5.3.1. Simulation of view. We start by describing a machine SIM that simulates the view of A. This requires simulating all of the left and the right interactions for A. In the right interactions SIM acts as a verifier. Thus, simulation is straightforward and is performed by simply playing the role of an honest verifier in all of the executions of the protocol. In the left interaction, on the other hand, SIM is supposed to act as a prover, and thus the simulation task is more involved. Towards its goals, SIM acts as follows:

1. For all $i \in [m]$, pick random $\bar{r}_i = (r_{i,1}, \ldots, r_{i,k})$ honest verifier messages for the right interactions. Messages in the right interactions are then emulated by playing the role of the honest verifiers with the fixed random messages $\bar{r}_1, \ldots, \bar{r}_m$. That is, in order to emulate the jth message in the ith right interaction, SIM forwards the message $r_{i,j}$ to A.
2. The left interaction is simulated as follows. SIM views the execution of A and the emulation of the right interactions (with the fixed messages $\bar{r}_1, \ldots, \bar{r}_m$) as a *stand-alone* verifier for the left interaction and applies a close variant of the basic simulator to this interaction. Let $\Pi(\cdot)$ denote the joint code of A and the emulation of the right interactions (including the coins $\bar{r}_1, \ldots, \bar{r}_m$). Whereas the basic simulator would have committed to $\Pi(\cdot)$, we instead let SIM commit to a program $\Pi'(b, \cdot)$ that is defined as follows:
   (a) If $b = 0$, execute $\Pi(\cdot)$;
   (b) if $b = 1$, execute $\Pi(\cdot)$ with the exception that messages $\bar{r}_i = (r_{i,1}, \ldots, r_{i,k})$ (i.e., messages of the ith right interaction) are not emulated but rather received externally as input.

   Thereafter, SIM proceeds exactly as the basic simulator, by additionally using both $b = 0$ and Π' as a witness in stage 2 of the protocol. More concretely, SIM starts by computing $h = \Pi(\cdot)$. It then generates prover commitments $c_1 = \mathrm{Com}(h(\Pi'); s_1)$ and $c_2 = \mathrm{Com}(h(\Pi'); s_2)$, where $s_1, s_2 \overset{\mathrm{R}}{\leftarrow} \{0,1\}^{\mathrm{poly}(n)}$. By using c_1 and c_2, it computes $r_1 = \Pi(c_1)$ and $r_2 = \Pi(c_1, c_2)$. By combining the messages together, this results in a stage 1 transcript $\tau_1 = \langle h, c_1, r_1, c_2, r_2\rangle$. By the definition of $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle$, the transcript τ_1 induces a stage 2 *WIUARG* with $\Pi(h, c_1, c_2)$ as the verifier and $(x, \langle h, c_1, r_1\rangle, \langle h, c_2, r_2\rangle)$ as the common input. By using $\langle \Pi', (0, h, c_1), s_1\rangle$ as the witness for the statement $\langle h, c_1, r_1\rangle \in L_{\mathsf{sim}}$, the SIM follows the prescribed prover strategy of the *WIUARG* and produces a convincing stage 2 transcript τ_2. Since $r_1 = \Pi(c_1) = \Pi'(0, c_1)$ and since $|0| + |c_1| \leq \ell(n)$, it follows that SIM can always succeed in this task.

Figure 9 demonstrates the definition of SIM as well as of the program $\Pi'(b, \cdot)$ (for simplicity the various sessions are depicted as if they were executed sequentially).

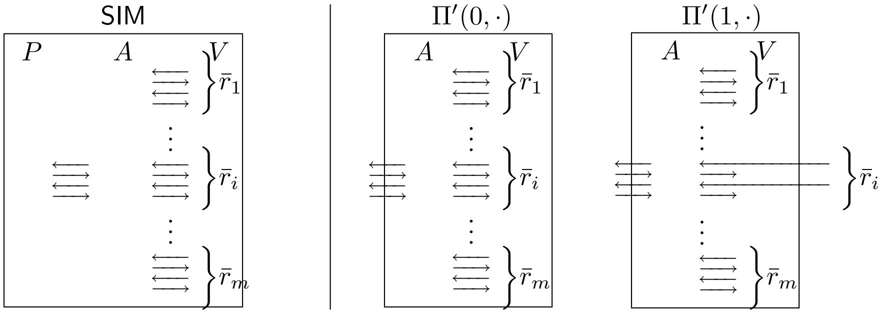


FIG. 9. *The simulator* SIM *and the program* $\Pi'(b, \cdot)$.

5.3.2. Extraction of witnesses. Once the view of A has been simulated, we turn to the extraction of witnesses to the statements proved by A. Note that we need to extract witnesses to all concurrent right interactions. Towards this goal we rely on a variant of Lindell's concurrent extraction technique [33] combined with the alternative simulator technique described in section 5.2. In a sense, this can be seen as a (nontrivial) extension of the method of Pass and Rosen [42] (which was used to show a similar property for the simpler case of only one right interaction).

The machine EXT fixes the random coins of the simulator SIM and iteratively extracts witnesses for each of the right interactions. More specifically, EXT starts by sampling a random execution of SIM, using random coins $\bar{s}, \bar{r}$. Let $x_1, \ldots, x_m$ be the inputs corresponding to the m sessions that have taken place in the right interaction.

For each $i \in [m]$ such that the ith right session was *not* accepting, EXT will assume that no witness exists for the corresponding statement x_i and will refrain from extraction. For all $i \in [m]$ so that the ith right session *is* accepting in this execution of SIM, and for which the tag of the ith session is different from the tag of the left session, EXT will attempt to extract a witness for the statement x_i being proved in the corresponding session.

To do so EXT constructs a stand-alone prover P_i for the ith right interaction $\langle P_{\tilde{\mathsf{tag}}_i}, V_{\tilde{\mathsf{tag}}_i}\rangle$ and from which it will later attempt to extract the witness. In principle, the prover P_i will follow SIM's actions by using the same random coins $\bar{s}, \bar{r}$ used for initially sampling the execution of SIM. However, P_i's execution will differ from SIM's execution in the following important ways:

1. Messages in the ith right session are no longer emulated internally but forwarded externally.
2. In the simulation of the left protocol, use the alternative simulator from section 5.2 in order to complete stage 2 of the protocol.

Figure 10 demonstrates the definition of P_i (as before, the sessions are depicted sequentially).

The reason for using the alternative simulation instead of the basic one is that

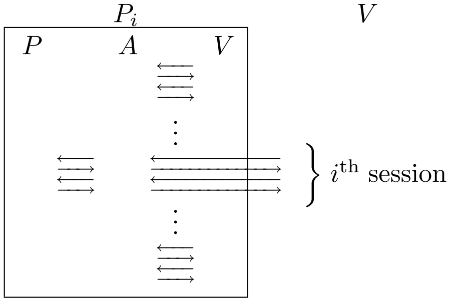


FIG. 10. *The prover P_i.*

the latter might not be able to commit to the external messages of the ith right interaction (as it might not know these messages at the time it commits). Note that the way the simulation within P_i is defined the program committed to in stage 1 is Π'. To enable the alternative simulation with a commitment to Π' in stage 1, we let the simulator additionally provide the input $b = 1$ to Π' as part of the witness in stage 2 (this enables Π' to depend on the external messages in the ith right session). The alternative simulation technique, combined with the fact that there is only *one* external interaction on the right-hand side, is what eventually enables the simulation to go through.

The actual witness used by the simulator in stage 2 depends on the scheduling of the messages. We distinguish between the following cases, depending on where the ith session has started with relation to the messages c_1, c_2 in the left-hand side protocol (see Figure 11).

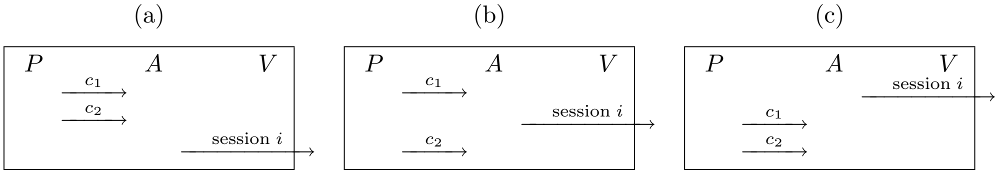


FIG. 11. *Three possible "starting points" for session i.*

In each corresponding case, EXT acts as follows:

- *Both c_1 and c_2 have been sent before session i begins (Figure* 11(a)*).* In this event slot 1 has no external messages, and the basic simulation can be performed; i.e., EXT can use Π' as a witness for $\langle h, c_1, r_1 \rangle \in L_{\mathsf{sim}}$ in stage 2 (just as in SIM).
- *c_1 has been sent but not c_2 (Figure* 11(b)*).* Let M_1 and M_2 denote the "external" messages A receives on the right-hand side in slots 1 and 2 of the left interaction, respectively (see Figure 12 for two "representative" schedulings). In this case, we define $\Pi'_2(b, \cdot) = \Pi'(b, M_1, \cdot)$ and let EXT send $c_2 = \mathrm{Com}(\Pi'_2; s)$ (whereas c_1 is defined just as in SIM).

  Consider a stage 1 transcript $\tau_1 = \langle h, c_1, r_1, c_2, r_2 \rangle$ of the left interaction. By the construction of $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}} \rangle$, and from the fact that $\tilde{\mathsf{tag}}_i$ is different from tag, it must be the case that either $|M_1| \leq |r_1| - \ell(n)$ or $|M_2| \leq |r_2| - \ell(n)$. This is implied by the following simple fact.

  Fact 5.2. If $\mathsf{tag} \neq \tilde{\mathsf{tag}}$ then there exists $i \in \{1, 2\}$ so that $|\tilde{r_i}| \leq |r_i| - \ell(n)$.

  In particular, either $|M_1| + |c_1| + n \leq |r_1| - n$ or $|M_2| + |c_1| + n \leq |r_2| - n$.

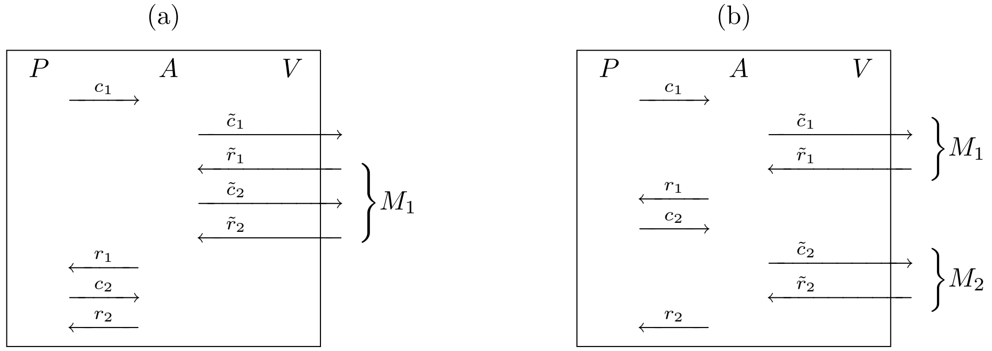


FIG. 12. *Two "representative" schedulings.*

Furthermore, c_1 is a commitment to Π', c_2 is a commitment to Π'_2, $r_1 = \Pi'(1,(c_1,M_1))$, and $r_2 = \Pi'_2(1,(c_1,M_2))$. Thus, either $w_1 = \Pi', 1, (c_1, M_1), s_1$ is a valid witness for $\langle h, c_1, r_1\rangle \in L_{\mathsf{sim}}$ or $w_2 = \Pi'_2, 1, (c_2, M_2), s_2$ is a valid witness for $\langle h, c_2, r_2\rangle \in L_{\mathsf{sim}}$. If the former is true, EXT follows the prescribed prover by using w_1 as a witness and otherwise uses w_2 as a witness.

- *Neither of c_1 or c_2 have been sent (Figure* 11(c)*).* In this case EXT first generates a commitment c_1 just as SIM would, i.e., lets c_1 be a commitment to Π', and then performs the same operations as in the previous case.

It follows from the description above that the simulation employed by P_i on the left interaction is always able to convince A of the validity of the statement proved on the left interaction.

Once P_i is constructed, EXT can apply the (stand-alone) extractor, guaranteed by the proof of knowledge property of $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle$, to P_i and extract a witness to the statement x_i. In the unlikely event that the extraction failed in any of the m executions, EXT outputs *fail*; otherwise, it outputs all of the extracted witnesses.

Remark 3. It is important to have a P_i for the *entire* protocol $\langle P_{\tilde{\mathsf{tag}}_i}, V_{\tilde{\mathsf{tag}}_i}\rangle$ (and not just for $\langle P_{UA}, V_{UA}\rangle$). This is in order to argue that the witness extracted is a witness for x_i and not a witness to $\langle h, c_1, r_1\rangle \in L_{\mathsf{sim}}$ or to $\langle h, c_2, r_2\rangle \in L_{\mathsf{sim}}$ (which could indeed be the case if we fixed the messages $\langle h, c_1, r_1, c_2, r_2\rangle$ in advance).

The output of S. Finally the combined simulator-extractor S outputs *fail* whenever EXT does. Otherwise, S outputs whatever SIM outputs, followed by whatever EXT outputs.

Remark 4. It is important to have both SIM and EXT use the same simulator program S (with the same random coins) in their respective executions. Otherwise, we are not guaranteed that the statement $\tilde{x}$ appearing in the output of SIM is the same one from which EXT extracts a witness.[18]

5.4. Correctness of the simulation-extraction. We proceed to show the correctness of the combined simulator-extractor $S = (\mathsf{SIM}, \mathsf{EXT})$. We start by showing that the view of A in the simulation by SIM is identical to its view in an actual interaction. Let the random variable $\mathsf{SIM}(x, z, \textsc{tag})$ denote the view of A in the simulation by SIM performed in the execution of $S(x, z, \textsc{tag})$.

[18] The statement $\tilde{x}_i$ will remain unchanged because $\tilde{x}_i$ occurs prior to any message in $\langle P_{\tilde{\mathsf{tag}}_i}, V_{\tilde{\mathsf{tag}}_i}\rangle$ (and hence does not depend on the external messages received by P_i).

Claim 5.3.

$$\{\mathsf{SIM}(x, z, \text{TAG})\}_{x\in L, z\in\{0,1\}^*, \mathsf{tag}\in[2|x|]} \quad \textit{and} \quad \{\mathsf{view}_A(x, z, \text{TAG})\}_{x\in L, z\in\{0,1\}^*, \mathsf{tag}\in[2|x|]}$$

are identically distributed.

Proof. The claim follows from (1) the perfect zero-knowledge property of $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle$, and (2) the fact that the emulation of the right interactions by SIM is perfect. Specifically, consider the following hybrid experiments:

1. Let H_0 denote the view of A in the simulated execution.
2. Let H_1 denote the view of A in a simulated execution when letting the simulator use the true witness w for x in the special-purpose *UARG* $\langle P_{\mathsf{sUA}}, V_{\mathsf{sUA}}\rangle$ in stage 2 (instead of using the "fake" witness). Thus the only difference between H_0 and H_1 is the choice of the witness used in $\langle P_{\mathsf{sUA}}, V_{\mathsf{sUA}}\rangle$.
3. Let H_2 denote the real execution. Note that the only difference between H_1 and H_2 is that, in H_1, A receives commitments c_1, c_2 to a program Π', whereas in H_2 it receives a commitment to the string 0^k.

Subclaim 5.4. *H_0 is identically distributed to H_1.*

Proof. The claim follows from the witness-independent property of $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}}\rangle$ used in stage 2. More precisely, assume for contradiction that H_0 is not identically distributed to H_1. Then there must exist some stage 1 transcript such that the proofs generated in stage 2 in H_0 and H_1 are not identically distributed, in contradiction to the witness independence property. (Here we use the fact that there exist two possible witnesses for stage 2—one is the witness used by the simulator and the other is w.) □

Subclaim 5.5. *H_1 is identically distributed to H_2.*

Proof. The claim directly follows from the perfect hiding property of the commitments used to generate c_1 and c_2. □

This completes the proof of Claim 5.3. □

We proceed to show that EXT outputs `fail` with negligible probability. Let the random variable $\mathsf{EXT}(x, z, \mathsf{tag})$ denote the output of EXT in the execution of $S(x, z, \mathsf{tag})$.

Claim 5.6. *There exists a negligible function $\mu(\cdot)$ such that for every $x \in L$, $z \in \{0,1\}^*$, $\mathsf{tag} \in [2|x|]$*

$$\Pr\left[\mathsf{EXT}(x, z, \mathsf{tag}) = \texttt{fail}\right] \le \mu(|x|).$$

Proof. Recall that EXT proceeds by constructing stand-alone provers P_i and then applying the (stand-alone) extractor, guaranteed by the proof of knowledge property of $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle$, to P_i, in order to extract witnesses w_i to the statements x_i. Note that EXT outputs `fail` only in the event that extraction from one of the right interactions $i \in [m]$ fails. By the proof of knowledge property of $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle$ it holds that for each execution i extraction for execution i fails only with negligible probability (recall that the extractor is invoked only when A provides an accepting proof in execution i). Since the extraction procedure is repeated at most m times (at most once per right interaction), we conclude (by the union bound) that the probability that extraction fails for any of the right interactions is negligible. □

By combining Claims 5.3 and 5.6, we conclude that the first output of S is statistically close to A's view in an actual interaction. As in Definition 4.1, let the random variable $S_1(x, z, \text{TAG})$ denote the first output of $S(x, z, \text{TAG})$.

Claim 5.7. $\{S_1(x, z, \text{TAG})\}_{x\in L, z\in\{0,1\}^*, \mathsf{tag}\in[2|x|]}$ *and* $\{\mathsf{view}_A(x, z, \text{TAG})_{x\in L, z\in\{0,1\}^*, \mathsf{tag}\in[2|x|]}$ *are statistically close over L.*

Proof. Recall that the first output of S consists of the view of A as generated by SIM. By Claim 5.3 it follows that the first output of S is identically distributed to a "real" interaction, conditioned on the event that S does not output $\mathtt{fail}$. However, since this event happens only when EXT outputs $\mathtt{fail}$, which by Claim 5.6 happens only with negligible probability, the claim follows. □

We proceed to show the correctness of the extraction.

CLAIM 5.8. *Let $x \in L$, $z \in \{0,1\}^*$, and* $\textsc{tag} \in \{0,1\}^{2|x|}$, *and let* $(\mathsf{view}, \bar{w})$ *denote the output of* $S(x, z, \textsc{tag})$ *(on input some random tape). Let $\tilde{x}_1 \ldots, \tilde{x}_m$ be the right-execution statements appearing in* view, *and let* $\tilde{\mathsf{tag}}_1 \ldots \tilde{\mathsf{tag}}_m$ *denote the corresponding right-execution tags. Then, for any $i \in [m]$ such that the ith right execution in* view *is accepting* and $\mathsf{tag} \neq \tilde{\mathsf{tag}}_i$, *$\bar{w}$ contains a witness w_i so that $R_L(\tilde{x}_i, w_i) = 1$.*

Proof. First, note that, since S always outputs $\mathtt{fail}$ whenever the extraction by EXT fails, the claim trivially holds in the event that the extraction by EXT fails. Consider, next, the case when extraction by EXT does not fail. Recall that EXT performs extraction for all right executions which satisfy the properties described in the hypothesis (i.e., accepting proofs and different tags). Furthermore, for each such interaction i, the stand-alone prover P_i—constructed by EXT—uses the same random coins as SIM in order to emulate all of the interactions before session i begins. In addition, the prescribed actions for the simulation by EXT are identical to the prescribed actions for the simulation by SIM. This means that the statement proved by P_i will be identical to the statement proved in the view output by SIM. Finally, by our assumption that the extraction by EXT does not fail, we conclude that a valid witness for the statement proved by P_i is extracted. This concludes the claim. □

We conclude the proof by bounding the running time of the combined simulator-extractor S.

CLAIM 5.9. *$S(x, z, \mathsf{tag})$ runs in expected polynomial time (in $|x|$).*

Proof. We start by noting that the running time of SIM is polynomial. Recall that the program Π' committed to by SIM is of size $\mathrm{poly}(n)$. It thus directly follows that simulation of stage 1 messages can be done in polynomial time. Furthermore, it follows that the verification time of R_{sim} on the instance $\langle h, c_1, r_1 \rangle$ is polynomial in n. Finally, by the relative prover efficiency of $\langle P_{\mathsf{UA}}, V_{\mathsf{UA}} \rangle$ it holds that the simulator can also generate stage 2 messages in polynomial time.

It now remains only to show that the *expected* running time of EXT is also polynomial. Recall that EXT proceeds by first sampling a view by using SIM and then proceeds to extract witnesses in all *accepting* right executions. We show that, for every right execution i, the expected running time needed to extract a witness from that execution is polynomially bounded. Since the number of right interactions is polynomially bounded, we conclude by linearity of expectations that the total expected running time of the combined simulator-extractor $S = (\mathsf{SIM}, \mathsf{EXT})$ is polynomially bounded.

Let $\mathtt{view}_i$ denote the partial view for A in an emulation by SIM up until A is about to start its ith right execution. Let $p_i(\mathtt{view}_i)$ denote the probability that A produces an accepting proof in the ith right execution in the simulation by SIM, given that SIM has fed to A the view $\mathtt{view}_i$. Let $p'_i(\mathtt{view}_i)$ denote the probability that A produces an accepting proof in the ith right execution in the simulation by P_i (constructed in EXT), given that EXT has fed A the view $\mathtt{view}_i$.

SUBCLAIM 5.10. *Let $\mathtt{view}_i$ denote the partial view for A in an emulation by* SIM *up until A is about to start its ith right execution. Then $p_i(\mathtt{view}_i) = p'_i(\mathtt{view}_i)$.*

Proof. The claim follows from the *perfect* indistinguishability of the basic simulator used by SIM and the alternative simulator used by EXT (this is proved similarly to Claim 5.3). □

Note that if we assume only that $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle$ is statistical zero knowledge, we could conclude only that $p'_i(\mathtt{view}_{\mathtt{i}})$ is *negligibly close* to $p_i(\mathtt{view}_{\mathtt{i}})$. This would not be sufficient to bound the running time of the simulator (as this would have introduced difficulties similar to the ones discussed in [21]).

By the proof of knowledge property of $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle$ it holds that, for any partial view $\mathtt{view}_i$ up until A is about to start its ith right execution, the expected running time of the extractor is bounded by

$$\frac{\mathrm{poly}(n)}{p'_i(\mathtt{view}_i)}.$$

Since the probability of invoking the extraction procedure given this partial view is $p_i(\mathtt{view}_i)$, the expected number of steps used to extract a witness is[19]

$$p_i(\mathtt{view}_{\mathtt{i}})\frac{\mathrm{poly}(n)}{p'_i(\mathtt{view}_{\mathtt{i}})} = p_i(\mathtt{view}_{\mathtt{i}})\frac{\mathrm{poly}(n)}{p_i(\mathtt{view}_{\mathtt{i}})} = \mathrm{poly}(n).$$

We conclude that the expected time needed to extract the witness in the ith right execution is polynomially bounded. The claim follows. □

This completes the proof of Lemma 5.1. □

5.5. Analyzing the family of 2^n protocols. By relying on the proof from section 5.3, we now argue that the family $\{\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle\}_{\text{TAG}\in\{0,1\}^n}$ is also one-many simulation extractable. The key for demonstrating this is to show that the protocols $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle$ are simulation extractable as long as the number of left interactions is a priori *bounded* (in contrast to the single left interaction considered in Definition 4.1) and even if the number of right interactions is unbounded.

Specifically, consider a man-in-the-middle adversary A that is simultaneously participating in $k = k(n)$ left interactions of $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle$, acting as a verifier, and an (unbounded) polynomial number of right interactions of $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle$, acting as a prover. Let $\mathsf{view}_A(x, z, \mathsf{tag})$ denote the view of $A(x, z)$ when receiving left proofs of statements $\bar{x} = x_1, \ldots, x_k$, using identity strings $\bar{\mathsf{tag}} = \mathsf{tag}_1, \ldots, \mathsf{tag}_k$, and proving statements of its choice in the right interaction (using tags of its choice). Given a function $t = t(n)$ and some $k \in N$, we use the notation $\{\cdot\}_{\bar{x},z,\bar{\mathsf{tag}}}$ as shorthand for $\{\cdot\}_{x_1,\ldots,x_k\in L, z\in\{0,1\}^*, \mathsf{tag}_1,\ldots,\mathsf{tag}_k\in\{0,1\}^{t(|x|)}}$.

DEFINITION 5.11 (bounded-many simulation extractability). *A family $\{\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle\}_{\mathsf{tag}\in\{0,1\}^*}$ of interactive proofs for the language L is said to be* k-bounded-many simulation extractable *with tags of length $t = t(n)$ if, for* any *polynomial $p(\cdot)$ and any man-in-the-middle adversary A that participates in $k = k(n)$ left interactions and at most $m = p(n)$ right interactions, there exists a probabilistic expected poly-time machine S such that:*

1. *The probability ensembles $\{S_1(\bar{x}, z, \bar{\mathsf{tag}})\}_{\bar{x},z,\bar{\mathsf{tag}}}$ and $\{\mathsf{view}_A(\bar{x}, z, \bar{\mathsf{tag}})\}_{\bar{x},z,\bar{\mathsf{tag}}}$ are statistically close over L, where $S_1(\bar{x}, z, \bar{\mathsf{tag}})$ denotes the first output of $S(\bar{x}, z, \bar{\mathsf{tag}})$.*
2. *Let $x_1, \ldots, x_k \in L$, $z \in \{0,1\}^*$, and $\mathsf{tag}_1 \ldots \mathsf{tag}_k \in \{0,1\}^{t(|x|)}$, and let* $(\mathsf{view}, \bar{w})$ *denote the output of $S(\bar{x}, z, \bar{\mathsf{tag}})$ (on input some random tape). Let $\tilde{x}_1 \ldots, \tilde{x}_m$ be the right-execution statements appearing in* view, *and let*

[19] It is here that complications arise in the case when $p'_i \neq p_i$. Note that the expected number of steps is no longer guaranteed to be polynomial in this case, *even* if p'_i is negligibly close to p_i.

$\tilde{\mathsf{tag}}_1, \ldots, \tilde{\mathsf{tag}}_m$ *denote the corresponding right-execution tags. Then, for any* $i \in [m]$ *such that the* i*th right execution in* view *is accepting* and*, for all* $j \in [k]$ $\tilde{\mathsf{tag}}_i \neq \mathsf{tag}_j$*,* $\bar{w}$ *contains a witness* w_i *so that* $R_L(\tilde{x}_i, w_i) = 1$*.*

LEMMA 5.12. *Suppose that* $\{\mathcal{H}_n\}_n$ *is collision-resistant for* $T(n)$*-sized circuits, that* **Com** *is perfectly hiding, that* $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}}\rangle$ *is a witness-independent argument of knowledge, that* $\langle P_{\mathsf{UA}}, V_{\mathsf{UA}}\rangle$ *is a universal argument, and that* $\ell(n) \geq 2n^3 + n$*. Then* $\{\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle\}_{\mathsf{tag}\in[2n]}$ *is* n*-bounded-many simulation extractable with tags of length* $t(n) = \log n + 1$*.*

Proof. The proof is essentially identical to the proof of one-many simulation extractability of $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle$ (Lemma 5.1). The only difference is that, in the simulation by SIM (and EXT), the message r_1 in the ith left execution can no longer be computed as $\Pi(c_1)$ but is in fact defined as $\Pi(M)$, where M denotes all left-hand side prover messages that have occurred before r_1 (the same holds analogously for r_2). This creates a potential problem when simulating the stage 2 messages in the jth left protocol.

The key observation is that the *total* length of all prover messages on the left interaction does not exceed $2n^3$ (here we assume without loss of generality that the length of all prover messages in a session is upper bounded by n^2). Thus SIM (as well as EXT) can include *all* left-hand side prover messages sent to A before the message r_1 (or r_2 depending on what "slot" the simulator uses) as part of the witness for either $\langle h, c_1, r_1\rangle \in L_{\mathsf{sim}}$ or $\langle h, c_2, r_2\rangle \in L_{\mathsf{sim}}$. □

Our main technical lemma (Lemma 4.3) is finally obtained by combining Lemmas 5.1 and 5.12.

LEMMA 4.3 (main technical lemma). *Suppose that* $\{\mathcal{H}_n\}_n$ *is collision-resistant for* $T(n)$*-sized circuits, that* **Com** *is perfectly hiding, that* $\langle P_{\mathsf{pWI}}, V_{\mathsf{pWI}}\rangle$ *is a witness-independent argument of knowledge, that* $\langle P_{\mathsf{UA}}, V_{\mathsf{UA}}\rangle$ *is a universal argument, and that* $\ell(n) \geq 2n^3+n$*. Then* $\{\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle\}_{\text{TAG}\in\{0,1\}^n}$ *is one-many simulation extractable with tags of length* $t(n) = n$*.*

Proof. Consider a man-in-the-middle adversary A that is verifying a statement x with identity string $\text{TAG} = \text{TAG}_1, \ldots, \text{TAG}_n$ in the left interaction while proving m statements $\tilde{x}_1, \ldots, \tilde{x}_m$ in the right interaction, where for $i \in [m]$ the ith right session has identity string $\tilde{\text{TAG}}^i = \tilde{\text{TAG}}^i_1, \ldots, \tilde{\text{TAG}}^i_n$. We show how to construct a simulator-extractor $S = (\mathsf{SIM}, \mathsf{EXT})$ that simulates the view of A while extracting all of the witnesses for statements $\tilde{x}_i$ for which $\tilde{\text{TAG}}^i \neq \text{TAG}$.

First, observe that, for any $i \in [m]$ so that $\tilde{\text{TAG}}^i \neq \text{TAG}$, there exist $i_0 \in [n]$ for which $(i_0, \tilde{\text{TAG}}^i_{i_0}) \neq (j, \text{TAG}_j)$ for all $j \in [n]$ (just take the i_0 for which $\tilde{\text{TAG}}^i_{i_0} \neq \text{TAG}_{i_0}$). Let $\tilde{\mathsf{tag}}_i = (i_0, \tilde{\text{TAG}}^i_{i_0})$.

Given a one-many adversary A for $\langle P_{\text{TAG}}, V_{\text{TAG}}\rangle$, we next construct an n-many adversary A' for $\langle P_{\mathsf{tag}}, V_{\mathsf{tag}}\rangle$ that runs n parallel sessions in the left interaction and $m' = mn$ concurrent sessions in the right interaction. The inputs and identity strings for the various sessions are defined as follows:

- *Left sessions.* For $j \in [n]$ the common input of the jth left session is $x_j = x$, and the identity string is $\bar{\mathsf{tag}} = (j, \text{TAG}_j)$.
- *Right sessions.* For $(i, j) \in [m] \times [n]$, the input to the (i, j)th right session is $\tilde{x}_j$, and the identity string is $\tilde{\mathsf{tag}}^i_j = (j, \tilde{\text{TAG}}^i_j)$.

By Lemma 5.12 there exists a simulator S' that produces a view that is statistically close to the real view of A' and outputs witnesses to all right executions for which the tag is different from all of $(1, \text{TAG}_1), \ldots, (n, \text{TAG}_n)$. By relying on S', we construct the simulator-extractor S. $S(x, z, \text{TAG})$ proceeds as follows. It parses TAG as

$\textsc{tag} = \textsc{tag}_1, \ldots, \textsc{tag}_n$, where $\textsc{tag}_i \in \{0,1\}$. For $i \in [n]$, let $x_i = x$, $\mathsf{tag}_i = (i, \textsc{tag}_i)$. Let $(\mathsf{view}, \bar{w})$ denote the output of $S'(x_1, \ldots x_n, z, \mathsf{tag}_1, \ldots \mathsf{tag}_n)$. Additionally, let $\tilde{x}_1 \ldots, \tilde{x}_{m'}$ be the right-execution statements appearing in view and $\tilde{\textsc{tag}}^1 \ldots \tilde{\textsc{tag}}^{m'}$ the correspoding right-execution tags. As observed above, for any $i \in [m]$ so that $\tilde{\textsc{tag}}^i \neq \textsc{tag}$, there exists some identity $\tilde{\mathsf{tag}}_i$ that A' uses in the proof of the ith right interaction which is different than *all* identities $(\mathsf{tag}_1, \ldots, \mathsf{tag}_n)$ used in the n left interactions. Thus, for every $i \in [m]$ so that $\tilde{\textsc{tag}}_i \neq \textsc{tag}$, S can successfully find a witness for the statement $\tilde{x}_i$ in $\bar{w}$. S finally outputs view and the witnesses obtained above. The correctness of the simulator-extractor S follows directly from the construction. □

Acknowledgments. We are grateful to Silvio Micali and Moni Naor for many helpful discussions and encouragement. Thanks also to the anonymous reviewers for their careful reading and thoughtful comments.

REFERENCES

[1] B. Barak, *How to go beyond the black-box simulation barrier*, in Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science, 2001, pp. 106–115.

[2] B. Barak, *Constant-round coin-tossing or realizing the shared random string model*, in Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002, pp. 345–355.

[3] B. Barak, R. Canetti, Y. Lindell, R. Pass, and T. Rabin, *Secure computation without authentication*, in Proceedings of CRYPTO 2005.

[4] G. Brassard, D. Chaum, and C. Crépeau, *Minimum disclosure proofs of knowledge*, J. Comput. System Sci., 37 (1988), pp. 156–189.

[5] B. Barak and O. Goldreich, *Universal arguments and their applications*, in Proceedings of the 17th CCC, 2002, pp. 194–203.

[6] M. Bellare and O. Goldreich, *On defining proofs of knowledge*, in Proceedings of CRYPTO'92, Lecture Notes in Comput. Sci. 740, Springer, New York, 1993, pp. 390–420.

[7] M. Blum, *Coin flipping by telephone*, in Proceedings of CRYPTO 1981, 1981, pp. 11–15.

[8] M. Bellare, R. Impagliazzo, and M. Naor, *Does parallel repetition lower the error in computationally sound protocols?*, in Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science, 1997, pp. 374–383.

[9] R. Canetti and M. Fischlin, *Universally composable commitments*, in Proceedings of CRYPTO 2001, pp. 19–40.

[10] R. Canetti, S. Halevi, J. Katz, Y. Lindell, and P. D. MacKenzie, *Universally composable password-based key exchange*, in Proceedings of EUROCRYPT 2005, pp. 404–421.

[11] I. Damgård and J. Groth, *Non-interactive and reusable non-malleable commitment schemes*, in Proceedings of the 35th Annual ACM Symposium on the Theory of Computing, 2003, pp. 426–437.

[12] I. Damgård, T. Pedersen, and B. Pfitzmann, *On the existence of statistically hiding bit commitment schemes and fail-stop signatures*, in Proceedings of CRYPTO 1993, pp. 250–265.

[13] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai, *Robust non-interactive zero knowledge*, in Proceedings of CRYPTO 2001, pp. 566–598.

[14] G. Di Crescenzo, J. Katz, R. Ostrovsky, and A. Smith, *Efficient and non-interactive non-malleable commitment*, in Proceedings of EUROCRYPT 2001, pp. 40–59.

[15] G. Di Crescenzo, Y. Ishai, and R. Ostrovsky, *Non-interactive and non-malleable commitment*, in Proceedings of the 30th Annual ACM Symposium on the Theory of Computing, 1998, pp. 141–150.

[16] D. Dolev, C. Dwork, and M. Naor, *Nonmalleable cryptography*, SIAM J. Comput., 30 (2000), pp. 391–437.

[17] U. Feige, D. Lapidot, and A. Shamir, *Multiple noninteractive zero knowledge proofs under general assumptions*, SIAM J. Comput., 29 (1999), pp. 1–28.

[18] U. Feige and A. Shamir, *Witness indistinguishability and witness hiding protocols*, in Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing, 1990, pp. 416–426.

[19] M. Fischlin and R. Fischlin, *Efficient non-malleable commitment schemes*, in Proceedings of CRYPTO 2000, pp. 413–431.

[20] O. Goldreich, *Foundation of Cryptography: Basic Tools*, Cambridge University Press, London, 2001.

[21] O. Goldreich and A. Kahan, *How to construct constant-round zero-knowledge proof systems for NP*, J. Cryptology, 9 (1996), pp. 167–189.

[22] O. Goldreich and Y. Lindell, *Session-key generation using human passwords only*, J. Cryptology, 19 (2006), pp. 241–340.

[23] O. Goldreich, S. Micali, and A. Wigderson, *Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems*, J. ACM, 38 (1991), pp. 691–729.

[24] O. Goldreich, S. Micali, and A. Wigderson, *How to play any mental game—A completeness theorem for protocols with honest majority*, in Proceedings of the 19th Annual ACM Symposium on the Theory of Computing, 1987, pp. 218–229.

[25] O. Goldreich and Y. Oren, *Definitions and properties of zero-knowledge proof systems*, J. Cryptology, 7 (1994), pp. 1–32.

[26] S. Goldwasser and S. Micali, *Probabilistic encryption*, J. Comput. System Sci., 28 (1984), pp. 270–299.

[27] S. Goldwasser, S. Micali, and C. Rackoff, *The knowledge complexity of interactive proof systems*, SIAM J. Comput., 18 (1989), pp. 186–208.

[28] I. Haitner and O. Reingold, *Statistically-hiding commitment from any one-way function*, in Proceedings of the 38th Annual ACM Symposium on the Theory of Computing, 2007, pp. 1–10.

[29] S. Halevi and S. Micali, *Practical and provably-secure commitment schemes from collision-free hashing*, in Proceedings of CRYPTO 1996, Lecture Notes in Comput. Sci. 1109, Springer, New York, 1996, pp. 201–215.

[30] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby, *A pseudorandom generator from any one-way function*, SIAM J. Comput., 28 (1999), pp. 1364–1396.

[31] J. Katz, R. Ostrovsky, and M. Yung, *Efficient password-authenticated key exchange using human-memorable passwords*, in Proceedings of EUROCRYPT 2001, pp. 475–494.

[32] J. Kilian, *A note on efficient zero-knowledge proofs and arguments*, in Proceedings of the 24th Annual ACM Symposium on the Theory of Computing, 1992, pp. 723–732.

[33] Y. Lindell, *Bounded-concurrent secure two-party computation without setup assumptions*, Chic. J. Theoret. Comput. Sci., (2006), article 1.

[34] S. Micali, *Computationally sound proofs*, SIAM J. Comput., 30 (2000), pp. 1253–1298.

[35] M. Naor, *Bit commitment using pseudorandomness*, J. Cryptology, 4 (1991), pp. 151–158.

[36] M. Nguyen, S. Ong, and S. Vadhan, *Statistical zero-knowledge arguments for NP from any one-way function*, in Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science, 2006, pp. 3–14.

[37] M. Naor, R. Ostrovsky, R. Venkatesan, and M. Yung, *Perfect zero-knowledge arguments for NP using any one-way permutation*, J. Cryptology, 11 (1998), pp. 87–108.

[38] M. Naor and M. Yung, *Universal one-way hash functions and their cryptographic applications*, in Proceedings of the 21st Annual ACM Symposium on the Theory of Computing, 1989, pp. 33–43.

[39] M. Nguyen and S. Vadhan, *Simpler session-key generation from short random passwords*, in Proceedings of the 1st Theory of Cryptology Conference, Lecture Notes in Comput. Sci. 2951, Springer, New York, 2004, pp. 428–445.

[40] R. Pass, *Bounded-concurrent secure multi-party computation with a dishonest majority*, in Proceedings of the 36th Annual ACM Symposium on the Theory of Computing, 2004, pp. 232–241.

[41] R. Pass and A. Rosen, *Bounded-concurrent secure two-party computation in a constant number of rounds*, in Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science, 2003, pp. 404–413.

[42] R. Pass and A. Rosen, *New and improved constructions of non-malleable cryptographic protocols*, SIAM J. Comput., to appear.

[43] R. Pass, A. Shelat, and V. Vaikuntanathan, *Relations among notions of non-malleability for encryption schemes*, in Proceedings of AsiaCrypt'07, 2007, to appear.

[44] A. Sahai, *Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security*, in Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science, 1999, pp. 543–553.

SIAM J. COMPUT.
Vol. 37, No. 6, pp. 1926–1952

A LINEAR-TIME APPROXIMATION SCHEME FOR TSP IN UNDIRECTED PLANAR GRAPHS WITH EDGE-WEIGHTS*

PHILIP N. KLEIN[†]

Abstract. We give an algorithm requiring $O(c^{1/\epsilon^2} n)$ time to find an ϵ-optimal traveling salesman tour in the shortest-path metric defined by an undirected planar graph with nonnegative edge-lengths. For the case of all lengths equal to 1, the time required is $O(c^{1/\epsilon} n)$.

Key words. algorithms, combinatorial optimization, graphs, planar graphs, approximation algorithm, traveling salesman

AMS subject classifications. 68W25, 68R10

DOI. 10.1137/060649562

1. Introduction.

The traveling salesman problem (TSP) is often the first problem researchers use to test a new optimization technique [32]. In a metric space, a *tour* is a cycle $(v_0\ v_2\ \ldots\ v_{n-1})$ of the points of the metric space, and the weight of the tour is the sum $\sum_{i=0}^{n} \mathrm{dist}(v_i, v_{(i+1) \bmod n})$, where $\mathrm{dist}(u, v)$ is the distance between u and v. The goal is to find the minimum-weight tour. The problem is MAXSNP-hard [36, 37] in arbitrary metric spaces, and the best approximation ratio known, that proved by Christofides [14], is 1.5. For the shortest-path metric of an unweighted planar graph (one in which every edge has weight one), Grigni, Koutsoupias, and Papadimitriou [23] gave an algorithm that requires $n^{O(1/\epsilon)}$ to find a $1+\epsilon$-optimal tour. Thus for fixed ϵ, the algorithm runs in polynomial time. Such a family of polynomial-time algorithms is called an *approximation scheme.*

Arora et al. [5] subsequently gave a polynomial-time approximation scheme (PTAS) for the more general problem in which the planar graph's edges have arbitrary nonnegative weights. Their algorithm requires $n^{O(\epsilon^{-2})}$ time. Both algorithms are somewhat complicated and involve a recursive decomposition using new planar-separator lemmata. The latter paper introduced the idea of using a *spanner* result to handle edge-weights.

Arora [3] and Mitchell [34] had shown that a PTAS exists for *Euclidean TSP* (i.e., the subcase in which the points lie in $\Re^2$ and distance is measured using the Euclidean metric). This PTAS finds an ϵ-optimal tour in $n^{O(1/\epsilon)}$ time. Arora [4, 2] improved the running time of his algorithm to $O(n \cdot (\log n)^{O(1/\epsilon)})$, using randomization. Finally, Rao and Smith [38] gave a PTAS for the two-dimensional Euclidean case that takes time $O(\epsilon^{-O(\epsilon)} n + n \log n)$. (Their algorithm also used a spanner result.) The latter two approximation schemes are said to be *efficient* polynomial-time approximation schemes (EPTAS) because the time can be bounded by a function of ϵ times a polynomial function of n. Thus for an EPTAS, the degree of the polynomial does not grow with $1/\epsilon$.

In view of the fact that an ϵ-optimal tour can be found in the Euclidean case in time that is polynomial with a fixed degree, independent of ϵ, it seems natural to

*Received by the editors January 11, 2006; accepted for publication (in revised form) December 12, 2007; published electronically March 26, 2008. A preliminary version of this paper was published in *Proceedings of the IEEE Symposium on Foundations of Computer Science*, 2005, pp. 647–656.
http://www.siam.org/journals/sicomp/37-6/64956.html

[†]Department of Computer Science, Brown University, Providence, RI 02912 (klein@cs.brown.edu).

ask whether the same holds true for the planar case. In this paper, we answer this question.

THEOREM 1.1. *There is an algorithm that, for any $\epsilon > 0$ and any planar graph G with nonnegative edge-weights, finds a $1 + \epsilon$-optimal tour. The running time is $O(c^{1/\epsilon^2} n)$, where c is a constant. For the special case where all weights are 1, a similar algorithm requires $O(c^{1/\epsilon} n)$ time.*

Marx [33] subsequently showed that the running time for the unit-weight case is essentially optimal under a widely held complexity assumption.

1.1. Other related work. In a seminal paper, Baker [6] gives a method for obtaining PTASs for a variety of optimization problems in planar graphs, e.g., maximum-weight independent set and minimum-weight vertex cover. The resulting algorithms are linear time (for fixed ϵ). The key idea (interpreted in modern parlance) is to turn a problem in a planar graph into a problem in a graph with bounded treewidth.

Grigni and Sissokho ([24], building on [25]) have given a quasi-polynomial approximation scheme for weighted TSP in minor-excluded graphs. This paper proved a spanner result for minor-excluded graphs. Berger et al. ([7], building on [16]) give a PTAS for the problem of finding a minimum-weight 2-edge-connected spanning multisubgraph[1] of an edge-weighted planar graph, and a quasi-polynomial approximation scheme for finding a minimum-weight 2-edge-connected or biconnected spanning subgraph[2] of an edge-weighted planar graph. This paper introduced a new spanner construction.

Demaine and Hajiaghayi [17] describe a framework for PTASs that is based on the notion of *bidimensionality*. They derive approximation schemes for subclasses of minor-excluded graphs that involve turning the input graph into a low-treewidth graph. Their results apply to graphs that are not planar. Their framework can be viewed as a way to generalize Baker's approach so as to derive algorithms for nonlocal problems, such as feedback vertex set and connected dominating set. For planar graphs in particular, they derive EPTASs for several unit-weight problems. In relation to their framework, our result is an example of how one can more thoroughly exploit planarity to derive a fast and simple EPTAS.

For a positive number s, an s-spanner of a graph G is a subgraph of G that approximately preserves the node-to-node distances of G: for any pair u, v of nodes of G, the distance in the subgraph must be at most s times the distance in G. There is a vast literature on spanner constructions. In this paper, we require a construction for $1 + \epsilon$-spanners of planar graphs. Henceforth, for brevity we use the term *spanner* and omit mention of the parameter $1 + \epsilon$.

1.2. The approach. The TSP approximation scheme consists of the following steps.

Spanner step: Delete some edges of the input graph while approximately preserving the optimal value.[3]

Slicing step: Using breadth-first search in the planar dual together with a shifting argument, identify subgraphs (called *slices*). The weight of edges belonging to more than one slice is at most $1/k$ times the weight of the graph, and each connected

[1] Duplicate edges of the input graph are allowed in the solution.

[2] No duplicates are allowed.

[3] This was the also the first step in [5] and subsequently in, e.g., [24] and one of the algorithms of [7].

component of each slice has a spanning tree of depth at most $k + 1$, where k is a parameter.

Dynamic-programming step: Use dynamic programming to find an optimal solution in each connected component of each slice.

Combining step: The union of the tours found in the previous step comprises a tour for the original graph.

The time required by the dynamic-programming step is exponential in k. We show that there is a choice of k that depends only on ϵ for which the resulting tour is nearly optimal.

In the preliminary version of this paper [30], a slightly different algorithm was described. In the second step, a procedure called *thinning* was applied to the planar dual of the graph. Thinning involves deleting edges; thinning the planar dual corresponds to *contracting* edges in the primal. Thinning in either the primal or the dual results in a graph with small branch-width. The method of thinning in the dual graph is novel, though quite simple. One nice way to formulate the result is as follows:

> For any positive integer k, there is a partition of the edges of a planar graph into k sets such that contracting the edges in any one of the sets yields a graph with bounded treewidth (where the bound depends on k).

This formulation of the result is due to Demaine, Hajiaghayi, and Mohar [19], who learned of this result from the preliminary version of this paper and subsequently generalized the result to apply to graphs of any bounded genus.

Because of the potential applicability of the planar result and of its role in subsequent developments, we provide a proof in section 7.

The approach used in this version of the paper to formulate the TSP approximation scheme, which we call *slicing*, emerged from joint work with Borradaile and Mathieu [10, 11]. This formulation does not require the algorithm to perform any contractions, which leads to a simpler algorithm.

The general approach used for TSP has proved useful in obtaining approximation schemes for other problems in planar graphs, including minimum-weight 2-edge-connected spanning multisubgraph,[4] TSP on a subset of the nodes [31], minimum-weight 2-edge-connected spanning subgraph [8], and Steiner tree [10]. As mentioned above, the basic technique has been generalized [19] to apply to bounded-genus graphs, giving rise to new approximation schemes for such graphs.

1.3. Spanner step. The spanner step requires an algorithm that, given an n-node planar graph G_0 with edge-weights and given a parameter ϵ, deletes edges so as to obtain a graph G such that

S1: $\mathit{weight}(G) \leq \rho_\epsilon \cdot \mathrm{OPT}(G_0)$,

S2: $\mathrm{OPT}(G) \leq (1+\epsilon)\mathrm{OPT}(G_0)$, and

where $\mathrm{OPT}(G)$ is the value of the optimum for input graph G, and $\mathit{weight}(G)$ is the sum of weights of edges in G.

We refer to the first step as *spanner step* because of the connection to s-spanners. An *s-spanner* of a graph G_0 is a subgraph G of G_0 with the same set of nodes, such that, for any pair u, v of nodes, the u-to-v distance in G is at most s times the u-to-v distance in G_0. As discussed in Lemma 3.2, to achieve property S2 in the case of

[4] An $O(c^{1/\epsilon}n)$ algorithm for this problem can be obtained from the TSP algorithm by modifying the dynamic program.

TSP, it suffices that G be a $1+\epsilon$-spanner of G_0. In section 3, we discuss a spanner construction that also achieves property S1.

An n-node planar graph G_0 with no parallel edges or self-loops has at most $3n$ edges. For unit-weight edges, $\mathrm{OPT}(G_0)$ is at least n, so $\mathit{weight}(G_0) \leq \rho_\epsilon \mathrm{OPT}(G_0)$ holds for $\rho_\epsilon{=}3$. In this sense, a trivial spanner result suffices for the unit-weight case.

We remark that properties S1 and S2 can be considered for optimization problems other than TSP, and indeed for problems where a traditional s-spanner would not suffice. We propose use of the term *spanner result* to refer more generally to a construction achieving properties S1 and S2. We have obtained such constructions for two other problems in planar graphs, leading to approximation schemes for these problems. The first problem [31] is a generalization of the problem studied here; the tour must visit a specified subset of nodes of the input graph (not necessarily all the nodes). The second problem [10] is Steiner tree, in which one seeks a minimum-weight tree spanning a specified subset of nodes.

2. Preliminaries.

In this section, we describe the basic definitions and results on planar embeddings and planar duals. Most of the material is standard in concept, but the notation may be unfamiliar, and we also introduce a variant of contraction that we call *compression*, and state some related results. In subsection 2.5, we give some definitions and results that help us reformulate the TSP.

For a rooted tree T and a node v that is not the root of T, the *parent edge* of v is the edge of T that connects v to its parent.

2.1. Combinatorial embeddings.

The traditional geometric definition of planar embeddings involves drawings of a graph on the plane. Proofs and algorithms become simpler when one uses an alternative definition of embedded planar graphs, a combinatorial definition. See [35].

The idea of a combinatorial embedding was implicit in the work of Heffter [27]. Edmonds [21] first made the idea explicit, and Youngs [46] formalized the idea. A combinatorial embedding is sometimes called a *rotation system*. The idea is to represent at each node the arrangement of edges around that node, as illustrated in Figure 1.

However, it is convenient to represent at each node not just which edges are incident to the node and in what order but more specifically which ends of which edges are incident to the node. For example, if e is a self-loop (an edge whose endpoints are the same), the edge e would appear twice in the arrangement of incident edges, and it is helpful to be able to distinguish these two occurrences. We will refer to the ends of an edge as its *darts*, as we explain next.

For any given finite set E, we can interpret E as a set of edges, and we define $E \times \{\pm 1\}$ to be the corresponding set of *darts*. For each edge e, the darts of e, namely $\langle e, 1\rangle$ and $\langle e, -1\rangle$, represent the two opposite orientations of e. The edge of $\langle e, i\rangle$ is e. We define $\mathrm{rev}(\cdot)$ (*rev* is short for *reverse*) to be the function that takes each dart to the corresponding dart in the opposite direction: $\mathrm{rev}(\langle e, i\rangle) = \langle e, -i\rangle$.

We define an embedded graph on E to be a pair $G = \langle \pi, E\rangle$, where π is a permutation of the darts of E. The permutation cycles of π are called the *nodes* of G. Note that nodes are defined in terms of edges, rather than the other way around. This definition precludes isolated nodes. Each node v is a permutation cycle $(d_1\ d_2\ \ldots\ d_k)$.

For a graph G, we use $V(G)$, $E(G)$, and $D(G)$ to denote the node-set, the edge-set, and the dart-set of G. We use the same notation for subgraphs of G.

For a dart d of G, we define the tail of d in G, denoted $\mathrm{tail}_G(d)$, to be the permutation cycle of π containing d. (We may omit the subscript when doing so

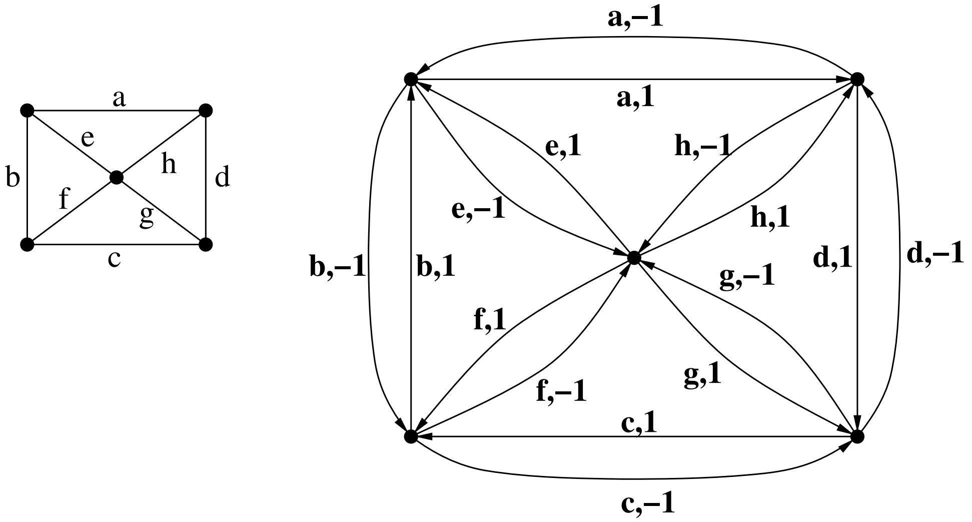


FIG. 1. *The figure on the left shows an undirected planar graph embedded in the plane, with its edges labeled. The* rotation *corresponding to the top left node is the permutation cycle* (a b e)*, indicating that the edges* a*,* b*, and* e *are incident to that node and are arranged counterclockwise around that node in the order* a*,* b*,* e*. Similarly, the rotation in the middle node is* (e f g h)*. The figure on the right shows the same undirected graph but with darts instead of edges. There are two oppositely directed darts for each (undirected) edge. The darts corresponding to edge* e *are* $\langle e, 1\rangle$ *and* $\langle e, -1\rangle$*. The rotation corresponding to a node consists of the darts that point away from the node. Thus the rotation corresponding to the top left node is* $(\langle a, 1\rangle\ \langle b, -1\rangle\ \langle e, -1\rangle)$*. The rotation corresponding to the middle node is* $(\langle e, 1\rangle\ \langle f, 1\rangle\ \langle g, 1\rangle\ \langle h, 1\rangle)$*.*

creates no ambiguity.) We define $\text{head}_G(d) = \text{tail}_G(\text{rev}(d))$. The tail and head of a dart d are called the *endpoints* of d and also the endpoints of the edge of d.

A *walk* of darts in G is a sequence $d_1 \ldots d_k$ of darts such that, for $i = 2, \ldots, k$, $\text{head}_G(d_{i-1}) = \text{tail}(d_i)$.[5] The *start* of the walk is $\text{tail}_G(d_1)$ and the *end* is $\text{head}_G(d_k)$. It is a *closed* walk if in addition $\text{head}_G(d_k) = \text{tail}_G(d_1)$. It is a simple path/cycle (cycle if closed, path if not) if no node occurs twice as the head of a dart. The walk, path, or cycle is said to contain an edge e if it contains a dart of e. It is said to contain a node v if v is the head or tail of some dart in the sequence. We define $\text{rev}(d_1 \ldots d_k) = \text{rev}(d_k) \ldots \text{rev}(d_1)$. A walk/path whose start is u and whose end is v is called a *u-to-v* walk/path.

To define the faces of the embedded graph, we define another permutation π^* of the set of darts by composing π with rev: $\pi^* = \pi \circ \text{rev}$. Then the *faces* of the embedded graph $\langle \pi, E\rangle$ are defined to be the permutation cycles of π^*. (See Figure 2.) Note that a face of G can be interpreted as a closed walk in G.

Note that this definition diverges from the traditional geometric definition of faces in the case of a disconnected graph. In that case, according to the definition considered here, for each connected component there will be a different external face. (In fact, this is necessary if one wishes to preserve the desirable property that the dual of the dual is the primal.)

[5] Note that, even though we are concerned with undirected graphs, we use (directed) darts in our definition of walks because they provide more information about the structure of the walks.

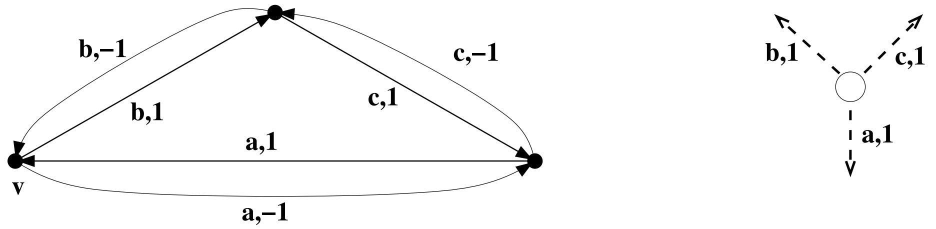


FIG. 2. *The figure on the left shows the dart representation of part of a graph. We can trace out the face containing the dart $\langle a, 1\rangle$ as follows. First apply rev, obtaining the dart $\langle a, -1\rangle$ emanating from the node v. Next, apply π, obtaining $\langle b, 1\rangle$, the dart after $\langle a, -1\rangle$ in the permutation cycle corresponding to v. This shows that $\langle b, 1\rangle$ is the successor to $\langle a, 1\rangle$ in the face. We apply the same process to $\langle b, 1\rangle$, obtaining its successor $\langle c, 1\rangle$ and that dart's successor in turn, $\langle a, 1\rangle$. The face (permutation cycle of $\pi \circ rev$) is thus $(\langle a, 1\rangle\ \langle b, 1\rangle\ \langle c, 1\rangle)$. The figure on the right shows the corresponding fragment of the dual graph. There is a dual node corresponding to the face discussed above. The permutation cycle corresponding to this dual node is exactly the permutation cycle comprising the face: $(\langle a, 1\rangle\ \langle b, 1\rangle\ \langle c, 1\rangle)$. However, we follow the convention of drawing the dual in such a way that the permutation cycle gives the* clockwise *order of darts. This convention helps when drawing the dual superimposed on the primal, for it enables us to draw primal and dual edges at approximately right angles to one another, as shown in Figure* 3. *The convention does not affect the dual graph as a mathematical object, only its depiction.*

2.2. Planarity. We say that an embedding π of a graph G is *planar* if it satisfies Euler's formula: $n - m + \phi = 2\kappa$, where n=number of nodes, m=number of edges, ϕ=number of faces, and κ=number of connected components. In this case, we say $G = \langle \pi, E\rangle$ is a *planar embedded graph.* We say a graph is a *planar graph* if there is a planar embedding for it.[6] Finding a planar embedding for a planar graph is a well-studied problem, and linear-time algorithms are known,[7] so we assume throughout this paper that every planar graph comes equipped with an embedding. It follows from Euler's formula that an n-node planar graph with no parallel edges has $O(n)$ edges.

2.3. Duality. The *dual* of a connected embedded graph $G = \langle \pi, E\rangle$ is defined to be the embedded graph $G^* = \langle \pi^*, E\rangle$. The permutation cycles of π^* are the faces of G. (See Figure 3.) According to this definition, the edge-set of the dual is identical to the edge-set of the original graph (called the *primal*). This identification of primal edges and dual edges is mathematically and notationally convenient (albeit sometimes confusing).

Since rev $\circ$ rev is the identity, $(\pi^*)^* = \pi$, we obtain the following.

PROPOSITION 2.1. *$G^{**} = G$.*

It can be shown that the dual of a connected graph is connected. It follows that the connected components of G^* correspond one-to-one with the connected components of G. Hence if G satisfies Euler's formula, then so does G^*. Thus the dual of a planar embedded graph is a planar embedded graph.[8]

[6] For the purpose of the current result, all we need is that every graph embeddable on an orientable surface of genus zero has a combinatorial embedding that satisfies Euler's formula. However, it is known (see, e.g., [35]) more generally that for any graph embedded on a closed, orientable surface, the corresponding combinatorial embedding determines the geometric embedding up to homeomorphism.

[7] The first was due to Hopcroft and Tarjan [29]. See [12] for a discussion of later work.

[8] For disconnected graphs, this definition of dual diverges from the geometric definition in that it assigns multiple dual nodes to a single region of the sphere/plane. According to the geometric definition, the dual of a graph is always connected. However, choosing that definition means giving up, for example, the nice property that $G^{**} = G$.

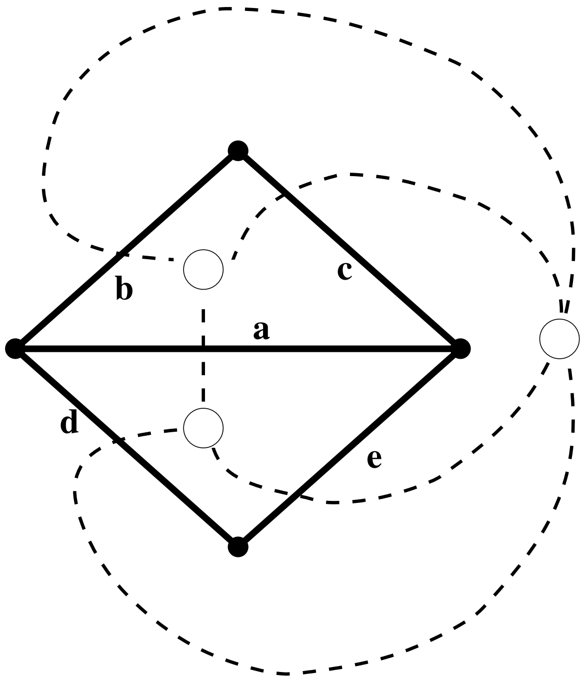


FIG. 3. *The figure shows a graph (the solid nodes and edges) and, superimposed, its planar dual (the open nodes and dashed edges).*

Let T be a spanning tree of G. For an edge $e \notin T$, there is a unique simple cycle consisting of e and the unique path in T between the endpoints of e. This cycle is called the *elementary cycle* of e with respect to T in G.

For a spanning tree T of G, we denote by T^* the set of edges of G that are not in T. The following is a classical result.

PROPOSITION 2.2. *If G is a planar embedded graph and T is a spanning tree of G, then T^* is a spanning tree of G^*.*

We refer to T^* as the tree dual to T.

If $S \subseteq V(G)$, we use $\Gamma_G(S)$ to denote the set of edges e such that in G the edge e has one endpoint in S and one endpoint not in S. A set of this form is called a *cut* of G. Note that $\Gamma_G(S) = \Gamma_G(V(G) - S)$.

If S is connected in G and $V(G) - S$ is connected in G, we call $\Gamma_G(S)$ a *bond*.

PROPOSITION 2.3. *If G is a planar embedded graph, the edges of a bond in G form a simple cycle in G^* and vice versa.*

It follows from Proposition 2.3 that every simple cycle C in G defines a bipartition of the faces of G; namely the bipartition $(S, V(G) - S)$ where $E(C) = \Gamma_{G^*}(S)$.

Let f_∞ be a face of G. We call f_∞ the *infinite face* by analogy to geometric embeddings. For combinatorial embeddings, the choice is arbitrary.[9]

We say the simple cycle C *encloses* a face f with respect to f_∞ if f belongs to the set S such that $E(C) = \Gamma_{G^*}(S)$ and $f_\infty \notin S$. We say that C encloses an edge with respect to f_∞ if the edge belongs to a face enclosed by C, and that it strictly

[9] For geometric intuition, consider that a planar graph can be embedded on the surface of a sphere. According to this embedding, every face is finite.

encloses the edge if in addition the edge does not belong to C.

LEMMA 2.4. *Let G be a connected planar embedded graph, let T be a rooted spanning tree of G, let v be a nonroot node of T, and let e be the parent edge of v. Then the elementary cycle of e in G^* with respect to T^* consists of the edges of Γ_G(descendents of v in T).*

Proof. Removing e from T breaks T into two connected components: one containing the descendents of v in T and one containing the nondescendents. It follows that the cut Γ_G(descendents of v in T) is a bond, and therefore, by Proposition 2.3, the edges in that cut form a simple cycle C in G^*. The only edge of $E(T)$ belonging to $E(C)$ is e, so C consists of e together with a simple path of edges not in $E(T)$ connecting the endpoints of e in G^*. The edges not in $E(T)$ are in $E(T^*)$, so the simple path is a simple path in T^*. This proves the lemma. □

2.4. Deletion and compression. We discuss two ways of removing edges from an embedded graph—deleting and compressing, both of which preserve the embedding (and preserve planarity). Compressing an edge is very similar to the operation of contracting the edge (the difference arises when the edge is a self-loop).

Deleting an edge e of an embedded graph $G = \langle \pi, E\rangle$ is an operation that produces the graph $G' = \langle \pi', E'\rangle$, where $E' = E - \{e\}$ and, for each dart of E',

$$\pi'[d] = \begin{cases} \pi[\pi[\pi[d]]] & \text{if } \pi[d] \text{ and } \pi[\pi[d]] \text{ are the darts of } e, \\ \pi[\pi[d]] & \text{if } \pi[d] \text{ is a dart of } e \text{ but } \pi[\pi[d]] \text{ is not,} \\ \pi[d] & \text{otherwise.} \end{cases}$$

For a set S of edges, we denote by $G - S$ the embedded graph obtained by deleting the edges of S. The order of deletion does not affect the final embedded graph. It is easy to see that deletion preserves planarity.

PROPOSITION 2.5. *An edge e is a self-loop of G iff it is a cut-edge of G^*.*

We define edge *compression* to be deletion in the dual. That is, compressing an edge e of G is an operation that produces the graph $(G^* - \{e\})^*$. We denote the result as $G/\{e\}$. Since deletion preserves planarity and the dual of a planar embedded graph is a plane graph, compression preserves planarity. The operations of deletion and compression commute.

Figure 4 illustrates the effect of edge compression on the underlying graph in three examples. If e is not a self-loop in G, then the effect of compressing e in G is to contract e as shown in the top left diagram. The thick line represents the edge to compress. If e is a self-loop in G and thus a cut-edge in G^* and is not the only edge incident to either of its endpoints in G^*, then the effect is to duplicate v, as shown in the bottom left diagram; one copy has as its incident edges those edges that in G are incident to v and strictly enclosed by e (with respect to some designated face f_∞) and the other copy has as its incident edges those edges that in G are incident to v and not enclosed by e (and not equal to e). If e is a self-loop in G and is the only edge incident to one of its endpoints in G^*, the effect is to delete e.

2.5. Preliminaries related to TSP. For an assignment $weight(\cdot)$ of nonnegative weights to the edges of G and a set S of edges, define $weight(S) = \sum\{weight(e) : e \in S\}$. For a subgraph H, define $weight(H) = weight(E(H))$.

For the metric space of shortest paths in a graph, a tour corresponds to a closed walk in the graph that visits every node. The weight of the tour is the sum of weights of the edges comprising the walk, counting multiplicities. For a connected graph G, let $\mathrm{OPT}(G, weight)$ be the minimum weight of such a tour. (We omit the second argument when doing so creates no ambiguity.)

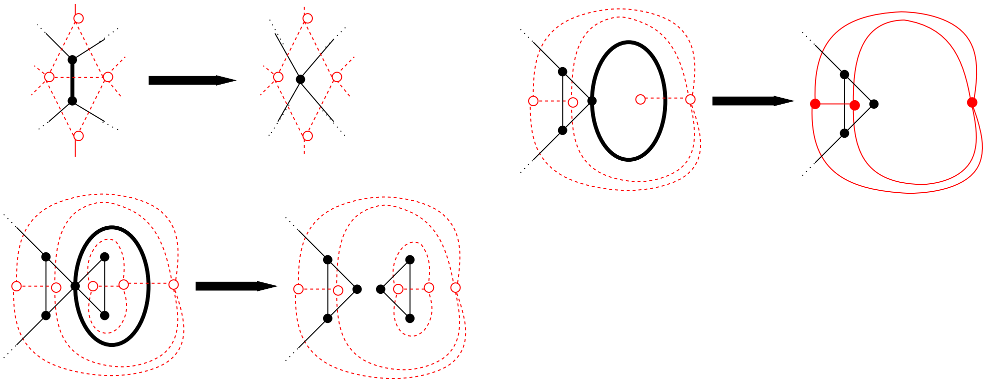


FIG. 4. *Three examples of compression. The graph with solid edges and solid nodes is the primal. The graph with dashed edges and open nodes is the dual. The edge being compressed is signified by a heavy line.*

LEMMA 2.6. *For any walk W in a graph, there is a walk W' that visits the same nodes as W, such that every edge used by W' is used by W and occurs at most twice in W'.*

Proof. Let W be a closed walk in G, and suppose some dart d occurs at least twice in W. Write $W = W_1\ d\ W_2\ d$. Then $W_1\ \mathrm{rev}(W_2)$ is a closed walk of G that visits the same nodes as W but uses dart d fewer times. Repeating this step yields the lemma. □

Lemma 2.6 shows that, in seeking the minimum-weight walk visiting a given set of nodes, we can restrict ourselves to considering walks in which each edge occurs at most twice.

Let W be a walk, and let $P = a\ W\ b$ and $Q = c\ W\ d$ be walks that are identical except for their first and last darts. Let c' be the successor of c in Q and let d' be the predecessor of d in Q. We say Q forms a *crossing configuration* with P (see Figure 5) if the permutation cycle at $\mathrm{head}(c)$ induces the cycle $(c\ c'\ \mathrm{rev}(a))$ and the permutation cycle at $\mathrm{tail}(d)$ induces the cycle $(\mathrm{rev}(d')\ b\ d)$.

We say a walk P *crosses* a walk Q if a subwalk of P and a subwalk of Q form a crossing configuration. The following folklore result was used by Arora et al. in [5].

PROPOSITION 2.7. *For any tour in a planar graph, there exists a tour that visits the same nodes and comprises the same darts in the same multiplicities, and does not cross itself.*

Proof. Suppose $\hat{W} = W_1\ a\ W\ b\ W_2\ c\ W\ d$ is a closed walk where $c\ W\ d$ forms a crossing configuration with $a\ W\ b$. Then $d\ W_1\ a\ \mathrm{rev}(c\ W_2\ b)\ \mathrm{rev}(W)\ W$ is a closed walk visiting the same nodes and comprising the same darts in the same multiplicities, and with one fewer crossing configurations. □

Proposition 2.7 shows that we can restrict our attention to non-self-crossing walks.

An *Eulerian graph* is a graph G with the following properties:

- G is connected, and
- every node of G has even degree.

Perhaps the best-known result in graph theory is the following.

PROPOSITION 2.8. *A graph G is Eulerian iff there is a walk in G in which each edge occurs exactly once.*

Such a walk is called an Eulerian cycle. There is a linear-time algorithm that, given an Eulerian graph, finds an Eulerian cycle.

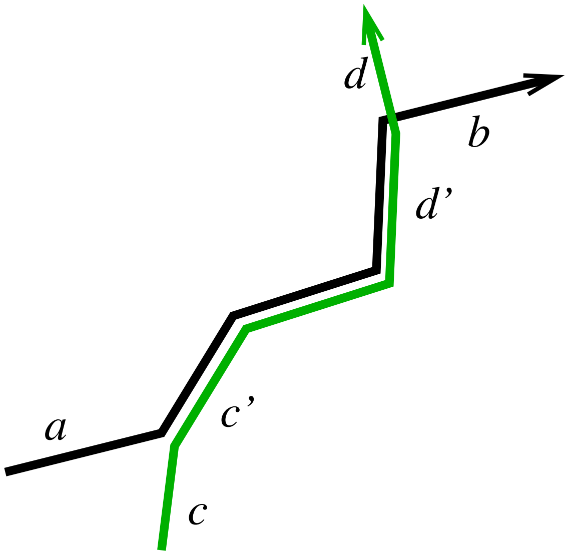


FIG. 5. *The light walk forms a crossing configuration with the bold walk.*

A graph H is a *multisubgraph* of G if H can be obtained from a subgraph of G by duplicating some edges. We call it a *bisubgraph* if the maximum multiplicity of any edge is at most two.

It follows from the Eulerian characterization that finding a minimum-weight tour in a graph G is equivalent to finding a minimum-weight Eulerian multisubgraph of G that includes every node of G. Lemma 2.6 shows that furthermore it suffices to find a minimum-weight Eulerian bisubgraph that includes every node.

We slightly generalize the notion of Eulerian multisubgraph to handle disconnected graphs. For a possibly disconnected graph G, we say H is a *multi-Eulerian multisubgraph* of G if for each connected component K of G there is a connected component of H that is an Eulerian multisubgraph of K. For a disconnected graph, define $\mathrm{OPT}(G, weight)$ to be the sum over connected components K of $\mathrm{OPT}(K, weight)$. Then $\mathrm{OPT}(G, weight)$ is the minimum weight of a multi-Eulerian multisubgraph of G.

3. Spanner.

Althöfer et al. [1] considered a simple and general procedure for producing a spanner in a (not necessarily planar) graph G_0: start with an empty graph G, consider the edges of G_0 in increasing order of weight, and add an edge to G if the edge's weight was much smaller than the minimum-weight path in G_0 between its endpoints. They did not address the exact running time of the procedure, but it clearly consists of $O(n)$ iterations, each involving a shortest-path computation. For planar graphs, therefore, it runs in $O(n^2)$ time [28]. They proved several results about the size and weight of the resulting spanner, including the following result that is specific to planar graphs.

THEOREM 3.1 (Althöfer et al.). *For any planar graph G_0 with edge-weights and any $\epsilon > 0$, there is an edge subgraph G such that*

A1: *$weight(G) \leq (1+2\epsilon^{-1})MST(G_0)$, where $MST(G_0)$ is the weight of the minimum spanning tree of G_0, and*

A2: *for every pair of nodes u and v,*

$$\begin{aligned}&\textit{minimum weight of a } u\textit{-to-}v \textit{ path in } G\\ &\le (1+\epsilon)\cdot \textit{minimum weight of a } u\textit{-to-}v \textit{ path in } G_0.\end{aligned}$$

Lemma 3.2. *Properties* A1 *and* A2 *imply properties* S1 *and* S2 *of section* 1.3 *with* $\rho_\epsilon = 1 + 2\epsilon^{-1}$.

Proof. Because a tour includes a spanning tree, $MST(G) \le \text{OPT}(G)$. Hence property A1 implies that property S1 of section 1.3 is achieved with $\rho_\epsilon = 1 + 2\epsilon^{-1}$.

Now we show that property A2 implies property S2, i.e., that $\text{OPT}(G) \le (1 + \epsilon_0)\text{OPT}(G_0)$. (This argument was used in [5].) Let T_0 be an optimal tour of G_0. For each edge uv of T_0 that is not in G, there is a u-to-v path in G of weight at most $(1+\epsilon)weight(uv)$; replace uv in T_0 with that path. The result of all the replacements is a tour T_1 whose weight is at most $1+\epsilon$ times that of T_0. This shows $\text{OPT}(G) \le (1+\epsilon)\ \text{OPT}(G_0)$. □

By exploiting planarity, we can give an algorithm that runs in linear time but that can be shown (using the same analysis technique used by Althöfer et al.) to achieve the same properties.

Theorem 3.3. *There is a linear-time algorithm that, given a planar graph G_0 with edge-weights and any $\epsilon > 0$, outputs an edge subgraph G with properties* A1 *and* A2.

The algorithm is as follows. (Refer to Figure 6.)

```
define SPANNER($G_0, \epsilon$):
    let $x[\cdot]$ be an array of numbers, indexed by edges
    find a minimum spanning tree $T$ of $G_0$
    assign $x[e] := weight(e)$ for each edge $e$ of $T$
    initialize $S := \{$edges of $T\}$
    let $T^*$ be the dual tree, rooted at the infinite face
    for each edge $e$ of $T^*$, in order from leaves to root
      let $f_e$ be the face of $G_0$ whose parent edge in $T^*$ is $e$
      let $e=e_0, e_1, \ldots, e_s$ be the sequence of edges comprising $f_e$
      $x_{\text{omit}} := \sum_{i=1}^{s} x[e_i]$
      if $x_{\text{omit}} > (1+\epsilon)weight(e)$
        then add $e$ to $S$ and assign $x[e] := weight(e)$
        else assign $x[e] := x_{\text{omit}}$
    return $S$
```

The minimum spanning tree of G_0 can be found in linear time using the algorithm of Cheriton and Tarjan [13].

Now we address correctness of the procedure. Say an edge e is *accepted* when e is added to S, and *rejected* if e is considered but not added to S.

Lemma 3.4. *In the for-loop iteration in which e is considered, for every other edge e_i of f_e, $x[e_i]$ has been assigned a number.*

Proof. The face f_e has only one parent edge in T^*, and it is e. For every other edge e_i of f_e, either e_i belongs to T or e_i is a child edge of f_e in T^*. □

For any edge e of G_0 not in T,

- let $\hat{G}_e$ denote the subgraph of G_0 consisting of accepted edges together with e,
- let $\hat{f}_e$ denote the face of $\hat{G}_e$ that contains e and encloses f_e,
- let $\hat{W}_e$ denote the walk formed by the sequence of edges comprising $\hat{f}_e$ not including e itself, and

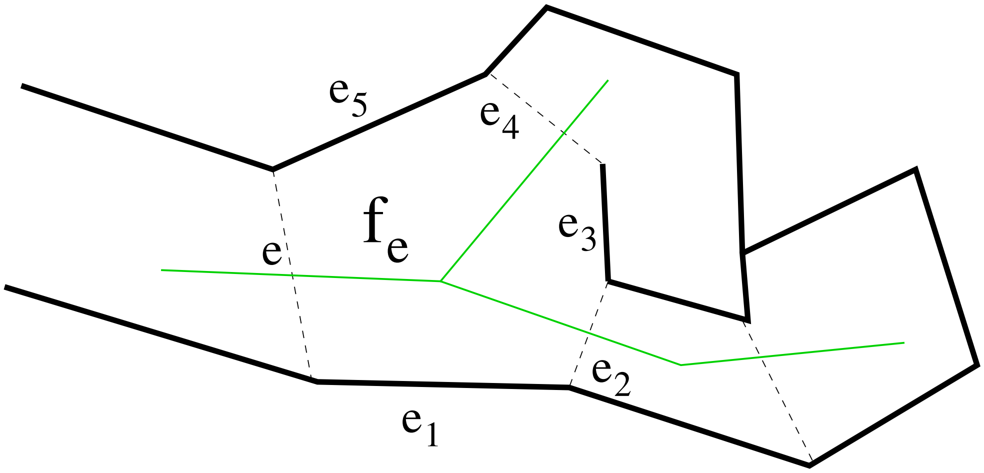


FIG. 6. *Diagram showing part of dual tree (in light edges) and primal tree (in dark edges) and primal nontree edges (dashed): e_2 and e_4 are child edges of e in the dual tree. The face f_e is indicated.*

- let

$$P_e = \begin{cases} e & \text{if } e \text{ is accepted,} \\ \hat{W}_e & \text{otherwise.} \end{cases}$$

Note that each of $\hat{W}_e$ and P_e has the same endpoints as e. For an edge e of T, define $P_e = e$. The basic argument of the following lemma comes from [1].

LEMMA 3.5. *For any edge e of G_0 not in T,*

1. *every edge of $\hat{f}_e$ is either in T or is a descendent of e in T^*, and*
2. *$\hat{W}_e = P_{e_1} \cdots P_{e_s}$, where $e_1 \ldots e_s$ is the walk consisting of the edges comprising f_e other than e.*

Proof. The proof is by induction. Consider the case in which e is a leaf-edge of T^*. Let f be the corresponding leaf-node in G_0^*. Because f is a leaf, the only incident edge that is in T^* is e itself, so $e_1, \ldots, e_s$ belong to T. All these edges are accepted, proving part 1. To prove part 2, note that $W_e = e_1 \cdots e_s$ and that $P_{e_i} = e_i$ for $i = 1, \ldots, s$. Thus the lemma holds for e.

Consider the case where e is not a leaf. Let $\hat{G}_{e+}$ be the subgraph of G_0 consisting of accepted edges together with $e, e_1, \ldots, e_s$. For each e_i, recall that $\hat{f}_{e_i}$ is the face of $\hat{G}_{e_i}$ that contains e_i and encloses f_{e_i}. We claim that $\hat{f}_{e_i}$ is also a face of $\hat{G}_{e+}$. To prove the claim, note that $\hat{G}_{e_i}$ can be obtained from $\hat{G}_{e+}$ by deleting a subset of $\{e, e_1, \ldots, e_s\} - \{e_i\}$. None of these edges are edges of T or descendents in T^* of e_i, so, by part 1 of the inductive hypothesis, none belongs to $\hat{f}_{e_i}$.

Note that $\hat{G}_e$ can be obtained from $\hat{G}_{e+}$ by deleting those edges among $e_1, \ldots, e_s$ that are rejected. By the claim, each such deletion replaces a rejected edge e_i in f_e with the walk $\hat{W}_{e_i}$. This, together with the definition of P_{e_i}, proves part 2. By part 1 of the inductive hypothesis, every edge in each $\hat{W}_{e_i}$ is an edge of T or a descendent of e_i in T and hence a descendent of e as well. This proves part 1. □

LEMMA 3.6. *In the for-loop iteration that considers e,*

- *the value assigned to x_{omit} is weight$(\hat{W}_e)$, and*
- *the value assigned to $x[e]$ is weight(P_e).*

Proof. The proof is by induction. By Lemma 3.4, the edges $e_1, \dots, e_s$ are considered before e. By the inductive hypothesis, $x[e_i] = weight(P_e)$. By Lemma 3.5, $weight(\hat{W}_e) = \sum_{i=1}^{s} x[e_i]$, which proves the first statement. The second statement follows by definition of P_e. ∎

COROLLARY 3.7. *For each edge e, $weight(P_e) \leq (1+\epsilon)weight(e)$.*

Proof. If e is accepted, $P_e = e$ so the statement holds trivially. Suppose e is rejected. By the conditional in the algorithm, in the iteration considering e, the value assigned to x_{omit} was at most $(1+\epsilon)weight(e)$. By the first part of Lemma 3.6, $weight(\hat{W}_e)$ and therefore $weight(P_e)$ are at most $(1+\epsilon)weight(e)$. ∎

COROLLARY 3.8. *The graph of accepted edges satisfies property* A2.

Proof. For any pair of nodes u and v, let P be the shortest u-to-v path in G_0. For each edge e of P, there is a walk P_e consisting of accepted edges between the endpoints of e. By Corollary 3.7, $weight(P_e) \leq (1+\epsilon)weight(e)$. Replacing each edge e of P with P_e therefore yields a walk of weight at most $\sum_{e \in P}(1+\epsilon)weight(e)$, which is at most $(1+\epsilon)weight(P)$. ∎

LEMMA 3.9. *At any time during the algorithm's execution, the weight of the infinite face in the graph consisting of accepted edges is at most*

$$2 \cdot MST(G_0) - \epsilon \cdot weight(\textit{accepted edges not in } T).$$

Proof. The proof is by induction. Before the for-loop commences, the graph of accepted edges is T, the minimum spanning tree of G_0. Hence the weight of the infinite face is exactly $2 \cdot MST(G_0)$, so the lemma's statement holds for this time. Consider a for-loop iteration, and let e be the edge being considered. If e is not accepted, there is no change to the set of accepted edges, so the lemma's statement continues to hold.

Suppose e is accepted. Let G_{after} be the subgraph consisting of edges accepted so far, and let $G_{\text{before}} = G_{\text{after}} - \{e\}$. Note that G_{after} can be obtained from $\hat{G}_e$ by deleting edges that will be accepted in the future. By the leaves-to-root ordering, none of the deleted edges are descendents of e in T^*. By part 1 of Lemma 3.5, therefore, $\hat{f}_e$ is a face of G_{after}. Let g be the other face of G_{after} that contains e.

We claim that g is the infinite face of G_{after}. To prove the claim, note that G_{after} can be obtained from G_0 by deleting edges that have already been rejected and edges not yet considered. By the leaves-to-root ordering, e's proper ancestors in T^* have not yet been considered, so they are among the edges deleted. These deletions are contractions in the dual. The root of T^* is the infinite face, so the contractions result in g being the infinite face.

Note that G_{before} can be obtained from G_{after} by deleting e. This deletion replaces e in the face g with $\hat{W}_e$. This shows that

$$\begin{aligned}
&\text{weight of infinite face in } G_{\text{before}} - \text{weight of infinite face in } G_{\text{after}} \\
&= weight(\hat{W}_e) - weight(e) \\
&> (1+\epsilon)weight(e) - weight(e) \text{ because } e \text{ was accepted} \\
&= \epsilon \cdot weight(e),
\end{aligned}$$

which shows that the lemma's statement continues to hold. ∎

COROLLARY 3.10. *The graph G of accepted edges satisfies property* A1.

Proof. By Lemma 3.9, the weight of the infinite face in the graph consisting of all accepted edges is at most

$$2 \cdot MST(G_0) - \epsilon \cdot weight(\text{accepted edges not in } T),$$

so $weight$(accepted edges not in T) $\leq 2\epsilon^{-1}\cdot MST(G_0)$. Since $weight(T) = MST(G_0)$, it follows that the weight of all accepted edges is at most $(1 + 2\epsilon^{-1})MST(G_0)$. □

This completes the proof of Theorem 3.3.

4. Slices.

Let G be a connected planar embedded graph and let $weight(\cdot)$ be an edge-weight assignment. Let k be a parameter. Recall that G^* denotes the planar dual of G. Let f_∞ be the infinite face of G, which is a vertex of G^*. Define the *level* of a node v of G^* to be its breadth-first-search distance in G^* from f_∞, i.e., the minimum number of edges in an f_∞-to-v path in G^*. Define the level of an edge e to be ℓ if one endpoint has level ℓ and the other endpoint has level $\ell + 1$.

For $j = 0, 1, \ldots, k-1$, let S_j denote the set of edges e whose levels are congruent to j mod k. Let $t = \text{minarg}_j weight(S_j)$, and let $S = S_t$. We obtain the following bound:

$$weight(S) \leq (1/k)\, weight(G). \tag{1}$$

For $i = 0, 1, 2, \ldots$, let E_i be the set of edges e having at least one endpoint with level in the range $(t + (i-1)k, t + ik]$. We define *slice* i of G to be the subgraph of G (the primal graph) consisting of the edges E_i. Note that an edge of G belongs to two distinct slices only if the edge belongs to S.

The theorem below shows that the total weight of optimal tours in the slices exceeds the weight of the optimal tour of G by at most twice the weight of S.

Theorem 4.1. $\sum_i OPT(\textit{slice } i) \leq 2\, weight(S) + OPT(G)$.

The next theorem states that the planar dual of each slice has a low-depth spanning tree.

Theorem 4.2. *For $i = 0, 1, 2, \ldots$, each connected component of the planar dual of slice i has a rooted spanning tree of depth at most $k + 1$.*

In the rest of this section, we prove Theorems 4.1 and 4.2.

The following lemma is illustrated in Figure 7.

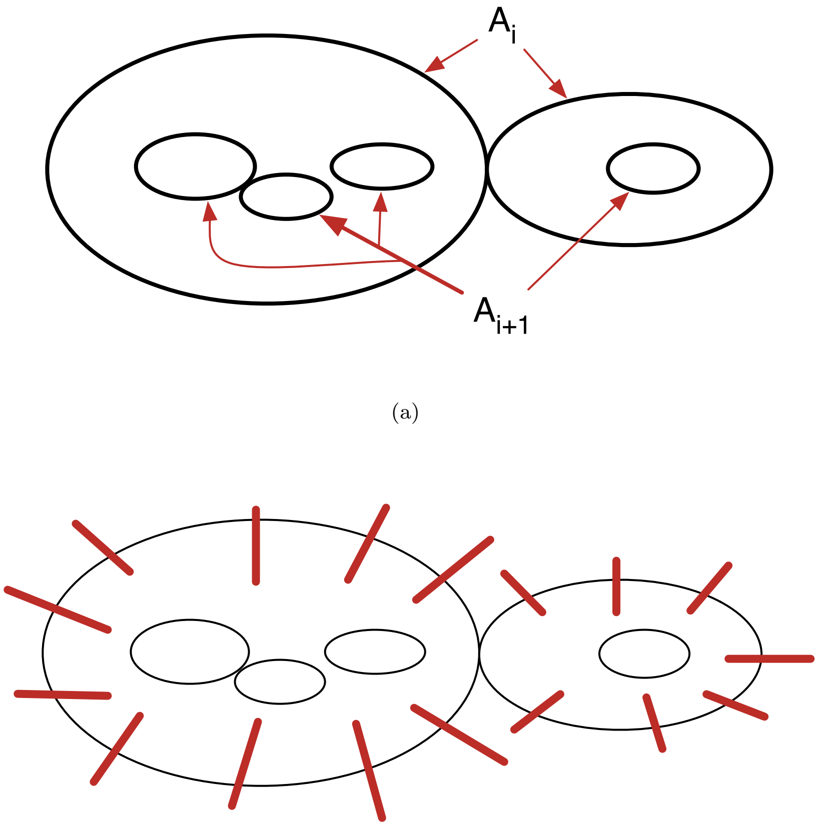
Lemma 4.3. *For $i = 1, 2, 3, \ldots$, the edges of level $t + (i-1)k$ form a set A_i of simple cycles in G with the following properties:*

1. *The cycles are edge-disjoint.*
2. *Every face is enclosed by at most one of the cycles.*
3. *A face u of G is enclosed by one of the cycles iff in the dual graph G^* the node u has level greater than $t + (i-1)k$.*

Proof. Let T be a breadth-first-search tree of G^* rooted at f_∞. For $i = 1, 2, \ldots$, let $\mathcal{K}_i$ be the set of connected components of the subgraph of G^* consisting of nodes whose levels exceed $t + (i-1)k$. Let $A_i = \{\Gamma_{G^*}(V(K)) \ : \ K \in \mathcal{K}_i\}$.

Let K be a connected component in $\mathcal{K}_i$. For any node v not in K, if v is not f_∞, then v has a parent p whose level is one less than that of v. If p's level is at most $t + (i-1)k$, then p is not in K; if p's level is greater than $t + (i-1)k$, then so is v's, so if p were in K, then v would also be in K, a contradiction. Thus p is not in K. By induction, G^* contains a v-to-f_∞ path that avoids K, proving that the nodes of G^* not in K are connected, so $\Gamma_{G^*}(V(K))$ is a bond. By Proposition 2.3, the edges of $\Gamma_{G^*}(V(K))$ form a simple cycle C_K in G. The faces enclosed by C_K are the nodes of K.

Consider two components $K_1, K_2 \in \mathcal{K}_i$. Since $V(K_1)$ and $V(K_2)$ are disjoint, the faces enclosed by C_{K_1} are disjoint from the faces enclosed by C_{K_2}. Furthermore, for $j = 1, 2$, an edge belongs to C_{K_j} if in G^* the edge connects a node of K_j to a node at level $t + (i-1)k$, which shows that C_{K_1} and C_{K_2} are edge-disjoint. □



(a)

(b)

Fig. 7. *Cycles in A_i and A_{i+1} are shown. Note that the cycles of A_{i+1} are enclosed within cycles of A_i. In the figure on the bottom, the dual edges corresponding to cycles in A_i are indicated by thick lines.*

Lemma 4.4. *Let $A_1, \ldots$ be the set of cycles from Lemma 4.3. For $i \geq 1$, an edge e of G belongs to slice i iff e is enclosed by some cycle in A_i and not strictly enclosed by any cycle in A_{i+1}. An edge e belongs to slice 0 iff e is not strictly enclosed by any cycle in A_1.*

Proof. By definitions of slice and dual, for $i \geq 1$, an edge e belongs to slice i iff e belongs to a face f whose level in G^* is in $(t+(i-1)k, t+ik]$. By Lemma 4.3, the level of f is in $(t+(i-1)k, t+ik]$ iff f is enclosed by a cycle in A_i and not by a cycle in A_{i+1}. The lemma follows by the definition of a cycle enclosing an edge. The case of slice 0 is similar. □

We say a subgraph is *even* if every node has even degree.

Lemma 4.5. *Let R be an Eulerian multisubgraph of G, let C be a simple cycle of G, and let X be the set of nodes enclosed by C. There is a subset $\widehat{C}$ of the edges of C such that*

1. *$weight(\widehat{C}) \leq \frac{1}{2} weight(C)$, and*

2. *each connected component of* $R - \Gamma_G(X) \cup \widehat{C}$ *is even.*

For a graph G, a node v, and a set A of edges, we define $\deg_G(v, A)$ to be the number of edges in A that in G are incident to v.

Proof. Because R is Eulerian, $\deg(v, R)$ is even for every node v, so $\sum\{\deg(v, R) : v \in V(C)\}$ is even. The closed walk (Eulerian cycle) corresponding to R enters X the same number of times as it leaves, so $|R \cap \Gamma_G(X)|$ is even. It follows that $\sum\{\deg(v, R - \Gamma_G(X)) : v \in V(C)\}$ is even. Hence the set

$$Y = \{v \in V(C) : \deg(v, R - \Gamma_G(X)) \text{ is odd}\}$$

has even cardinality.

Write $C = P_1 \ \dots \ P_{|Y|}$, where each P_i is a path whose endpoints belong to Y and whose internal nodes do not. Let $\widehat{C}$ denote the set of edges in $P_1, P_3, P_5, \dots, P_{|Y|-1}$ or the set of edges in $P_2, P_4, P_6, \dots, P_{|Y|}$, whichever has less weight. This choice ensures property 1 in the lemma's statement. Also, for each vertex $v \in V(C)$, $\deg(v, \widehat{C})$ is odd iff $v \in Y$, which proves property 2. □

Lemma 4.6. *For some* $i \geq 1$*, let* W *denote the set of nodes on cycles* $C \in A_i$*. Two nodes of* W *are connected via a path in slice* i *iff they are connected via a path that consists only of edges belonging to cycles* C *in* A_i*.*

Proof. For two nodes $x, y \in W$, let P be the x-to-y path in slice i that uses the fewest edges not belonging to cycles $C \in A_i$. Assume for a contradiction that P contains some edge e that does not belong to a cycle $C \in A_i$. Let $\widehat{P}$ be the maximal subpath of P that contains e but whose internal nodes do not belong to W.

By Lemma 4.4, e is strictly enclosed by some cycle $C \in A_i$. Since no internal node of $\widehat{P}$ belongs to W, every edge of $\widehat{P}$ must be enclosed by the same cycle C. But then the endpoints of $\widehat{P}$ must belong to that same cycle C. Consequently, $\widehat{P}$ can be replaced by a subpath of C, contradicting the choice of P. □

Now we can prove Theorems 4.1 and 4.2.

Proof of Theorem 4.1. Let M be the multiset of edges comprising the optimal tour of G. Then M is an Eulerian bisubgraph of G. Let M_i denote the submultiset of M consisting of edges in slice i. To prove Theorem 4.1, we will show that, for each slice i, there is a multiset D_i of edges of slice i such that $M_i \cup D_i$ is an Eulerian multisubgraph of slice i, i.e.,

1. every node of slice i has even degree with respect to $M_i \cup D_i$, and
2. for every connected component K of slice i, there is a corresponding connected component of $M_i \cup D_i$ that visits all nodes of K.

We ensure that $\sum_i weight(D_i) \leq 2\, weight(S)$.

We build D_i in three steps. The first two steps address achieving property 1. By Lemma 4.4, slice i consists of the edges enclosed by cycles of A_i and not strictly enclosed by cycles of A_{i+1}. If v belongs to no cycle in either A_i or A_{i+1}, then every edge of M incident to v belongs to M_i, so $\deg(v, M_i)$ is already even.

In step one, we address the case of nodes v belonging to cycles in A_i. For each cycle $C \in A_i$, we apply Lemma 4.5 to M and C, obtaining an edge-subset $\widehat{C} \subseteq E(C)$, and we include $\widehat{C}$ in D_i. This change affects the degree of a node v only if v belongs to some cycle $C \in A_i$. Such a node v has even degree with respect to those edges of $M \cup \widehat{C}$ that are enclosed in C. Summing over all cycles $C \in A_i$, the node has even degree with respect to those edges of $M \cup \bigcup\{\widehat{C} : C \in A_i\}$ that belong to slice i.

In step two, we address the case of nodes v belonging to cycles in A_{i+1}. Because the infinite face of a planar embedded graph can be chosen arbitrarily (and by definition of *enclosed*), we can apply Lemma 4.5 to each cycle $C \in A_{i+1}$ and to the set X of nodes *not* strictly enclosed by C, obtaining a set $\widehat{C}$ of edges such that $\deg(v, R - \Gamma_G(X) \cup \widehat{C})$ is even for each node v of C. We include each set $\widehat{C}$ in D_i. As before, this change affects the degree of a node v only if v belongs to some cycle $C \in A_{i+1}$, and it ensures that such a node v has even degree with respect to those edges of $M \cup \bigcup\{\widehat{C} \,:\, C \in A_{i+1}\}$ that belong to slice i.

In step three, we address property 2. For each $C \in A_i$, we add $E(C)$ to D_i. This does not change the parity of any node's degree. Let v_1 and v_2 be two nodes in slice i. For $j = 1, 2$, it follows from Lemma 4.4 that v_j is enclosed by some cycle $C_j \in A_i$. Let w be a node on the boundary of the infinite face. Let P_j be a v_j-to-w path using edges of M, and let $\widehat{P_j}$ be a maximal prefix of P_j consisting of edges enclosed by C_j. Since w is not strictly enclosed by C_j, $\widehat{P_j}$ must end at a node w_j of C_j.

Suppose v_1 and v_2 belong to the same connected component K of slice i. Then w_1 and w_2 also belong to K. By Lemma 4.6, there is a w_1-to-w_2 path using only edges of $\{E(C) \,:\, C \in A_i\}$ and hence using only edges of D_i. Combining this path with $\widehat{P_1}$ and $\widehat{P_2}$, we obtain a path using only edges of $M \cup D_i$ that belong to slice i. This proves property 2.

Finally, we bound $\sum_i \mathit{weight}(D_i)$. The cycles $C \in A_i$ consist of edges having level $t + (i-1)k$, so

$$\sum_i \sum \{\mathit{weight}(C) \,:\, C \in A_i\} = \mathit{weight}(S).$$

The weight added to $\bigcup D_i$ in each of steps one and two is at most $\sum_i \frac{1}{2} \sum\{\mathit{weight}(C) \,:\, C \in A_i\}$. The weight added in step three is $\sum_i \sum\{\mathit{weight}(C) \,:\, C \in A_i\}$. The total is at most $2\,\mathit{weight}(S)$. This completes the proof of Theorem 4.1. □

Proof of Theorem 4.2. Let K be a connected component of slice i where $i \geq 1$. By Lemmas 4.4, slice i can be obtained from G by (i) deleting edges properly enclosed by cycles of A_{i+1} and (ii) deleting edges not enclosed by cycles of A_i. For each cycle $C \in A_{i+1}$, deleting the edges properly enclosed by C merges the faces enclosed by C into a single face. Let D be the set of edges not enclosed by cycles of A_i. Deleting the edges in D corresponds in the planar dual G^* to compressing edges both of whose endpoints have levels at most $t + (i-1)k$. Let T be the breadth-first-search tree of G^*, and consider the effect of these operations on T. Recall that compressing a non-self-loop edge is equivalent to contracting. First, in the planar dual, compress all the edges of T that are in D. Because these edges form a subtree of T, none is a self-loop, so these compressions are contractions. For each (dual) node v having level at most $t + (i-1)k$, there is a path in T consisting of edges of D from v to the root, so the contractions merge all these nodes into a single node $\hat{r}$. Let $\widehat{T}$ be the set of edges of T that remain.

Each face of slice i that was a face of G had distance at most $t + ik$ from the root in T, and hence has distance at most $(t + ik) - (t + (i-1)k)$ from the root $\hat{r}$ in $\widehat{T}$. Each face arising from deleting edges properly enclosed by cycles of A_{i+1} is adjacent in the dual to some node that had been at level $t + ik$, and hence has distance at most $k + 1$ from $\hat{r}$.

Each of the remaining edges of D is now a self-loop with common endpoint $\hat{r}$. Compressing these edges in the dual might in general split $\hat{r}$ into multiple nodes corresponding to multiple connected components in the primal graph. However, each connected component retains its own low-depth spanning tree. This completes the proof of Theorem 4.2. □

5. TSP algorithm.

Now we describe the TSP algorithm.

Let G_0 be the input planar embedded graph, and let $weight(\cdot)$ be the input edge-weight assignment.

Step 1 *(spanner step).* Let ϵ_0 be the desired accuracy. Define $\epsilon = \epsilon_0/2$. Obtain a subgraph G of G_0 that has properties S1 and S2 of section 1.3.

Step 2 *(slicing step).* Use breadth-first search in the planar dual to find the slices as described in section 4, with $k = 2\epsilon^{-1}\rho_\epsilon$, where ρ_ϵ is the multiplier in property S1. For each slice, for each connected component of that slice, the planar dual has a spanning tree of depth at most $k + 1$.

Step 3 *(dynamic programming).* For each slice, for each connected component of that slice, find a minimum-weight Eulerian multisubgraph of that component.

Step 4. Combine the multisubgraphs to obtain an Eulerian multisubgraph of G, then turn it into a tour of G.

5.1. Running time.

Let n be the number of nodes in the input graph G_0. Assume G_0 has no parallel edges, so it has $O(n)$ edges. For unit-weight graphs, Step 1 is trivial: $G = G_0$ and ρ_ϵ is a constant. For arbitrary weights, Theorem 3.3 gives an $O(n)$ algorithm achieving $\rho_\epsilon = 1 + 2\epsilon^{-1}$. Steps 2 and 4 take $O(n)$ time.

As for Step 3, Cook and Seymour observe [15] that TSP can be solved in a graph of bounded branchwidth. In section 7, we state a theorem, due to Tamaki [45], that shows that each slice has branch-width at most $2k + 3$.

Because Cook and Seymour do not formally describe or analyze their dynamic program, in section 6 we describe a dynamic program that can be used in Step 3. This dynamic program exploits planarity to get a running time of $O(c^k n')$ for a graph of size n' (where c is a constant). Summing over all connected components of all slices, the running time for Step 3 is $O(c^k n)$. The choice of k yields a running time of $O(d^{1/\epsilon}n)$ for unit-weight graphs and $O(d^{1/\epsilon^2}n)$ for arbitrary weights, where d is a constant.

5.2. Correctness.

THEOREM 5.1. *The algorithm finds a tour of weight at most* $(1 + \epsilon_0)OPT(G_0)$.

Proof. The tours found in Step 3 are connected and jointly visit all nodes, so their union is connected and spans all nodes. Every node v has even degree with respect to every tour that contains it, so v has even degree with respect to the multiset union of these tours. Thus the multiset union is Eulerian. The Eulerian characterization (Proposition 2.8) implies that the union can be transformed into a tour.

The weight of the tour is $\sum_i \mathrm{OPT}(\text{slice } i)$, which by Theorem 4.1 is at most $\mathrm{OPT}(G) + 2\,weight(S)$. To complete the proof of Theorem 5.1, we bound these two terms. Property S2 states that $\mathrm{OPT}(G) \leq (1 + \epsilon)\mathrm{OPT}(G_0)$. Observe that

$$\begin{aligned} 2\,weight(S) &\leq (2/k)\,weight(G) && \text{by (1)} \\ &\leq \epsilon\rho_\epsilon^{-1} weight(G) && \text{by choice of } k \\ &\leq \epsilon \cdot \mathrm{OPT}(G_0) && \text{by property S1.} \end{aligned}$$

Since $\epsilon_0 = 2\epsilon$, this completes the proof of Theorem 5.1. □

6. Solving TSP in a planar embedded graph with bounded dual radius.

In this section we describe an algorithm that, given an edge-weighted planar embedded graph H, a low-depth spanning tree of H^*, and a set R of nodes, finds a minimum-weight walk W such that $R \subseteq V(W)$. To find an optimal tour, R is set to $V(H)$.

Rather than describe the algorithm for this special case, we describe the algorithm for the more general case because doing so requires very little change.

THEOREM 6.1. *There is an algorithm that, given a planar embedded graph H without parallel edges, an edge-weight assignment for H, a subset R of nodes of H, and a spanning tree T^* of H^* in which every simple path has length at most ℓ, finds a minimum-weight connected even multisubgraph of H that visits all nodes in R. The algorithm takes time $O(c^\ell |V(H)|)$ for some constant c.*

First we show how to reduce the problem to the case in which the degree of the input graph is bounded by three. Then we show how to solve this case using dynamic programming.

6.1. Reduction to degree three.

LEMMA 6.2. *Let H be a graph, let W be an even connected multisubgraph of H, and let e be an edge of H. Then $W - \{e\}$ is an even connected multisubgraph of $H/\{e\}$.*

Proof. Since the edges of W are connected in H, the edges of $W - \{e\}$ are connected in $H/\{e\}$. For every node v that is not an endpoint of e in H, the degree of v in $H/\{e\}$ with respect to $W - \{e\}$ equals the degree of v in H with respect to W. Let u_1 and u_2 be the endpoints of e in H. These nodes are coalesced in $H/\{e\}$ to form a single node whose degree with respect to $W - \{e\}$ is

$$\sum_{i=1}^{2}(\deg_H(u_i, W) - |W \cap \{e\}|) = \deg_H(u_1, W) + \deg_H(u_2, W) - 2|W \cap \{e\}|.$$

Since each of the terms on the right-hand side is even, the sum is even. □

LEMMA 6.3. *Let H be a graph, let e be an edge of H, and let W be an even connected multisubgraph of $H/\{e\}$. Then one of W, $W \cup \{e\}$, $W \cup \{e\} \cup \{e\}$ is an even connected multisubgraph of H.*

Proof. The proof is trivial if e is a self-loop. Otherwise, let u_1 and u_2 be the endpoints of e in H. These nodes are coalesced in $H/\{e\}$ to form a node v. Since $\deg_{H/\{e\}}(v, W)$ is even, $\sum_{i=1}^{2} \deg_H(u_i, W)$ is even.

Case 1: $\deg_H(u_1, W)$ is odd. Then $\deg_H(u_2, W)$ is also odd, and there are edges in W incident to u_1. Hence $\deg_H(u_i, W \cup \{e\})$ is even for $i = 1$ and 2, and $W \cup \{e\}$ is connected.

Case 2: $\deg_H(u_1, W)$ is even but at least one edge of W is incident to u_1 or u_2. Then $\deg_H(u_2, W)$ is also even, so $\deg_H(u_i, W \cup \{e\} \cup \{e\})$ is even for $i = 1$ and 2, and $W \cup \{e\} \cup \{e\}$ is connected.

Case 3: No edge of W is incident to u_1 or u_2. Then in $H/\{e\}$ no path in W passes through v, so the fact that W is connected in $H/\{e\}$ implies that W is connected in H. Clearly $\deg_H(u_i, W)$ is even for $i = 1$ and 2. □

Now we give the reduction to the degree-three case.

Step 1. Triangulate the faces of H^* by adding zero-weight artificial edges until every face has size at most three. Let A be the set of artificial edges added. Let $\hat{H}^*$ be the resulting planar embedded graph.

Step 2. H can be obtained from $\hat{H}$ by contracting the artificial edges, which merges some nodes. Let $\hat{R} = \bigcup_{v \in R}\{\text{nodes of } \hat{H} \text{ merged to form } v\}$.

Step 3. Let W be a minimum-weight connected even multisubgraph of $\hat{H}$ that visits all nodes of $\hat{R}$.

Step 4. Return $W - A$.

LEMMA 6.4. *$W - A$ is a minimum-weight even multisubgraph visiting all nodes of R.*

Proof. The proof is by repeated application of Lemma 6.3, using the fact that the artificial edges have zero weight. We infer $\mathrm{OPT}(\hat{H}) \leq \mathrm{OPT}(H)$. Therefore $weight(W) \leq \mathrm{OPT}(H)$. By repeated application of Lemma 6.2, we infer that $W - A$ is an even connected multisubgraph of H that visits all nodes of R. □

6.2. Overview of dynamic program. Now we describe how to find an optimal tour of $\hat{H}$ visiting all nodes of $\hat{R}$. The graph H^* has a rooted spanning tree T^* in which every simple path has at most ℓ edges, and $\hat{H}^*$ is obtained from H^* by adding edges, so T^* is also a spanning tree of $\hat{H}$. Because every face of $\hat{H}^*$ is a triangle, $\hat{H}$ has degree at most three. Let $\hat{T}$ be the set of edges of $\hat{H}$ not in T^*. Then $\hat{T}$ is a spanning tree of $\hat{H}$ and hence has degree at most three. Root $\hat{T}$ at a node r of degree 1 in $\hat{T}$. The dynamic program will work up $\hat{T}$ from the leaves to the root. For each edge of $\hat{T}$, the dynamic program will construct a table. The value of $\mathrm{OPT}(\hat{H})$ will be computed from the table associated with the edge connecting the root to its child. The dynamic program can be augmented to maintain enough information that the tour itself can be constructed in a postprocessing phase by working down from the root to the leaves. (The postprocessing is straightforward, and we do not describe it here.)

6.3. Terminology. Before giving a detailed description of the tables, we need to introduce some terminology.

Traversals. Let $\Gamma_{\hat{H}}(S)$ be a cut. We say a nonempty, dart-disjoint set $\mathcal{P}$ of walks in $\hat{H}$ is a *traversal of S in $\hat{H}$* if

- the start node and end node of each path are not in S,
- the internal nodes of each path are in S.

It follows that the first and last darts of each path belong to $\Gamma_{\hat{H}}(S)$, i.e., that each is a dart of some edge in $\Gamma_{\hat{H}}(S)$.

Define

$$\begin{aligned} weight(\mathcal{P}) = &\sum\{weight(d)\ : d \in \mathcal{P}, d \text{ not a dart of } \Gamma_{\hat{H}}(S)\} \\ &+\frac{1}{2}\sum\{weight(d)\ : d \in \mathcal{P}, d \text{ a dart of } \Gamma_{\hat{H}}(S)\}. \end{aligned}$$

Configurations. A *configuration K* of a cut $\Gamma_{\hat{H}}(S)$ is a nonempty set of ordered pairs $\langle d_i, d_j\rangle$ of darts of $\Gamma_{\hat{H}}(S)$ such that each dart occurs at most once in each position. The number of configurations is at most $(2\eta)!$, where $\eta = |\Gamma_{\hat{H}}(S)|$.

If S is connected in $\hat{H}$, then the embedding determines a cyclic ordering of the edges of $\Gamma_{\hat{H}}(S)$, say $(e_1 \ \cdots \ e_\eta)$. In this case, we say that a configuration is *crossing* if it includes a dart pair corresponding to the pair $\langle e_p, e_q\rangle$ of edges and also a dart pair corresponding to the pair $\langle e_r, e_s\rangle$ of edges, where $p < r < q < s$. A Catalan bound shows that the number of noncrossing configurations is $2^{O(\eta)}$. This is where planarity is used in the dynamic program.[10]

For a configuration K, define $weight(K)$ to be the sum of the weights of the darts in K.[11]

[10] For a discussion of Catalan numbers, see any text on combinatorics, e.g., [43]. Noncrossing configurations and a Catalan bound were used in a dynamic program for TSP by Arora et al. [5]. Concurrent with the appearance of a preliminary version of this paper, Dorn et al. [20] published an extended abstract discussing planar-graph algorithms that also exploited Catalan-type bounds and noncrossing matchings.

[11] The weight of a dart is the weight of the corresponding edge.

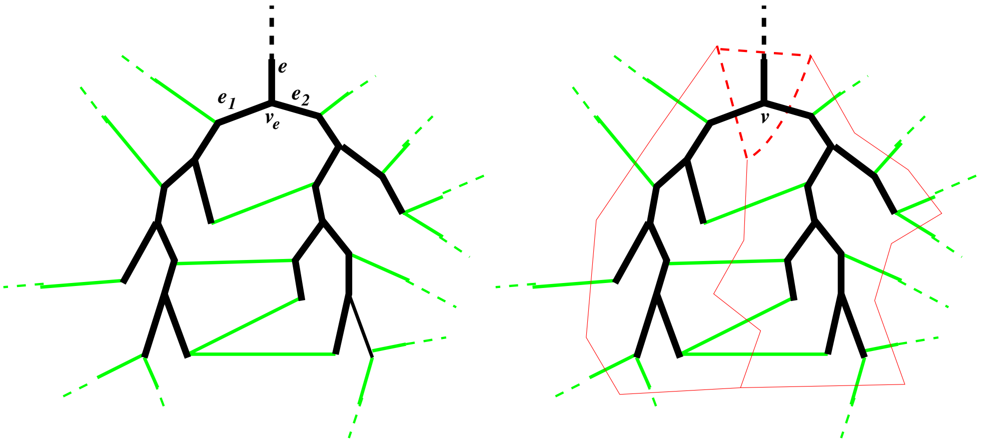


FIG. 8. *A subgraph arising in the dynamic program. The edge* e*, the child node* v_e*, and the child edges* e_1 *and* e_2 *are labeled. The dark edges are tree edges. On the right is the same subgraph with some edges of the dual graph also shown. Note that the edges of* $\Gamma(\{$*descendents of* $v_e\})$ *form an elementary cycle in the dual, as do the edges of* $\Gamma(\{$*descendents of* $e_1\})$ *and the edges of* $\Gamma(\{$*descendents of* $e_2\})$*.*

Let $C_1, \ldots, C_d$ be cuts in a graph, and let $K_1, \ldots, K_d$ be corresponding configurations. We say $K_1, \ldots, K_d$ are *consistent* if, for each pair C_i, C_j of cuts, each dart represented in both C_i and C_j occurs in both K_i and K_j or occurs in neither.

Define

$$\kappa(\mathcal{P}) = \{\langle \text{first dart of } P,\ \text{last dart of } P\rangle \ : \ P \in \mathcal{P}\}.$$

6.4. Definition of the tables. In this subsection we describe the tables produced by the dynamic program. For each edge e of $\hat{T}$, let v_e denote the child endpoint of e, and let D_e denote the descendents of v_e. By Lemma 2.4, the edges comprising $\Gamma_{\hat{H}}(D_e)$ are exactly the edges comprising the elementary cycle of e in $\hat{H}^*$ with respect to T^*. We denote this cycle by C_e. (See Figure 8.) That elementary cycle consists of e together with a simple path in T^* between the endpoints of e. The cycle therefore contains at most $\ell + 1$ edges. This shows $|\Gamma_{\hat{H}}(D_e)| \leq \ell + 1$.

For a cut $\Gamma_{\hat{H}}(S)$ of $\hat{H}$ where S is connected in $\hat{H}$, for a configuration K of $\Gamma_{\hat{H}}(S)$, define

$$\begin{aligned} M_S(K) = \min\{weight(\mathcal{P}) \ : \quad & \kappa(\mathcal{P}) = K, \\ & \mathcal{P} \text{ is a traversal of } S, \text{ and} \\ & S \cap \hat{R} \subseteq V(\mathcal{P})\}. \end{aligned}$$

We show in Corollary 6.8 that, for each edge e of T, the dynamic program will construct a table $\textsc{Tab}_e$, indexed by the noncrossing configurations K of $\Gamma_{\hat{H}}(D_e)$, such that $\textsc{Tab}_e[K] = M_{D_e}(K)$.

For the root edge $\hat{e}$ of T (the edge of T incident to r), each edge of $\Gamma_{\hat{H}}(D_{\hat{e}})$ is incident to the root r. It follows that every traversal of $D_{\hat{e}}$ defines a tour of $\hat{H}$ using each dart at most once, and vice versa. By Proposition 2.7, there is an optimal tour that is noncrossing. Hence

$$\text{OPT}(\hat{H}) = \min\left\{M_{\hat{e}}(K) + \frac{1}{2}weight(K) \ : \ K \text{ a configuration of } C_{\hat{e}}\right\}$$

because only half the weight of each edge of K is counted in $M_{\hat{e}}(K)$. Since r has degree at most three, $C_{\hat{e}}$ has $O(1)$ configurations. Thus $\mathrm{OPT}(\hat{H})$ can be computed in $O(1)$ time from the table $\textsc{tab}_{\hat{e}}$.

6.5. The recurrence relation. Let e be an edge of the tree $\hat{T}$, and let $e_1, \ldots, e_s$ be its child edges ($s \leq 2$). Let $D_0 = \{v_e\}$. For $i = 1, \ldots, s$, let $D_i = D_{e_i}$. Note that D_e is the disjoint union $\bigcup_{i=0}^{s} D_i$. For $i = 0, 1, \ldots, s$, let C_i denote $\Gamma_{\hat{H}}(D_i)$.

A traversal $\mathcal{P}$ of D_e *induces* a traversal $\mathcal{P}_i$ of D_i for $i = 0, 1 \ldots, s$ as follows: for each path $P \in \mathcal{P}$, break P into subpaths at the nodes of P that are not in D_i, and retain only those darts d such that at least one endpoint of d is in D_i. The remaining subpaths form a traversal of D_i. The following lemma is immediate.

LEMMA 6.5. *Let $\mathcal{P}$ be a traversal of D_e, and let $\mathcal{P}_0, \ldots, \mathcal{P}_s$ be the traversals that $\mathcal{P}$ induces for $D_0, \ldots, D_s$. Then*

$$\mathit{weight}(\mathcal{P}) = \sum_{i=0}^{s} \mathit{weight}(\mathcal{P}_i)$$

and $\kappa(\mathcal{P}), \kappa(\mathcal{P}_0), \kappa(\mathcal{P}_1), \ldots, \kappa(\mathcal{P}_s)$ are consistent.

LEMMA 6.6. *For traversals $\mathcal{P}_0, \ldots, \mathcal{P}_s$ of $D_0, \ldots, D_s$, if K is a configuration of C_e such that $K, \kappa(\mathcal{P}_0), \ldots, \kappa(\mathcal{P}_s)$ are consistent, then there is a traversal $\mathcal{P}$ of C_e that induces $\mathcal{P}_0, \ldots, \mathcal{P}_s$ such that $\kappa(\mathcal{P}) = K$.*

Proof. By gluing together paths from different $\mathcal{P}_i$'s that have a common dart, one constructs paths whose start and end darts are in $\Gamma(D_e)$. The consistency condition ensures that the gluing can be completed and that the start and end darts are represented in K. □

COROLLARY 6.7. *For any configuration K of C_e,*

$$M_{D_e}(K) = \min \left\{ \sum_{i=0}^{s} M_{D_i}(K_i) \ : \ K, K_0, \ldots, K_s \textit{ are consistent} \right\}.$$

Proof. To show that the left-hand side is at most the right-hand side, fix consistent configurations $K, K_0, \ldots, K_s$. For $i = 0, \ldots,$ let $\mathcal{P}_i$ be the traversal achieving the minimum in the definition of $M_{D_i}(K_i)$. (If there is no such traversal, the right-hand side is infinity.) By Lemma 6.6, there is a traversal $\mathcal{P}$ of D_e that induces $P_0, \ldots, P_s$. It follows that $D_e \cap \hat{R} \subseteq V(\mathcal{P})$.

By the first part of Lemma 6.5, $\mathit{weight}(\mathcal{P}) = \sum_{i=0}^{s} \mathit{weight}(\mathcal{P}_i)$, so $M_{D_e}(K) \leq \sum_{i=0}^{s} M_{D_i}(K_i)$.

To show that the right-hand side is at most the left-hand side, let K be a configuration such that $M_{D_e}(K)$ is finite, and let $\mathcal{P}$ be the traversal achieving the minimum in the definition of $M_{D_e}(K)$. Let $\mathcal{P}_0, \ldots, \mathcal{P}_s$ be the traversals that $\mathcal{P}$ induces for $D_0, \ldots, D_s$. It follows from Lemma 6.5 that the right-hand side is at most $\mathit{weight}(\mathcal{P})$. □

6.6. The dynamic program. We now give a recursive algorithm TSP-DP(e) that for each edge e of T populates the table $\textsc{Tab}_e$.

```
define TSP-DP($e$):
let $e_1, \ldots, e_s$ be the child edges of $e$ ($s \leq 2$)
for $i = 1, \ldots, s$,
    recursively call TSP-DP($e_i$).
initialize each entry of $\textsc{Tab}_e$ to $\infty$.
```

```
5 for each consistent tuple $(K, K_0, K_1, \ldots, K_s)$
        of configurations of $\Gamma(D_e), \Gamma(\{v_e\}), \Gamma(D_{e_1}), \ldots, \Gamma(D_{e_s})$
6     $\textsc{Tab}_e[K] := \min\{\textsc{Tab}_e[K]$ ,
                  $M_{\{v_e\}}(K_0) + \sum_{i=1}^{k} \textsc{Tab}_{e_i}[K_i]\}$
```

Note that in step 6, $M_{\{v_e\}}(K_0)$ can be computed directly in $O(1)$ time. The correctness of the algorithm follows from Corollary 6.7 by induction.

COROLLARY 6.8 (correctness of TSP-DP). *For each edge e of T, for each noncrossing configuration K of C_e, $\textsc{Tab}_e[K] = M_{D_e}(K)$.*

6.7. Analysis of the dynamic program.

In step 5, each of the cuts $\Gamma(D_{e_1}), \ldots, \Gamma(D_{e_s})$ has size at most $\ell + 1$, so it has $O(c^\ell)$ configurations for a constant c. The cut $\Gamma(\{v_e\})$ has size at most three, and $s \leq 2$, so the number of tuples in step 5 is $O(d^\ell)$ for a constant d. Thus each invocation of TSP-DP requires $O(d^\ell)$ time. The number of invocations is $|V(\hat{H})| - 1$, so the entire dynamic program takes time $O(d^\ell |V(\hat{H})|)$. Combined with the reduction of subsection 6.1, this completes the proof of Theorem 6.1.

7. Achieving low branch-width by contracting edges.

Branch-width is a graph measure akin to (and within a constant factor of) tree-width. (We will review the definition presently.) Many computational graph problems that are NP-hard for general graphs can be solved for graphs with bounded branch-width. The approach used for TSP in this paper can be used for other problems as well. The purpose of this section is to present a result that facilitates broader application of the approach.

THEOREM 7.1. *There is a linear-time algorithm that, for any planar graph G and integer k, finds a decomposition $S_0, \ldots, S_{k-1}$ of the edges of G such that, for $i = 0, 1, 2, \ldots, k-1$, the graph obtained from G by contracting the edges of S_i has branch-width at most $2(k+2)$.*

The analogous theorem with contraction replaced by deletion was implicit in the work of Baker [6] and made explicit by Demaine, Hajiaghayi, and Karabayashi [18], who proved a version for H-minor free graphs.

An easy but useful consequence of Theorem 7.1 is as follows.

COROLLARY 7.2. *There is a linear-time algorithm, that, for any planar graph G, edge-weight assignment $weight(\cdot)$, and integer k, finds a set S of edges of weight at most $(1/k)weight(G)$ whose contraction yields a graph of branch-width at most $2(k+2)$.*

Before proving Theorem 7.1, we review the definition of branch-width given by Seymour and Thomas [42]. For a graph G and a set X of edges, $\partial(X)$ denotes the set of nodes v of G such that at least one edge incident to v is in X and at least one is not. Two sets *cross* if neither contains the other and they are not disjoint.

For a finite set $\mathcal{X}$, a *carving* of $\mathcal{X}$ is a family $\mathcal{C}$ of subsets of $\mathcal{X}$ such that

1. $\emptyset, \mathcal{X} \notin \mathcal{C}$,
2. no two members of $\mathcal{C}$ cross, and
3. $\mathcal{C}$ is maximal subject to 1 and 2.

Let G be a graph. Let $\mathcal{C}$ be a carving of $E(G)$. The branch-width of $\mathcal{C}$ in G is $\max_{X \in \mathcal{C}} |\partial(X)|$. The *branch-width* of G is the minimum, over all carvings $\mathcal{C}$ of $E(G)$, of the width of $\mathcal{C}$.

The following lemma is implicit in Baker's approach [6], and the idea has been used several times since then (e.g., [9, 22, 26]).

LEMMA 7.3 (thinning algorithm). *There is a linear-time algorithm that, for any planar embedded graph G and integer k, finds a decomposition of the edges into subsets*

$S_0, \dots, S_{k-1}$ such that, for $i = 0, 1, 2, \dots, k-1$, there is a planar embedded graph H_i with the following properties:

1. *H_i has the same edges as G.*
2. *Each connected component of H_i has a spanning tree of depth at most k.*
3. *$H_i - S_i = G - S_i$.*

Proof. Assume without loss of generality that G is connected. For some node r, define the *level* of a node v of G to be its breadth-first-search distance from r, i.e., the minimum number of edges in an r-to-v path in G. Define the *level* of an edge e to be i if one endpoint has distance i from r and the other endpoint has distance $i+1$.

For $i = 0, 1, \dots, k-1$, let S_i denote the set of edges e whose levels are congruent to i mod k.

For $i = 0, 1, \dots, k-1$ and for $j = 0, 1, 2, \dots$, let $H_i^{(j)}$ be the graph obtained from G by deleting all nodes at distances greater than $jk + i$ and contracting every edge e of the breadth-first-search tree T whose level is less than $(j-1)k + i$. The contractions coalesce into a single root all nodes at distances less than or equal to $(j-1)k + i$. (For $j = 0$, there is already a single root, namely r.) Since every node of G that remains in $H_i^{(j)}$ had distance at most $jk + i$ in G, it follows that (A) in $H_i^{(j)}$ every node has distance at most k from the root. Moreover, (B) the graph $H_i^{(j)} - \{\text{edges at level } (j-1)k + i \text{ in } G\}$ is exactly the subgraph of G induced by nodes with levels in $((j-1)k + i, jk + i]$.

Let H_i be the disjoint union of $H_i^{(0)}, H_i^{(1)}, H_i^{(2)}, \dots$. Property 2 follows from (A). Property 3 follows from (B). Property 1 follows from property 3 and the definition of S_i. □

We first give a technical lemma; then we state a result of Tamaki [45] and prove Theorem 7.1.

Lemma 7.4. *Let G be a planar embedded graph and let e be a self-loop in G. Every biconnected component of G except $\{e\}$ is a biconnected component of $G/\{e\}$.*

Proof. Let v be the common endpoint of e. Any path in G that contains e must pass through e, so the only simple cycle in G that contains e is the cycle consisting solely of e. Any path in G from an edge enclosed by e to an edge not enclosed by e must pass through v. Hence a simple cycle C in G cannot include both an edge strictly enclosed by e and an edge not enclosed by e. Assume without loss of generality that C consists only of edges strictly enclosed by e. Any two such edges incident to a common node in G are also incident to a common node in $G/\{e\}$, which proves that C is a simple cycle in $G/\{e\}$. □

For a planar embedded graph G, Tamaki [45] defined $VF(G)$ to be the node-face incidence graph, i.e., the planar embedded bipartite graph whose node set is the union of the node set of G and the face set of G and where there is an edge between a node and a face if the node is on the boundary of the face. Tamaki proved the following theorem.

Theorem 7.5 (Tamaki). *There is a linear-time algorithm that, given a planar embedded graph G and a rooted spanning tree T of $VF(G)$, outputs a branch-decomposition of G whose width is at most the height of T.*

Proof of Theorem 7.1. Apply Lemma 7.3 to G^* to find a decomposition of the edges into subsets $L_0, \dots, L_{k-1}$ such that, for each i, there is a planar embedded graph H_i^* such that $H_i^* - L_i = G^* - L_i$ and each connected component of H_i^* has a spanning tree of depth at most k. The nodes of H_i^* are faces of H_i, so $VF(H_i)$ has a spanning tree of depth at most $2k+1$. By Tamaki's result, therefore, H_i has branch-width at most $2k+1$. It is known [41] that contraction does not increase branch-width.

Hence H_i/L_i has branch-width at most $2k+1$. Since $H_i^* - L_i = G^* - L_i$, we have $H_i/L_i = G/L_i$, so we have shown that G/L_i has branch-width at most $2k+1$.

Consider the process of obtaining G/L_i by compressing the edges of L_i one by one in an order that postpones compressing self-loops until only self-loops remain to be compressed. (Recall that a self-loop corresponds in the dual to cut-edges, so this corresponds in the dual to first deleting only non-cut-edges.)

Let S_i be the set of edges e of L_i such that e was not a self-loop when it was compressed, and let L_i' be the remaining edges of L_i. Then G/S_i is obtained from G by *contracting* the edges of S_i, and G/L_i is obtained from G/S_i by compressing the edges of L_i'. By Lemma 7.4, every biconnected component of G/S_i is a biconnected component of G/L_i. Since the branch-width of a graph is at most one more than the maximum of the branch-widths of its biconnected components (assuming at least one such component has more than one edge), it follows that that branch-width of G/S_i is at most one plus the branch-width of G/L_i. □

Acknowledgments. Claire Kenyon and Mikkel Thorup contributed greatly to the quality of the presentation through their criticisms and suggestions. Thanks also to Glencora Borradaile and Erik Demaine for helpful discussions and suggestions.

REFERENCES

[1] I. Althöfer, G. Das, D. Dobkin, D. Joseph, and L. Soares, *On sparse spanners of weighted graphs*, Discrete Comput. Geom., 9 (1993), pp. 81–100.

[2] S. Arora, *Polynomial time approximation schemes for Euclidean TSP and other geometric problems*, J. ACM, 45 (1998), pp. 753–782.

[3] S. Arora, *Polynomial time approximation schemes for Euclidean TSP and other geometric problems*, in Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science, Burlington, VT, 1996, pp. 2–11.

[4] S. Arora, *Nearly linear time approximation schemes for Euclidean TSP and other geometric problems*, in Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science, Miami, FL, 1997, pp. 554–563.

[5] S. Arora, M. Grigni, D. R. Karger, P. N. Klein, and A. Woloszyn, *A polynomial-time approximation scheme for weighted planar graph TSP*, in Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (San Francisco, CA), SIAM, Philadelphia, ACM, New York, 1998, pp. 33–41.

[6] B. Baker, *Approximation algorithms for NP-complete problems on planar graphs*, J. ACM, 41 (1994), pp. 153–180.

[7] A. Berger, A. Czumaj, M. Grigni, and H. Zhao, *Approximate minimum 2-connected subgraphs in weighted planar graphs*, in Proceedings of the 13th Annual European Symposium on Algorithms, Mallorca, Spain, 2005, pp. 472–483.

[8] A. Berger and M. Grigni, *Minimum weight 2-edge-connected spanning subgraphs in planar graphs*, in Proceedings of the 34th International Colloquium on Automata, Languages and Programming, 2007, pp. 90–101.

[9] H. L. Bodlaender, *Some classes of graph with bounded treewidth*, Bull. Eur. Assoc. Theoret. Comput. Sci. EATCS, 36 (1988), pp. 116–126.

[10] G. Borradaile, C. Kenyon-Mathieu, and P. Klein, *A polynomial-time approximation scheme for Steiner tree in planar graphs*, in Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (New Orleans, LA), SIAM, Philadelphia, ACM, New York, 2007, pp. 1285–1294.

[11] G. Borradaile, P. Klein, and C. Mathieu, *Steiner tree in planar graphs: An $O(nlogn)$ approximation scheme with singly exponential dependence on epsilon*, in Proceedings of the 10th International Workshop on Algorithms and Data Structures, Lecture Notes in Comput. Sci. 4619, Springer, Berlin, 2007, pp. 275–286.

[12] J. M. Boyer and W. J. Myrvold, *Simplified planarity*, J. Graph Algorithms Appl., 8 (2004), pp. 241–273.

[13] D. R. Cheriton and R. E. Tarjan, *Finding minimum spanning trees*, SIAM J. Comput., 5 (1976), pp. 724–742.

[14] N. Christofides, *Worst-case analysis of a new heuristic for the traveling salesman problem*, in Symposium on New Directions and Recent Results in Algorithms and Complexity, J. F. Traub, ed., Academic Press, New York, 1976, p. 441.

[15] W. J. Cook and P. D. Seymour, *Tour merging via branch-decomposition*, INFORMS J. Comput., 15 (2003), pp. 233–248.

[16] A. Czumaj, M. Grigni, P. Sissokho, and H. Zhao, *Approximation schemes for minimum 2-edge-connected and biconnected subgraphs in planar graphs*, in Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (New Orleans, LA), SIAM, Philadelphia, ACM, New York, 2004, pp. 496–505.

[17] E. D. Demaine and M. Hajiaghayi, *Bidimensionality: New connections between FPT algorithms and PTASs*, in Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (Vancouver, BC), SIAM, Philadelphia, ACM, New York, 2005, pp. 590–601.

[18] E. D. Demaine, M. T. Hajiaghayi, and K. Karabayashi, *Algorithmic graph minor theory: Decomposition, approximation, and coloring*, in Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, Pittsburgh, PA, 2005, pp. 637–646

[19] E. D. Demaine, M. Hajiaghayi, and B. Mohar, *Approximation algorithms via contraction decomposition*, in Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (New Orleans, LA), SIAM, Philadelphia, ACM, New York, 2007, pp. 278–287.

[20] F. Dorn, E. Penninkx, H. L. Bodlaender, and F. V. Fomin, *Efficient exact algorithms on planar graphs: Exploiting sphere-cut branch decompositions*, in Proceedings of the 13th Annual European Symposium on Algorithms, Mallorca, Spain, 2005, pp. 95–106.

[21] J. R. Edmonds, *A combinatorial representation for polyhedral surfaces*, Notices Amer. Math. Soc., 7 (1960), p. 646.

[22] D. Eppstein, *Subgraph isomorphism in planar graphs and related problems*, J. Graph Algorithms Appl., 3 (1999), pp. 1–27.

[23] M. Grigni, E. Koutsoupias, and C. H. Papadimitriou, *An approximation scheme for planar graph TSP*, in Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science, Los Alamitos, CA, 1995, pp. 640–645.

[24] M. Grigni and P. Sissokho, *Light spanners and approximate TSP*, in Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (San Francisco, CA), SIAM, Philadelphia, ACM, New York, 2002, pp. 852–857.

[25] M. Grigni, *Approximate TSP in graphs with forbidden minors*, in Proceedings of the 27th International Colloquium on Automata, Languages, and Programming, Geneva, Switzerland, 2000, pp. 869–877.

[26] M. Grohe, *Local tree-width, excluded minors, and approximation algorithms*, Combinatorica, 23 (2003), pp. 613–632.

[27] L. Heffter, *Über das problem der nachbargebiete*, Math. Ann., 38 (1891), pp. 477–508.

[28] M. R. Henzinger, P. N. Klein, S. Rao, and S. Subramanian, *Faster shortest-path algorithms for planar graphs*, J. Comput. System Sci., 55 (1997), pp. 3–23.

[29] J. Hopcroft and R. E. Tarjan, *Efficient planarity testing*, J. ACM, 21 (1974), pp. 549–568.

[30] P. N. Klein, *A linear-time approximation scheme for planar weighted TSP*, in Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, Pittsburgh, PA, 2005, pp. 647–657.

[31] P. N. Klein, *A subset spanner for planar graphs, with application to subset TSP*, in Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, 2006, pp. 749–756.

[32] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, eds., *The Traveling Salesman Problem*, John Wiley, Chichester, UK, 1985.

[33] D. Marx, *On the optimality of planar and geometric approximation schemes*, in Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science, Providence, RI, 2007, pp. 338–348.

[34] J. S. B. Mitchell, *Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric k-MST, TSP, and related problems*, SIAM J. Comput., 28 (1999), pp. 1298–1309.

[35] B. Mohar and C. Thomassen, *Graphs on Surfaces*, The Johns Hopkins University Press, Baltimore, MD, 2001.

[36] C. Papadimitriou and M. Yannakakis, *Optimization, approximation and complexity classes*, J. Comput. System Sci., 43 (1991), pp. 425–440.

[37] C. Papadimitriou and M. Yannakakis, *The traveling salesman problem with distances one and two*, Math. Oper. Res., 18 (1993), pp. 1–11.

[38] S. B. Rao and W. D. Smith, *Approximating geometrical graphs via "spanners" and "banyans,"* in Proceedings of the 30th Annual ACM Symposium on Theory of Computing, Dallas, TX,

1998, pp. 540–550.

[39] N. Robertson and P. D. Seymour, *Graph minors.* III. *Planar tree-width*, J. Combin. Theory Ser. B, 36 (1984), pp. 49–64.

[40] N. Robertson and P. D. Seymour, *Graph minors.* IV. *Tree-width and well-quasi-ordering*, J. Combin. Theory Ser. B, 48 (1990), pp. 227–254.

[41] N. Robertson and P. D. Seymour, *Graph minors.* X. *Obstructions to tree-decomposition*, J. Combin. Theory Ser. B, 52 (1991), pp. 153–190.

[42] P. D. Seymour and R. Thomas, *Call routing and the ratcatcher*, Combinatorica, 14 (1994), pp. 217–241.

[43] R. P. Stanley, *Enumerative Combinatorics*, Vol. 1, Cambridge University Press, Cambridge, UK, 2000.

[44] K. Talwar, *Bypassing the embedding: Algorithms for low-dimensional metrics*, in Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, 2004, pp. 281–290.

[45] H. Tamaki, *A linear-time heuristic for the branch-decomposition of planar graphs*, in Proceedings of the 11th Annual European Symposium on Algorithms, Budapest, Hungary, 2003, pp. 765–775.

[46] J. W. T. Youngs, *Minimal imbeddings and the genus of a graph*, J. Math. Mech., 12 (1963), pp. 303–315.